LISTING UNIQUE FRACTIONAL FACTORIAL DESIGNS

A Dissertation

by

ABHISHEK KUMAR SHRIVASTAVA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2009

Major Subject: Industrial Engineering

LISTING UNIQUE FRACTIONAL FACTORIAL DESIGNS

A Dissertation

by

ABHISHEK KUMAR SHRIVASTAVA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Yu Ding
Committee Members,	Sara A. McComb
	Sergiy Butenko
	Bani K. Mallick
	Jianhua Huang
Head of Department,	Brett A. Peters

December 2009

Major Subject: Industrial Engineering

ABSTRACT

Listing Unique Fractional Factorial Designs. (December 2009)

Abhishek Kumar Shrivastava, B. Tech. (Hons.), Indian Institute of Technology Kharagpur

Chair of Advisory Committee: Dr. Yu Ding

Fractional factorial designs are a popular choice in designing experiments for studying the effects of multiple factors simultaneously. The first step in planning 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 a 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 are isomorphic to each other if one can be obtained from the other by some relabeling of factor labels, level labels of each factor and reordering of runs. Clearly, two isomorphic designs are statistically equivalent. Design catalogs should therefore contain only designs unique up to isomorphism.

There are two computational challenges in generating such catalogs. Firstly, testing two designs for isomorphism is computationally hard due to the large number of possible relabelings, and, secondly, the number of designs increases very rapidly with the number of factors and run-size, making it impractical to compare all designs for isomorphism. In this dissertation we present a new approach for tackling both these challenging problems. We propose graph models for representing designs and use this relationship to develop efficient algorithms. We provide a new efficient isomorphism check by modeling the fractional factorial design isomorphism problem as graph isomorphism problem. For generating the design catalogs efficiently we extend

a result in graph isomorphism literature to improve the existing sequential design catalog generation algorithm.

The potential of the proposed methods is reflected in the results. For 2-level regular fractional factorial designs, we could generate complete design catalogs of run sizes up to 4096 runs, while the largest designs generated in literature are 512 run designs. Moreover, compared to the next best algorithms, the computation times for our algorithm are 98% lesser in most cases. Further, the generic nature of the algorithms makes them widely applicable to a large class of designs. We give details of graph models and prove the results for two classes of designs, namely, 2-level regular fractional factorial designs and 2-level regular fractional factorial split-plot designs, and provide discussions for extensions, with graph models, for more general classes of designs.

To my parents

ACKNOWLEDGMENTS

I find myself fortunate to have had Dr. Yu Ding as my PhD advisor. I thank him sincerely for mentoring me for my able academic and professional development. I have come to learn a lot from him over the years and will always admire him for his sharp intellect and will remain grateful for his patience and support.

I would like to express my gratitude towards my committee members, Dr. Sara A. McComb, Dr. Sergiy Butenko, Dr. Jianhua Huang and Dr. Bani K. Mallick, for their constant encouragement and support. Additionally, my sincere thanks go to the faculty and staff in the Department of Industrial and Systems Engineering for providing an intellectually stimulating and congenial environment for learning and research. A special vote of thanks goes to Judy Meeks for always being so helpful in attending to all problems in a timely manner while being ever so busy.

I am thankful to all my labmates, especially Haifeng (Heidi) Xia, Jung-Jin Cho, Eunshin Byun, Chiwoo Park and Chang-Ho Chin, for the very many interesting discussions and debates and for making the lab a pleasant workplace. Outside my lab, my friends have ensured that my time was equally enjoyable. My thanks go to Vandana and Homarjun Agrahari, Richa and Ajay Singh, Karambir Kalsi, Balabhaskar (Baski) Balasundaram, Gregory Fernandes, Lokesh Kulkarni, Rajaneesh Jandhyam, Sandeep Sachdeva, Vishal Karnik and many more, for their support, the interesting academic and non-academic debates, and all the good times.

Last, but not least, my deepest gratitude and regards go to my mother, my late father and my sister. They have been a strong force in my life and the reason that I am here. I thank them for their unconditional support, belief, love and affection. My earnest affection goes to my beautiful niece Sharanya for the numerous smiles she has gifted me in a very short time.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

TABLE Page

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

Technological advancements have lead to increased complexity in engineering systems over the last few decades. These have significantly improved productivity and quality of delivered products and services but at the cost of new challenges in the tasks of modeling and analysis of system-level behavior or phenomena. The study of such large-scale systems usually involves conducting experiments involving large number of variables and analyzing the collected data for discovering relationships. The subject area of design of experiments provides a wide array of designs for effectively conducting experiments and analyzing collected data. Yang and Speed (2002), for instance, provide a good discussion on the potential of experimental designs in designing DNA microarray experiments, a relatively recent area of research.

Among the various classes of designs, fractional factorial designs are among the most popular, if not *the* most popular (see Bisgaard (1992), Ilzarbe *et al.* (2008), Prvan and Street (2002) and citations within for over 250 case studies), class of designs used in the fields of science and engineering. Box and Hunter (1961a, 2000) give a good discussion of the different circumstances in which factorial designs may be a good choice. Fractional factorial designs are most commonly used when it is known that only a few effects will be significant. This is usually the case for screening experiments, where the task is to find the small number of significant factors (i.e., variables) from a large collection. Recently, fractional factorial designs with large run sizes have been reported, for example, of over 600 runs in Lin and Sitter (2008) and of 4096 runs in Mee (2004). The methodology developed in this dissertation aids the

---

The journal model is *IIE Transactions.*

	A	B	C	D	E	F	G
1	0	0	0	0	0	0	0
2	0	0	0	1	0	0	1
3	0	0	1	0	0	1	0
4	0	0	1	1	0	1	1
5	0	1	0	0	1	0	1
6	0	1	0	1	1	0	0
7	0	1	1	0	1	1	1
8	0	1	1	1	1	1	0
9	1	0	0	0	1	1	0
10	1	0	0	1	1	1	1
11	1	0	1	0	1	0	0
12	1	0	1	1	1	0	1
13	1	1	0	0	0	1	1
14	1	1	0	1	0	1	0
15	1	1	1	0	0	0	1
16	1	1	1	1	0	0	0

(a) Defining words: $\{ABE$, ACF, $BDG\}$

	A	B	C	D	E	F	G
1	0	0	0	0	0	0	0
2	0	0	0	1	0	0	1
3	0	0	1	0	0	1	1
4	0	0	1	1	0	1	0
5	0	1	0	0	1	0	0
6	0	1	0	1	1	0	1
7	0	1	1	0	1	1	1
8	0	1	1	1	1	1	0
9	1	0	0	0	1	1	0
10	1	0	0	1	1	1	1
11	1	0	1	0	1	0	1
12	1	0	1	1	1	0	0
13	1	1	0	0	0	1	0
14	1	1	0	1	0	1	1
15	1	1	1	0	0	0	1
16	1	1	1	1	0	0	0

(b) Defining words: $\{ABE$, ACF, $CDG\}$

	A	C	B	D	F	E	G
1	0	0	0	0	0	0	0
2	0	0	0	1	0	0	1
5	0	0	1	0	0	1	1
6	0	0	1	1	0	1	0
3	0	1	0	0	1	0	0
4	0	1	0	1	1	0	1
7	0	1	1	0	1	1	1
8	0	1	1	1	1	1	0
9	1	0	0	0	1	1	0
10	1	0	0	1	1	1	1
13	1	0	1	0	1	0	1
14	1	0	1	1	1	0	0
11	1	1	0	0	0	1	0
12	1	1	0	1	0	1	1
15	1	1	1	0	0	0	1
16	1	1	1	1	0	0	0

(c) Reordered matrix of (a). Relabeling $B \leftrightarrow C$, $E \leftrightarrow F$ and rows gives (b)

Fig. 1. Example of two isomorphic 7-factor designs. Design matrices of two 2^{7-3} designs are shown in (a) and (b). (c) shows that (a) and (b) are isomorphic designs.

experimenter in searching for an appropriate design for such experiments.

Fig. 1(a) shows the design table/matrix of a 7-factor design with 16 runs. In the design of experiments literature, the variables in an experiment are called *factors*

and each setting of a variable is called a *level*. The columns $A, B, \ldots, G$ in Fig. 1 represent the factors and each row is a run, also called a treatment combination, in the experimental design. The factors in this design have only two levels each, denoted by '0' and '1'. So each row gives the settings of the 7 factors in the corresponding run of the design. When an experiment is performed using such a design, the runs may be replicated and will (usually) be performed in random order. It may be noted that the design in Fig. 1(a) is a 2-level regular fractional factorial design, a particular type of fractional factorial design, but the design matrix representation shown is generic and can be used to present any type of fractional factorial designs.

The first step in planning 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 a small number of runs. By statistical properties we mean the main effects and the interaction effects between the factors (see Section II.1 for details on terminology) that can be estimated by analyzing the data collected after performing an experiment using the chosen design. A reasonable approach in selecting such a design is to start by selecting a (small) run-size for the design, for the given number of factors, and then look for a design that has the desired statistical properties. This requires that a catalog of candidate designs be available (or be possible to generate) for searching for the 'good' design. In this dissertation we present new efficient methods for generating catalogs of such large-size designs.

I.1. Unique designs

In the attempt to generate the catalog of candidate designs, the problem of design isomorphism must be addressed. Figs. 1(a) and 1(b) show two 2^{7-3} designs, i.e., 7-factor regular fractional factorial designs with each factor having two levels. The

columns $A, B, \ldots, G$ represent the factors and the rows are the treatment combinations (level settings of the factors at each run). Suppose that the two designs given in Figs. 1(a) and 1(b) are both in a catalog from which a design is to be chosen. Reordering the rows and columns of the design matrix in Fig. 1(a) gives the design matrix in Fig. 1(c). This design matrix is identical to the design matrix in Fig. 1(b) if we exchange the labels of factors B and C (simply represented by $B \leftrightarrow C$), and factors E and F (represented by $E \leftrightarrow F$), and relabel the rows, in the current order, from 1 to 16.

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. Therefore, the designs in Figs. 1(a) and 1(b) are isomorphic to each other. A natural question here is whether it makes sense to interchange the labels of two factors. The answer is yes, because, in a catalog, the factor labels have no physical meaning as they are not associated with any physical variable. Hence, all the factors labels are physically indistinguishable and can be interchanged. They become distinguishable only when they are associated with the physical variables of an experiment. In this dissertation, since we are interested in generating design catalogs, we will call a design *unique* in a collection of designs if no other design in the collection is isomorphic to this design. Further, we will call a collection of unique designs, i.e., a set where no two designs are isomorphic, a *non-isomorphic* collection.

A catalog of designs should contain only those designs that are unique or non-isomorphic to each other. This is so because the statistical properties of two isomorphic designs are the same. That is, the main effects and the interaction effects that can be estimated from two isomorphic designs, when the factor labels are physically indistinguishable, are the same. Thus, when choosing a design, keeping isomorphic

designs wastes the effort of the experimenter by presenting designs with the same statistical properties. Further, the size of the catalog can be greatly reduced by discarding isomorphs. The number of isomorphic designs can be very large even for moderate number of factors, levels and runs. For example, according to Chen *et al.* (1993), the total number of 2^{15-10} designs, i.e., 2-level regular fractional factorial designs with 15 factors and 32 ($= 2^{15-10}$) runs, is 5,311,735, but the number of unique or non-isomorphic designs of resolution ≥ 3 among these is only 144. Thus, keeping only non-isomorphic designs will reduce the experimenter's effort considerably as it will require comparing the statistical properties of a much smaller set of designs.

I.2. Issues in generating design catalogs

There are two primary issues in devising methods for generating non-isomorphic catalogs of fractional factorial designs. Firstly, the problem is computationally hard for any sub-class (type) of fractional factorial designs. Secondly, very many sub-classes of fractional factorial designs exist with somewhat differing mathematical structures making it difficult to have one method uniformly efficient for all types of fractional factorial designs.

I.2.1. Computational issues

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

The problem of testing two designs for isomorphism is computationally hard as the total number of relabelings is combinatorially large. For example, the total number of relabelings of a 2^{n-k} design, i.e., 2-level n-factor regular fractional factorial design, is $(n!)(2!)^n(2^{n-k}!)$. Further, the total number of these designs itself is also usually very large. In general, the total number of 2^{n-k} designs is $\binom{(2^{n-k}-1)-(n-k)}{k}$. So if we use the trivial approach of constructing the non-isomorphic catalog, by discarding isomorphs from the entire collection of designs using some isomorphism check, then we would be comparing a combinatorially large number of designs, where each comparison in itself is a costly one.

I.2.2. Complicated designs

Many subclasses of fractional factorial designs exist in literature. The major subclasses are regular fractional factorial designs (Montgomery, 2000, Wu and Hamada, 2000) and non-regular designs like orthogonal arrays (Hedayat *et al.*, 1999, Rao, 1947), including Plackett-Burman designs (Plackett and Burman, 1946). Regular fractional factorial designs have been the most popular, in practice, among these due to the relative ease in analyzing the experimental data and inferring the results. It may be noted that every regular fractional factorial design is an orthogonal array (while the opposite is not true).

Although all these various types of fractional factorial designs can be represented by a design matrix (as shown in Fig. 1(a)), the differing mathematical structures provide potential for developing subclass-specific algorithms that are more efficient than those developed for the general (non-regular) fractional factorial designs. For the problem of isomorphism testing, it can be shown that the general isomorphism checks, like that of Clark and Dean (2001), perform poorly for certain subclasses of designs, like 2-level regular fractional factorial designs which have alternative representations

(see Section II.1.4), when compared with methods developed specifically for this class, e.g., Lin and Sitter (2008)'s method (see Section VI.1).

The development of isomorphism checks for these various design classes has thus been done relatively independently. For example, Lin and Sitter (2008)'s isomorphism check, applicable only to 2-level regular fractional factorial designs, is based on a representation specific to regular fractional factorial designs. Even extending it to regular multi-level fractional factorial designs (i.e., designs having factors with levels more than two) is difficult (Lin and Sitter, 2008). Cheng and Ye (2004), Clark and Dean (2001), Sun *et al.* (2002) present isomorphism checks for general (i.e., both regular and non-regular) fractional factorial designs but there are no trivial extensions of these for, say, regular fractional factorial designs that would exploit their structure (see Section II.3.1 for an extension of Clark and Dean (2001)'s isomorphism check). Stufken and Tang (2007) present an isomorphism check for a specific subset of orthogonal arrays and note that the extension to the general class is non-trivial.

The generation algorithms for various subclasses of designs have also been developed quite independently for the same reason. Although these algorithms have a common sequential structure (cf. Bingham and Sitter (1999a), Chen *et al.* (1993) for regular fractional factorial designs, Sun *et al.* (2002) for Plackett-Burman designs, Angelopoulos *et al.* (2007), Schoen and Nguyen (2007) for orthogonal arrays) – larger designs are constructed from smaller designs, usually by adding a factor to the smaller design, the developments have been design class specific (like those in Bingham and Sitter (1999a), Chen *et al.* (1993) for regular fractional factorial designs) and cannot be trivially extended to other design classes.

A further extension of fractional factorial designs, which is of greater importance to practitioners, is the inclusion of blocking and randomization constraints in conducting experiments. These practical constraints have created variants of fractional

factorial designs that are usually derived from a regular or non-regular fractional factorial designs discussed above. But constraints are put on the randomization of the runs in the design matrix and/or blocking factors are used to capture the influence of extraneous variables. Examples of such designs are fractional factorial split-plot designs, split-split-plot designs, strip-split-plot designs (Montgomery, 2000, Chapter 13), split-lot designs (Mee and Bates, 1998) and blocked fractional factorial designs (Montgomery, 2000, Chapter 7), among others. Among these, regular fractional factorial split-plot designs have been the quite popular in industrial applications. These additional constraints further complicate the already hard problem of generating designs and, although of great interest, little has been done for generating these designs (Bingham and Sitter, 1999b, Butler, 2004, Kulahci and Bisgaard, 2005, McLeod and Brewster, 2004).

I.3. Research objectives and contributions

In this dissertation, we present a framework for efficiently constructing catalogs of non-isomorphic 2-level regular fractional factorial designs that is extensible to other classes of fractional factorial designs. In particular, we extend the framework to 2-level regular fractional factorial split-plot designs. The framework has two major components – the isomorphism check and a result for speeding the generation algorithm.

We provide a new approach for testing the isomorphism of fractional factorial designs by modeling them as graphs. The 2-level regular fractional factorial designs as bipartite graphs and the 2-level regular fractional factorial split-plot designs are modeled as vertex-colored 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 regular fractional factorial designs, we improve the existing sequential design generation algorithm (Lin and Sitter, 2008) 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 (McKay, 1998) to develop this reduction procedure. We also extend the result to the split-plot designs, and discuss extensions to other design classes (see Chapter VII).

The contributions of this dissertation are four-fold: (i) a new necessary and sufficient check for 2-level regular fractional factorial design isomorphism; (ii) a generation algorithm that can generate catalogs of non-isomorphic 2-level regular fractional factorial designs much faster than any of the previous methods; (iii) a unified framework to handle complicated designs; (iv) catalogs of up to 4096-run non-isomorphic 2-level regular fractional factorial and 2-level regular fractional factorial split-plot designs, not available earlier in literature.

The reason that our generation algorithm is faster can be understood as follows. Firstly, for removing isomorphs from a collection, we run the isomorphism check (which is usually the computationally expensive portion of a procedure for generating non-isomorphic designs) only once for each design and not once for each pair of designs. The only other existing method that does this is Lin and Sitter (2008)'s eigenvalue check. However, our graph-based method allows us to do the isomorphism check much faster than Lin and Sitter (2008). It should also be noted that Lin and Sitter (2008)'s eigenvalue check is not guaranteed to always distinguish two isomorphic designs. Other methods, e.g., Clark and Dean (2001), compare pairs of designs to determine whether they are isomorphic to each other or not. Secondly, because we model the problem as a graph isomorphism problem, we are able to reduce the size of

the collection of candidate designs, from which we need to eliminate isomorphs. Not only does our algorithm 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 any existing methods.

As we will see in Chapters V and VII, the isomorphism check can be extended to other classes of designs as long as we can construct a graph representation of these designs that obeys certain properties. We will present some examples of such graph representations. The generation algorithm, as noted earlier, is typically sequential for most classes of designs. It seems possible to extend the result developed (for 2-level regular fractional factorial designs) to reduce the number of candidate designs which need to be compared for isomorphism. This result is extended to the case of 2-level regular fractional factorial split-plot designs, and we later provide thoughts on extending it to other classes of fractional factorial designs.

I.4. Organization of this dissertation

The rest of this dissertation is organized as follows. Chapter II introduces the common terminology in fractional factorial designs and formally introduces the problem of fractional factorial design isomorphism. We then survey the various methods proposed in literature for the testing two designs for isomorphism. We specifically look at methods that are applicable to 2-level regular fractional factorial designs.

In Chapter III we present a new graph based isomorphism check for testing the isomorphism between two 2-level regular fractional factorial designs. In this chapter we propose a new graph model for modeling 2-level regular fractional factorial designs as graphs. We then transform the problem of design isomorphism to the problem of graph isomorphism and provide algorithmic details of the graph isomorphism algo-

rithm used for solving our design isomorphism problem.

Chapter IV will present the problem of efficiently generating design catalogs, given an isomorphism check. We will look at the sequential design generation procedure for 2-level regular fractional factorial designs and the methods proposed in literature for speeding up this procedure. We will then present a new method for further improving the computational efficiency of the existing design generation algorithm.

In Chapter V we will consider the problem of generating catalogs of 2-level regular fractional factorial split-plot designs. We will extend the methods developed in Chapters III and IV to efficiently generate these design catalogs.

Chapter VI presents the design catalogs that have been generated using the algorithms developed in this dissertation. We also provide comparisons of computational efficiency of generating 2-level regular fractional factorial designs with other methods.

We will conclude the dissertation in Chapter VII, giving insights into related research problems and discussions on extending the framework developed in this dissertation research to other classes of fractional factorial designs.

Since this dissertation is concerned only with the class of fractional factorial designs, in the remainder of this dissertation, whenever we write designs we will mean fractional factorial designs.

CHAPTER II

DESIGN ISOMORPHISM

In this chapter we formally introduce the problem of fractional factorial design isomorphism (Section II.2) and review the existing literature for solving this problem (Section II.3). But before that we introduce some definitions and terminology common in design of experiments and, in particular, fractional factorial designs literature, in the next section.

II.1. Some preliminaries on fractional factorial designs

We use an example from Wu and Hamada (2000, Section 4.1) to explain the concepts related to fractional factorial designs. We will refer to this example as the *leaf spring* example throughout the dissertation. The example has been modified from its presentation in Wu and Hamada (2000) to illustrate the concepts in fractional factorial designs relevant to this dissertation. Since the objective of this dissertation is to construct designs, the focus here is on highlighting the mathematical structure of the various designs and its implications to the ensuing data analysis. We thus provide no details on how the data is actually analyzed.

II.1.1. Example: leaf spring experiment

An experiment needs to be conducted to improve the heat treatment process that forms the curvature of a truck leaf spring. A leaf spring is a curved, rectangular cross-section steel bar (or connected layers of these) used in the suspension of heavy vehicles. The heat treatment process involves first heating in a high temperature furnace, followed by processing in a forming machine and finally quenching in an

Table 1. Leaf spring experiment – factors and levels (Wu and Hamada, 2000, Table 4.1, page 154)

Factor		Level	
Label	Description	0	1
A	furnace temperature (°F)	1840	1880
B	heating time (seconds)	23	25
C	transfer time (seconds)	10	12
D	hold-down time (seconds)	2	3
E	oil temperature (°F)	130–150	150–170

oil bath. The objective of this experiment is to determine settings of the process variables that will minimize the variation in the free height of the manufactured leaf springs. The free height of a leaf spring is the distance between the center point of the spring and the (imaginary) line joining the two end points of the (curved) leaf spring. Five process variables are believed to effect the free height of the springs. These are furnace temperature (A) and heating time (B) from the heating stage, transfer time (C) to take a spring from the furnace to the forming machine, hold-down time (D) under high pressure in the forming stage and the oil temperature (E) in the quenching stage. We will denote the factors (i.e., process variables) by letters $A, B, \ldots, E$, as noted before, according to the usual practice in design of experiments literature. Each of the five factors can be set at two levels, as shown in Table 1. The two levels for the designs are denoted by '0' and '1'. We thus need to design the experiment for determining the effect of these process variables.

II.1.2. Full factorial design

One choice of experimental design for performing the leaf spring experiment is the 5-factor *full factorial design*, denoted simply as 2^5 design. The design matrix of this design is shown in Fig. 2. A full factorial design contains all possible runs or treatment combinations, i.e., combinations of the settings of the factors, which is (as the notation suggests) $2^5 = 32$. The 2^5 design is an example of a is called a *2-level* design as all factors have only two levels. When the number of levels is more than two but the same, say, s, for all factors then the design is called a *multi-level* or *s-level* design. When all factors do not have the same number of levels, then the design is called a *mixed-level* design.

It should be noted that Fig. 2 gives the treatment combinations of the 2^5 design in a canonical order. When performing the actual experiment, the replications of the runs may be considered and the runs will usually be randomized. For example, if two replications of the 2^5 design are considered, then there are $\frac{64!}{(2!)^{32}}$ possible orderings of the runs of the experiment. One of these orderings will be randomly chosen as the *experimental plan* for the experiment.

The data collected from running the full factorial experiment can be used to estimate the individual effect of the process variables, called the *main effect*, and the *interaction effect* of combinations of process variables. Let z_i denote the measured free height of the leaf spring in run i of the experiment, and let $\bar{z}_{A=0}$ denote the average of the free height of the leaf springs for those runs in which factor A was set at level 0. The main effect of factor A, denoted by $ME(A)$, is then defined as the difference in the average values of the free heights at the two settings, i.e.,

$$ME(A) = \bar{z}_{A=1} - \bar{z}_{A=0}. \tag{2.1}$$

	A	B	C	D	E
1	0	0	0	0	0
2	0	0	0	0	1
3	0	0	0	1	0
4	0	0	0	1	1
5	0	0	1	0	0
6	0	0	1	0	1
7	0	0	1	1	0
8	0	0	1	1	1
9	0	1	0	0	0
10	0	1	0	0	1
11	0	1	0	1	0
12	0	1	0	1	1
13	0	1	1	0	0
14	0	1	1	0	1
15	0	1	1	1	0
16	0	1	1	1	1
17	1	0	0	0	0
18	1	0	0	0	1
19	1	0	0	1	0
20	1	0	0	1	1
21	1	0	1	0	0
22	1	0	1	0	1
23	1	0	1	1	0
24	1	0	1	1	1
25	1	1	0	0	0
26	1	1	0	0	1
27	1	1	0	1	0
28	1	1	0	1	1
29	1	1	1	0	0
30	1	1	1	0	1
31	1	1	1	1	0
32	1	1	1	1	1

Fig. 2. Full factorial design with 5 factors, each with 2 levels. Letters $A, \ldots, E$ denote the five factors and '0', '1' denote the two levels of each factor.

The interaction effect between two or more factors captures the effect of the factor combination that cannot be explained by simply adding the effects of the individual factors. For example, in the leaf spring experiment, the analysis later revealed that setting both furnace temperature (A) and oil temperature (E) at the same levels (either both at 0 or at 1) leads to larger free height than when the two factors are set

at different levels. The interaction effect AE is given by

$$\begin{aligned} INT(AE) &= \frac{1}{2}\{ME_{A=1}(E) - ME_{A=0}(E)\} \\ &= \frac{1}{2}\{ME_{E=1}(A) - ME_{E=0}(A)\}, \end{aligned} \tag{2.2}$$

where $ME_{A=1}(E)$ is $ME(E)$ computed by only considering runs for which A is set at level 1. The interaction effects can be computed for every factor combination, from 2-factor combinations to the (only) 5-factor combination ($ABCDE$). We skip the mathematical expressions for computing higher-order effects (like ABC, etc.). They can be found in Wu and Hamada (2000, Section 3.4.2). All the main effects and interaction effects among the factors of an experiment can be estimated in a full factorial design.

II.1.3. Regular fractional factorial designs

It is not always feasible to perform such a large number of runs. In the leaf spring experiment, running 32 runs (or 64, if two replications are made) may not be practical as this leads to disruptions in daily operations and subsequent losses due to decreased productivity and high experiment costs. Running a subset of the runs in Fig. 2 is then more practical. This design, which considers only a subset of the runs in the 2^5 design is called a *fractional factorial design*.

Suppose the fractional factorial design given in Fig. 3 is used for performing the experiment. This design uses the subset {1,3,5,7,10,12,14,16,18,20,22,24,25,27,29,31}, of 16 runs, of the runs given in the full factorial design in Fig. 2. The last column, E, in Fig. 3 can be obtained by taking the modulo-2 sum of columns A and B. Therefore, we have $E = A + B$, which is simply written as $E = AB$ and equivalently $I = ABE$. The modulo-2 term is omitted in the summation as all of $A, \ldots, E$ are

	A	B	C	D	E
1	0	0	0	0	0
2	0	0	0	1	0
3	0	0	1	0	0
4	0	0	1	1	0
5	0	1	0	0	1
6	0	1	0	1	1
7	0	1	1	0	1
8	0	1	1	1	1
9	1	0	0	0	1
10	1	0	0	1	1
11	1	0	1	0	1
12	1	0	1	1	1
13	1	1	0	0	0
14	1	1	0	1	0
15	1	1	1	0	0
16	1	1	1	1	0

Fig. 3. A 2^{5-1} regular fractional factorial design. Letters $A, \ldots, E$ denote the five factors. The defining relation is $I = ABE$.

in $GF(2)$ (since each factor takes two levels), where $GF(2)$ is the Galois field of two elements. Here, I is the identity element, and the relation $I = ABE$ is called a *defining relation*. ABE is called a *defining word* of the 2^{5-1} design. The '-1' in the exponent implies that the design has half the number of runs in the full factorial design, i.e., $2^{5-1} = 2^4 = 16$ runs. A fractional factorial design that can be constructed using such defining relations is called a *regular fractional factorial* design. All other fractional factorial designs are called *non-regular* designs.

If, instead of the runs shown in Fig. 3, the complementary subset of 16 runs would have been chosen, from the 2^5 design, for constructing the fractional factorial design, then the defining relation would have been $I = -ABE$ (i.e., $E = -A + B$). The former fraction (shown in Fig. 3) is called the *principal fraction* and the latter is called the *alternate fraction*. In this dissertation we ignore alternate fractions as the corresponding designs can be trivially obtained from the principal fractions. Moreover, alternate fractions and principal have the same statistical properties, and

they are isomorphic under relabelings of the level labels of factors (e.g., in the 2^{5-1} designs under consideration, changing the level labels of E in one fraction gives the other fraction).

The trade-off of using a smaller run-size design is that all the main effects and interaction effects cannot be estimated. For example, for the 2^{5-1} design in Fig. 3, we have $A = BE$, $B = AE$ and $E = AB$ from the defining relation $I = ABE$. Therefore, we can only estimate the sum $ME(A) + INT(BE)$ using either equation 2.1 or 2.2. The main effect of A cannot be estimated independent of the interaction effect BE. The main effect A is then said to be *aliased* with the interaction effect BE. Similarly, the effect of B is aliased with AE and that of E with AB. Usually though, it is assumed that lower order effects are more likely to be significant and the higher order effect is then neglected. Therefore, we may estimate $ME(A)$ using equation 2.1 by assuming that $INT(BE)$ is negligible. But, in practice, it is essential to ensure that such an assumption is reasonable.

It should be noted that any combinations of factors, other than ABE, could have also been chosen in the defining relation of the 2^{5-1} design. This would have then given a different 2^{5-1} design – a design with different pairs of aliased effects and hence capable of estimating a different set of effects. There are (theoretically) $2^5 - 1$ possible choices for the defining word, obtained by considering all possible combinations of the 5 letters of length one or more. But some of these are not practical. For example, a defining relation $I = AB$ would mean that the main effects of A and B are aliased and cannot be independently estimated. Therefore, usually words of length at least three are used as defining words. The number of possible choices for the defining word is then $2^5 - \binom{5}{2} - 5 - 1 = 16$.

A smaller, 8-run, regular fractional factorial design can also be considered for the leaf spring experiment by using two defining relations instead of one in the design.

	A	*B*	*C*	*D*	*E*
1	0	0	0	0	0
2	0	0	1	1	0
3	0	1	0	1	1
4	0	1	1	0	1
5	1	0	0	0	1
6	1	0	1	1	1
7	1	1	0	1	0
8	1	1	1	0	0

Fig. 4. A 2^{5-2} regular fractional factorial design. The defining contrast subgroup is $\{I, ABE, BCD, ACDE\}$.

Suppose the 2^{5-2} design is constructed by using defining words $\{BCD, ABE\}$. This design, shown in Fig. 4, has the runs {1,4,6,7,9,12,14,15} of Fig. 3. Taking the modulo-2 sum of the two defining words gives the word $ACDE$. Together, these words form an abelian group, $\{I, ABE, BCD, ACDE\}$, called the *defining contrast subgroup* of the design.

In general, an s-level regular fractional factorial design is denoted by s^{n-k}, and has n factors, each with s levels, and consists of s^a $(a = n - k)$ runs. Thus, it is the $\frac{1}{s^k}$th fraction of a s^n full factorial design, where the fraction is determined by k defining words. The defining words are a set of *generators* for the defining contrast subgroup (an abelian group). The group, therefore, consists of s^k words, including the identity element I. Hence, the fractional factorial designs generated in this manner are also called group-generated fractions. For an introduction to group theory see (Robinson, 1995, Rotman, 1995).

II.1.4. Representations of regular fractional factorial designs

A regular fractional factorial design is uniquely defined by the number of factors, n, and its defining contrast subgroup, S (or equivalently, a set of defining words). We will 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., there may exist two distinct sets of defining words that generate the same defining contrast subgroup. For example, another set of defining words for the design in Fig. 4 is $\{ABE, ACDE\}$.

An alternative representation of a regular fractional factorial 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^{5-2} design in Fig. 4, the treatment combination subgroup consists of 8 elements $\{I, CD, BDE, BCE, A, ACD, ABDE, ABCE\}$, each corresponding to the runs in the principal fraction. 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)). For a 2^{n-k} design, the size of S, the defining contrast subgroup, is 2^k and that of T, the treatment combination subgroup is 2^{n-k}. Therefore, we may choose the smaller of the two representations depending on whether $|T| = 2^{n-k} < 2^k = |S|$ or $n < 2k$.

II.1.5. Classification and ranking of designs

Suppose we decide to use a 2^{5-1} design for performing the leaf spring experiment. But, as we noted earlier, there are more than one 2^{5-1} designs. A natural question to ask then is which design is better? Since the different choices of 2^{5-1} designs lead to different sets of estimable effects, the designs are usually compared by their estimation capability. Of course, if we knew exactly which main and interaction effects are significant (and which not) then we could have precisely picked the one design that could estimate all the significant effects. But usually such information is not available, as in the case of our leaf spring experiment. In such a case we assume the *hierarchical ordering principle* for the effects (Wu and Hamada, 2000, pg. 112), i.e., lower-order effects are more important than higher order ones and effects of the same

order are equally significant. Some simple yet powerful metrics, namely *resolution* and *aberration*, have been developed for classifying and ranking regular fractional factorial designs based on this idea. These are based on what is known as the *word length pattern* of a regular fractional factorial design.

The word length pattern of a regular fractional factorial design is given by the vector $(a_3, a_4, \ldots, a_n)$, where a_i is the number of words of length i appearing in the defining contrast subgroup of the design. The length of a word is the number of letters appearing in the word. Since designs with words length smaller than 3 are not usually considered (as they would lead to aliased main effects, e.g., effects A and B are aliased if AB is in the defining contrast subgroup), there is no a_1 and a_2 in the word length pattern. The identity element I (taken to be of length identically zero) in the defining contrast subgroup is ignored when writing the word length pattern.

For example, for the design in Fig. 4, the defining contrast subgroup is $\{I, ABE, BCD, ACDE\}$. There are two words of length 3 (namely ABE and BCD) and one word of length 4 (namely $ACDE$) appearing in the defining contrast subgroup. Therefore, the word length pattern of this design is $(3, 1, 0)$.

Given the word length pattern, $(a_3, a_4, \ldots, a_n)$, of a regular fractional factorial design, the resolution of the design is defined as the smallest R such that $a_R > 0$. For the 2^{5-2} in the example above, the resolution is therefore 3, typically denoted in roman numerals as III. This design will usually be written as 2^{5-2}_{III}.

Designs with greater resolution are considered better as they have fewer lower-order effects aliased with each other. The *maximum resolution* criterion (Box and Hunter, 1961a,b) therefore recommends selecting the design that has the greatest resolution, in a collection of designs. But this criterion generally does not discriminate very well as usually more than a few designs may fall into the same resolution class. To further discriminate between regular fractional factorial designs, Fries and Hunter

(1980) proposed the *minimum aberration* criterion. This says that if two designs have the same resolution R, then the design with a smaller value of a_R in its word length pattern is better. In practice, this leads to choosing the design which has a smaller number of lower-order effects aliased with each other. The design which has the least aberration among all the designs in its class (e.g., the class of 2^{7-3} designs) is called the *minimum aberration design*.

Extensions of the word length pattern, resolution and aberration to non-regular designs also exist (Tang and Deng, 1999). But we will skip the details as their knowledge will not be required in this dissertation.

II.2. The design isomorphism problem

In the context of fractional factorial designs, two types of isomorphisms or equivalences have been identified in literature – *combinatorial isomorphism* and *geometric isomorphism*. The distinction between the two is based on whether the factors being considered are qualitative or quantitative. Quantitative factors are those whose levels can be ordered on a numerical scale, whereas qualitative factors are those whose levels cannot be put in any specific ordering. Combinatorial isomorphism is relevant for qualitative factors, whereas geometric isomorphism is relevant in the case of quantitative factors.

Two fractional factorial design matrices with qualitative factors are called combinatorially isomorphic to each other if one can be obtained from the other by some relabeling of the factor labels, level labels of factors and row labels. Two fractional factorial design matrices with quantitative factors are called geometrically isomorphic to each other if one can be obtained from the other by some relabeling of the factor labels, reversing the order of the levels of factors and relabeling of row labels. For

2-level designs, these two definitions coincide. For this reason, in the remainder of this dissertation we will refer to combinatorial isomorphism simply as isomorphism. We thus have the following definition for design isomorphism.

Definition II.1. Two fractional factorial design matrices are called *isomorphic* to each other if one can be obtained from the other by some relabeling of the factor labels, level labels of factors and row labels.

Fig. 1 gives an example of two 2^{7-3} design matrices that are isomorphic to each other. Since a design matrix is uniquely defined by its defining contrast subgroup, we have Proposition II.2, which essentially paraphrases Theorem 5 of Chen (1992). Hence, we omit its proof.

Proposition II.2. *Two 2-level regular fractional factorial designs, $d_1 \equiv \{n, S_1\}$ and $d_2 \equiv \{n, S_2\}$, where S_1, S_2 are defining contrast subgroups, 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.*

Corollary II.3. *Two 2-level regular fractional factorial designs, $d_1 \equiv \{n, T_1\}$ and $d_2 \equiv \{n, T_2\}$, where T_1, T_2 are treatment combination subgroups, are isomorphic to each other if and only if one of T_1 or T_2 can be obtained from the other by some permutation of factor labels and reordering of words.*

Proof. This follows from Prop. II.2 due to the one-to-one relationship between defining contrast subgroup and treatment combination group. □

It should be noted that, when two designs are isomorphic, the *isomorphism* is the permutation (or relabeling map) from the factor labels of one design to the other, under the action of which the two designs are identical.

The defining contrast subgroup of the 2^{7-3} design presented in Fig. 1(a) is $S_1 = \{I, ABE, ACF, BDG, ADEG, BCEF, CDEFG, ABCDFG\}$, and the defining contrast subgroup of the design in Fig. 1(b) is $S_2 = \{I, ABE, ACF, CDG, ADFG, BCEF, BDEFG, ABCDEG\}$. S_1 is isomorphic to S_2 under the factor label permutation $B \leftrightarrow C$, $E \leftrightarrow F$. This relabeling map is the isomorphism between the two designs.

Definition II.1 and Proposition II.2 (or Corollary II.3) provide ways for testing if two designs are isomorphic by comparing their design matrices or defining contrast subgroups, respectively, for isomorphism.

II.3. Isomorphism checks in literature

An isomorphism check gives a condition that can be used to test if two designs are isomorphic or not. 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 2 summarizes the major isomorphism checks proposed 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 and comparisons, see Katsaounis and Dean (2008).

Various necessary conditions for checking the equivalence of two designs have been studied in literature. Draper and Mitchell (1967) first proposed the problem of isomorphism of 2-level regular fractional factorial designs. They suggested comparing the word length patterns of two designs to decide if they are isomorphic or not, but

[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.

Table 2. Isomorphism checks proposed in literature

	Isomorphism check	Type of check	Type of design	Relevant papers
1	word length pattern	necessary	regular, multi-level	Draper and Mitchell (1967)
2	letter pattern matrix	necessary	regular, multi-level	Draper and Mitchell (1970)
3	exhaustive relabeling check	necessary and sufficient	regular, multi-level	Chen *et al.* (1993)
4	Hamming distance based	necessary and sufficient	general, multi-level	Clark and Dean (2001)
5	centered L_2 discrepancy	necessary	general, multi-level	Ma *et al.* (2001)
6	extended word length pattern	necessary	general, 2-level	Sun *et al.* (2002)
7	minimal column base	necessary and sufficient	general, 2-level	Sun *et al.* (2002)
8	indicator function representation based	necessary and sufficient	general, 2-level	Cheng and Ye (2004)
9	moment projection pattern	necessary	regular, multi-level	Xu (2005)
10	coset pattern matrix	necessary	regular, 2-level	Zhu and Zeng (2005)
11	eigenvalues of word pattern matrices	necessary, conjectured sufficient	regular, 2-level	Lin and Sitter (2008)

noted that this is not a sufficient check. The same authors devised the letter pattern matrix in Draper and Mitchell (1970), which they conjectured to be a sufficient condition also. This conjecture was disproved by Chen and Lin (1991), who gave examples of two distinct 2^{31-16}_{VII} designs with identical letter pattern matrices. Recently, Zhu and Zeng (2005) gave smaller examples, two 2^{12-7} designs, disproving the same conjecture.

Chen *et al.* (1993) proposed the first necessary and sufficient check, wherein they compared two designs using an exhaustive relabeling approach. More recently, Clark and Dean (2001) proposed a check based on the Hamming distances between points in a high-dimensional space. This is a necessary and sufficient condition for checking the isomorphism between two fractional factorial designs. Ma *et al.* (2001) extended Clark and Dean (2001)'s work and proposed a necessary check by using the Hamming distance matrix to define the centered L_2-discrepancy, a uniformity measure, between two designs. The latter two checks (Clark and Dean, 2001, Ma *et al.*, 2001) are applicable to non-regular designs also. These two checks have been found to be quite efficient (Katsaounis and Dean, 2008) and we will be comparing the efficiency of our proposed isomorphism check with them. Sections II.3.1 and II.3.2 present more details on these two methods.

Sun *et al.* (2002) used the extended word length pattern, a generalization of the word length pattern to non-regular designs (Tang and Deng, 1999), as a necessary condition. They also proposed a necessary and sufficient isomorphism check for 2-level (regular and non-regular) designs based on the minimal column base of a Hadamard matrix. Xu (2005) use a coding theory approach to construct moment projection patterns as a necessary condition for classifying fractional factorial designs. Zhu

and Zeng (2005) compare coset pattern matrices[2] to check for isomorphism. They prove with examples that this is only a necessary condition. Lin and Sitter (2008) construct a word pattern matrix and propose checking isomorphism by comparing the eigenvalues of certain submatrices of this word pattern matrix. They have conjectured that this condition is sufficient also. Lin and Sitter (2008)'s eigenvalues based check appears to be the most efficient isomorphism checks among the proposed checks in literature. Section II.3.3 presents details of this method.

In the next few subsections, we discuss the details of Clark and Dean (2001), Lin and Sitter (2008), Ma *et al.* (2001)'s isomorphism checks. We will be comparing our proposed isomorphism check with these isomorphism checks as they appear to be the most efficient ones in literature.

II.3.1. Clark and Dean (2001)'s Hamming distance based method

Let T_d denote the design matrix of a design d with n factors and N runs. Clark and Dean (2001) construct a Hamming distance matrix H_d of size $N \times N$ for the design d. The $(i,j)^{\text{th}}$ element of H_d is the number of mismatches in the i^{th} and j^{th} runs (or rows) of T_d; this is the *Hamming distance* between the runs i and j of T_d. The Hamming distance matrix obtained in this way is invariant to permutations of factor labels (columns in T_d) and level labels of factors.

Clark and Dean (2001) showed that two designs d_1 and d_2 with design matrices T_{d_1} and T_{d_2} (of size $N \times n$) are isomorphic if and only if (iff) there exists a permutation of factor labels (of T_{d_2}) and a row permutation of the Hadamard distance matrix (H_{d_2}) such that the resulting H_{d_2} (after row permutation) is identical to H_{d_1} when any subset of the columns in T_{d_1} are used. It should be noted that since this isomorphism check

[2] A coset is an algebraic structure defined in group theory (Robinson, 1995, Rotman, 1995).

	A	B	C	D	E	F	G
1	0	0	0	0	0	0	0
2	0	0	0	1	0	0	1
3	0	0	1	0	0	1	0
4	0	0	1	1	0	1	1
5	0	1	0	0	1	0	1
6	0	1	0	1	1	0	0
7	0	1	1	0	1	1	1
8	0	1	1	1	1	1	0
9	1	0	0	0	1	1	0
10	1	0	0	1	1	1	1
11	1	0	1	0	1	0	0
12	1	0	1	1	1	0	1
13	1	1	0	0	0	1	1
14	1	1	0	1	0	1	0
15	1	1	1	0	0	0	1
16	1	1	1	1	0	0	0

Fig. 5. A 2^{7-3} fractional factorial design with 7 factors, each with 2 levels, and generators $\{ABE, ACF, BDG\}$.

uses design matrices it is applicable to all classes of fractional factorial designs with qualitative factors and quantitative factors (after slight modification in the above definition of distance matrix H_d).

The necessary and sufficient condition directly compares two designs for isomorphism. In the worst case, it requires considering $n(n!)^2$ relabelings of a design. Although this is much smaller than considering all possible relabelings but can still be quite slow. For this reason, Clark and Dean (2001) recommended using a necessary check for comparing two designs before running their necessary and sufficient check. As a necessary check they compare the Hamming distance matrices constructed using subsets of the factors and checking if the rows of the two matrices contain the same set of distances with the same multiplicity.

As the Clark and Dean (2001)'s check targets the general class of fractional factorial designs, it does not exploit the extra structure in regular fractional factorial designs. For regular fractional factorial designs, we construct a new variant of Clark

	A	B	C	D	E	F	G
ABE	1	1	0	0	1	0	0
ACF	1	0	1	0	0	1	0
BDG	0	1	0	1	0	0	1
$ADEG$	1	0	0	1	1	0	1
$BCEF$	0	1	1	0	1	1	0
$CDEFG$	0	0	1	1	1	1	1
$ABCDFG$	1	1	1	1	0	1	1

Fig. 6. Matrix representation of a defining contrast subgroup for constructing variant of Clark and Dean (2001)'s isomorphism check.

and Dean (2001)'s check (Shrivastava and Ding, 2010). Instead of computing the Hadamard distance matrix from the design matrix we use the defining contrast subgroup. We construct matrix representation of the defining contrast subgroup of a design and then use this matrix to compute the Hadamard distance matrices. For example, consider the design in Fig. 5. The defining contrast subgroup of this design is $\{I,\ ABE,\ ACF,\ BDG,\ ADEG,\ BCEF,\ CDEFG,\ ABCDFG\}$. Fig. 6 shows the matrix representation of the defining contrast subgroup. Each word in the defining contrast subgroup corresponds to a row in the matrix. The isomorphism check of Clark and Dean (2001) can now be used with this matrix as input for each design. This representation has an advantage for large size designs where the number of runs may well exceed the number of words in the defining contrast subgroup. For example, a 2^{15-5} design has (2^{10} =) 1024 runs but has only (2^5 =) 32 words in its defining contrast subgroup. This speeds up the computation involved in constructing the Hadamard distance matrices.

II.3.2. Ma *et al.* (2001)'s centered L_2 discrepancy based method

Ma *et al.* (2001) extended Clark and Dean (2001)'s Hadamard distance matrix based method by computing the centered L_2 discrepancy, CD_2^2, of the designs for testing if

two designs are isomorphic or not. The centered L_2 discrepancy gives a measure of the uniformity of a design (Fang *et al.*, 2000). For a two level design d with N runs and n factors, the centered L_2 discrepancy can be computed by

$$CD_2^2(d) = \left(\frac{13}{12}\right)^n - 2\left(\frac{35}{32}\right)^n + \frac{1}{n^2}\left(\frac{5}{4}\right)^n \left(n + 2\sum_{i=1}^{n}\sum_{j=1}^{i-1}\left(\frac{4}{5}\right)^{H_d(i,j)}\right), \qquad (2.3)$$

where $H_d(i,j)$ is the $(i,j)^{\text{th}}$ element of the Hadamard distance matrix H_d described in the previous section.

The $CD_2^2(d)$ distribution can be computed by considering a subset of the factors. Since there are $\binom{n}{k}$ subsets of factors size k, the computed $CD_2^2(d)$ values are used to form a distribution called the k-dimensional $CD_2^2(d)$ distribution; this is denoted by $F_k(d)$. $F_1(d)$, ..., $F_n(d)$ together form the $CD_2^2(d)$ distribution of the design d.

Ma *et al.* (2001) suggested comparing the CD_2^2 distributions of two designs to test for isomorphism. It can be shown that the two isomorphic designs will have the same CD_2^2 distribution and is therefore a necessary check. They further conjectured that their method is also a sufficient check. The following example disproves this conjecture.

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

$$g_1 = \{ABF, ACG, ADG, BEH, BCDI\}$$

$$g_2 = \{ABF, ACG, BDG, CDH, BCEI\}$$

Designs d_1 and d_2 were found to be non-isomorphic (using our graph based isomorphism check presented in Chapter III, and Clark and Dean (2001)'s isomorphism check for defining contrast subgroups). Table 3 gives the identical CD_2^2 distribution of these two designs.

Table 3. Distribution of CD_2^2 values of two non-isomorphic 2^{10-5} designs

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

II.3.3. Lin and Sitter (2008)'s eigenvalue method

Lin and Sitter (2008) have proposed a new necessary isomorphism check for 2-level regular fractional factorial designs that they have conjectured to be sufficient also. The method first constructs matrices from the defining contrast subgroups of the

design, called the word pattern matrices. The following are the word pattern matrices W_3, W_4, W_5 and W_6 of the 2^{7-3} design in Fig. 5 with defining contrast subgroup $\{I$, ABE, ACF, BDG, $ADEG$, $BCEF$, $CDEFG$, $ABCDFG\}$:

$$W_3 = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix},$$

$$W_4 = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{pmatrix},$$

$$W_5 = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \text{ and}$$

$$W_6 = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

The word pattern matrices are stacked by taking their all possible combinations (which is $2^4 - 1 = 15$, in the example above). The eigenvalues of the squares of these stacked matrices are then computed. Lin and Sitter (2008) show that if two designs are isomorphic then the computed set of eigenvalues of these two designs will be the same. In their experiments they found that this criteria was always able to distinguish between two non-isomorphic designs.

It is 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, like Clark and Dean (2001). For comparing two designs, the eigenvalue check involves running an expensive computation for each design and then comparing pairs of eigenvalues (which is computationally cheap). The other methods, on the other hand, run the expensive computation on the pair of designs to determine if they are isomorphic or not. This fundamental difference makes the eigenvalue check more attractive for use in a design catalog generation algorithm. This is because, for

removing isomorphs from a collection of m designs, the eigenvalue check requires only m expensive runs compared to $\frac{m(m-1)}{2}$ runs, in the worst case, required by the other methods.

II.4. Summary

This chapter presented details on the mathematical structure and representations of fractional factorial designs, in particular, regular fractional factorial designs. In the next two chapters we will build on these representations of 2-level regular designs to develop a new isomorphism check and an efficient design generation algorithm.

The isomorphism checks covered in this chapter present the best approaches existing in current literature for comparing regular fractional factorial designs for isomorphism. Of these, we chose to present details on three methods as we will be comparing our proposed check with them in Chapter VI. This choice is motivated by the results in Katsaounis and Dean (2008), which compares these various isomorphism checks. They found Clark and Dean (2001)'s isomorphism check to be no slower than any other necessary and sufficient check, and found Ma *et al.* (2001)'s necessary check as a good trade-off between speed and discriminatory capability.

CHAPTER III

A NEW GRAPH BASED ISOMORPHISM CHECK

Graphs are mathematical structures that have been extensively studied in mathematics and computer science. 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. For an introduction to graph theory please see Diestel (2005). Fig. 7 gives an example of a simple graph. The graph has 6 vertices $\{A, B, C, D, E, F\}$ and 7 edges $\{(A, B), (A, F), (B, C), (C, D), (C, F), (D, E), (D, F)\}$. Due to their simple structure, graphs have found wide applications in science (Balaban, 1985, Mason and Verwoerd, 2007) and engineering (Cook *et al.*, 1998, Hayes, 2000a,b).

III.1. Graph models in design of experiments

There have been many studies relating the fields of experimental designs and graph theory. But the generated knowledge can best be described to be skewed towards

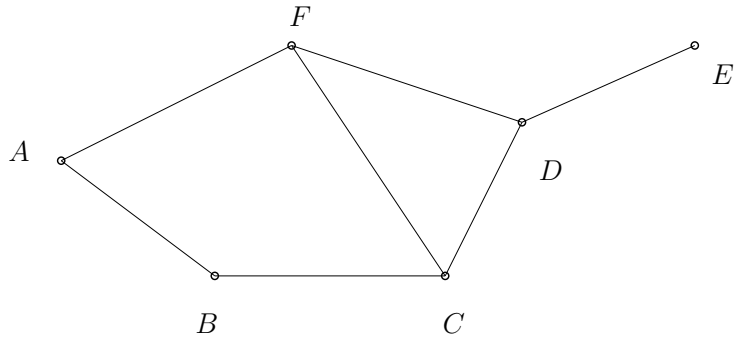


Fig. 7. A simple graph

the study of block designs, also called combinatorial designs. The relation between block designs and regular graphs has been well established (Bose, 1963), and has been used for constructing block designs (Kaski, 2002), including optimal designs (Cheng, 1981). The problem of testing isomorphism between block designs has also been shown to be computationally equivalent to the graph isomorphism problem (Colbourn and Colbourn, 1981). For further details on the use of graphs for studying block designs see Cameron and van Lint (1980).

The use of graphs for studying factorial designs has primarily been limited to representing the confounding (or aliasing) relationships. Recall that, in the design in Fig. 5, since ABE is a defining word, main effect A is confounded with interaction effect BE, effect B with AE, and effect E with AE. The first graphical representation for these confounding relations seems to have been proposed in Daniel (1962). Taguchi (Roy, 2001) provided graph representations of the orthogonal arrays and proposed comparing a requirements graph for selecting the design for an experiment. Sun and Wu (1994), Wu and Chen (1992) extended this design selection method by provided graph representations for regular fractional factorial designs. In both of these graph representations of designs, the vertices of the graph represent the factors and the edges represent the two-factor (three-factor in Sun and Wu (1994)) interaction effects that can be estimated by the design. These models are difficult to extend to include higher order interaction effects.

In this dissertation we present new graph representations of fractional factorial designs. This appears to be the first attempt to formally relate graphs with factorial designs in a way that can be used for constructing factorial designs. We will use this structural relationship to relate the problem of fractional factorial design isomorphism with the graph isomorphism problem, and efficiently solve it.

In this and the following chapter, we will propose graph representations for 2-level

regular fractional designs and use them to develop efficient algorithms for constructing non-isomorphic catalogs of these designs. For brevity, we will refer to these 2-level regular fractional factorial designs simply as fractional factorial designs or designs in the remainder of this chapter and the next chapter.

III.2. 2-level regular fractional factorial designs as graphs

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.

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

Input: design $d \equiv \{n, S\}$

Step 1. Start with an empty graph with no vertices, i.e., $V_a = \phi$ and $V_b = \phi$ (and hence, no edges, i.e., $E = \phi$).

Step 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.

Step 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.

Step 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.

The vertex sets V_a and V_b form the two partitions of the graph. From Algorithm III.1, it is clear that there exists a bipartite graph for each 2-level regular

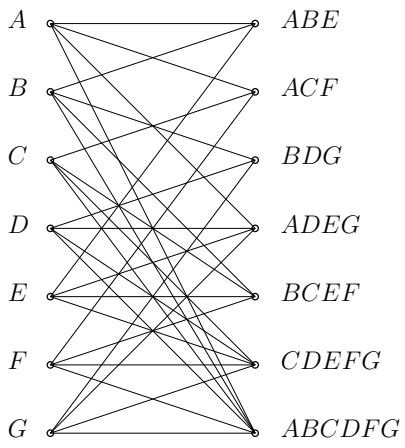


Fig. 8. Bipartite graph for the 2^{7-3} design in Fig. 1(a). Vertices on the left, set V_a, correspond to factors, and vertices on the right, set V_b, correspond to words in the defining contrast subgroup.

fractional factorial design.

Fig. 8 shows the graph representation of the 2^{7-3} design given in Fig. 1(a), with defining 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.

Due to Corollary II.3, an alternative 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 alternative graph representation for a design $\{n, T\}$ is also obtained by following Algorithm III.1, but with S replaced by T. For example, for the 2^{7-3} design in Fig. 1(a), that has the treatment combination subgroup $\{I, CF, DG, ACE, AEF, BDE, BEG, ABCD, ABCG, ABDF, ABFG, CDFG, ACDEG, ADEFG, BCDEF, BCEFG\}$, each element corresponding to the runs in the principal fraction, Fig. 9 shows the alternative graph representation.

It should be noted that for fixed n, since the size of the constructed graph

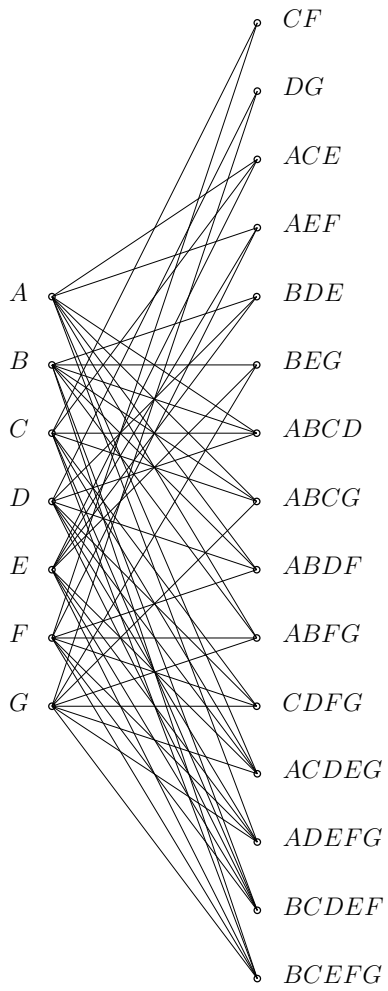


Fig. 9. Alternative bipartite graph for the 2^{7-3} design in Fig. 1(a). Vertices on the left, set V_a, correspond to factors, and vertices on the right, set V_b, correspond to words in the treatment combination group.

depends on the size of S or T, the alternative graph 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. In the 2^{7-3} design example considered above, we have $n = 7 > 6 = 2\times 3 = 2k$. So the graph constructed using the defining contrast subgroup is smaller and is selected.

III.3. Graph isomorphism and fractional factorial design isomorphism

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 (in Proposition II.2) 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 permutations that allow permutations only within each of the partitions.

Theorem III.1. *Two 2-level regular fractional factorial designs, $d_1 \equiv \{n, S_1\}$ and $d_2 \equiv \{n, S_2\}$, where n is the number of factors and S_1, S_2 are defining contrast subgroups, 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. Let $f_{i,j}$ denote the $i^{\text{th}}, i = 1, \ldots, n$, factor in design $d_j, j = 1, 2$.

First, assume that d_1 and d_2 are isomorphic.

Then $\exists$ a permutation (or relabeling) α of factor labels such that $S_1^\alpha = S_2$, i.e. S_1 is isomorphic to S_2 under the action of α.

Consider some word $w_1 = f_{1,1} \cdots f_{m,1} \in S_1$. Then $\exists$ a word $w_2 = f_{1,2} \cdots f_{m,2} \in S_2$, such that $w_1^\alpha = w_2$ ($\because S_1^\alpha = S_2$). Therefore, the edges in G_1^α are $\{f_{1,1}^\alpha, w_1^\alpha\}$, where $f_{1,1}^\alpha \in V_{a1}^\alpha$ and $w_1^\alpha \in V_{b1}^\alpha$, or equivalently $\{f_{1,2}, w_2\}$, where $f_{1,2} \in V_{a2}$ and $w_2 \in V_{b2}$ in G_2. Since w_1 and $f_{1,1}$ were arbitrary, and $V_{a1}^\alpha = V_{a2}$ and $V_{b1}^\alpha = V_{b2}$, we have G_1 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 $w_1 = f_{1,1} \cdots f_{m,1} \in S_1$. 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. Let $v_{b2} \in V_{b2}$ such that $v_{b1}^\beta = v_{b2}$. Let $v_{1,a2}, \ldots, v_{m,a2} \in V_{a2}$ be connected to v_{b2} by edges. Let $w_2 = f_{1,2} \cdots f_{m,2} \in S_2$ correspond to v_{b2}, then $v_{1,a2}, \ldots, v_{m,a2}$ correspond to $f_{1,2}, \ldots, f_{m,2}$ in some order. Without

loss of generality we assume that the correspondence is in the listed order. Further assume, 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. □

Corollary III.2. *Two 2-level regular fractional factorial designs, $d_1 \equiv \{n, T_1\}$ and $d_2 \equiv \{n, T_2\}$, where n is the number of factors and T_1, T_2 are treatment combination subgroups, 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. Follows from Theorem III.1 and Corollary II.3 □

Theorem III.1 and Corollary III.2 give a *necessary and sufficient condition* for checking if two fractional factorial designs are isomorphic by solving the graph isomorphism problem. In the next section we look at the graph isomorphism problem and our approach to solving the graph isomorphism problem.

III.4. The graph isomorphism problem

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. The relabeling map from one vertex set to the other is an isomorphism between the graphs. Fig. 10 shows an example of two

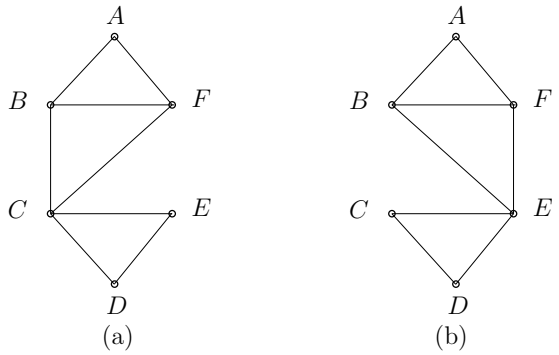


Fig. 10. Example of graph isomorphism. (a) and (b) show two simple graphs isomorphic to each other. Exchanging vertex labels B with F and C with E gives the other graph.

isomorphic graphs. The graphs in Fig. 10 are essentially mirror images of each other, and one can be obtained from the other by exchanging vertex labels B with F and C with E.

III.4.1. 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, please see Fortin (1996), Read and Corneil (1977). A problem closely associated with the graph isomorphism problem is the problem of finding the automorphisms of the graph. An *automorphism* is an isomorphism that maps a graph to itself (i.e., a vertex label permutation that does not alter the graph).

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 (i.e., an isomorphism) 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 function $C(G)$ is computed, for each graph G, that returns a canonical label for the graph. The canonical label is such that, for two graphs G and H, $C(G) = C(H)$ iff graphs G and H are isomorphic to each other.

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). In practice *nauty* has been found to be extremely efficient for most graphs and outperforms all other graph isomorphism algorithms (Kocay, 1996).

III.4.2. *nauty*

The algorithm *nauty* takes as input a graph and outputs a canonical label for the graph and the automorphisms of the graph. For computing a canonical label for a graph, the algorithm *nauty* first uses a vertex invariant to create ordered partitions of the vertex set of a graph. A partition of the vertex set V of a graph $G(V, E)$ is a sequence of disjoint subsets $V_1, V_2, \ldots, V_k$ of V. A vertex invariant is a function $i(v)$ such that if some isomorphism maps v to v' then $i(v) = i(v')$. It should be noted that the converse of this does not hold in general.

A popular vertex invariant is the *degree* of a vertex, $d(v, V)$, defined as the number of edges containing the vertex $v \in V$. An initial partition is constructed by computing the invariant for each vertex in the graph, i.e., $d(v, V)$. The partition is then refined by computing the invariant $d(v, V_i)$ for each vertex v but restricted to each subset V_i in the partition; $d(v, S)$ is defined as the number of edges between vertex v and the vertices in $S \subseteq V$. The partition is recursively refined until no further refinement of the partition is possible. The trivial total ordering scheme is used to order the newly formed subsets at each refinement step.

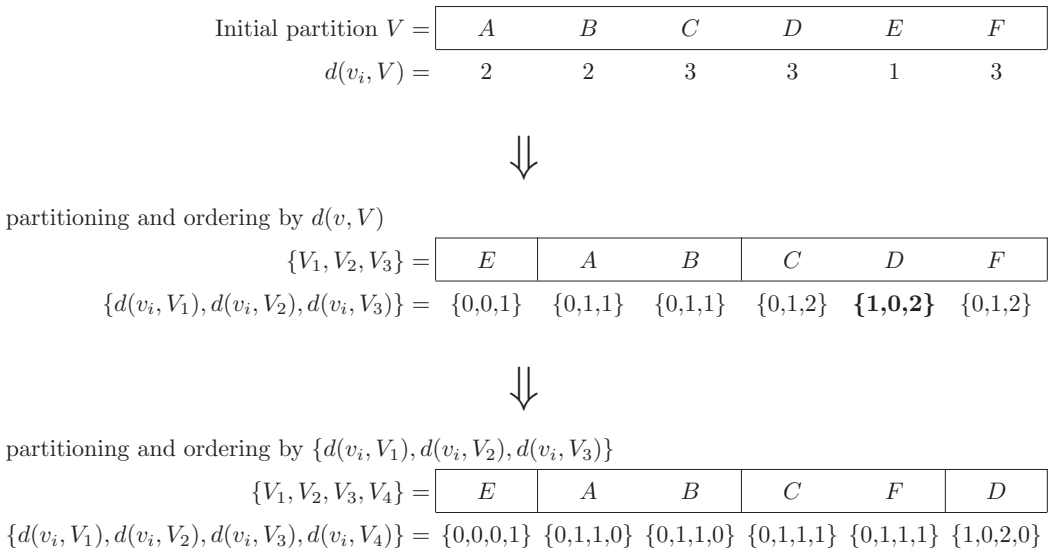
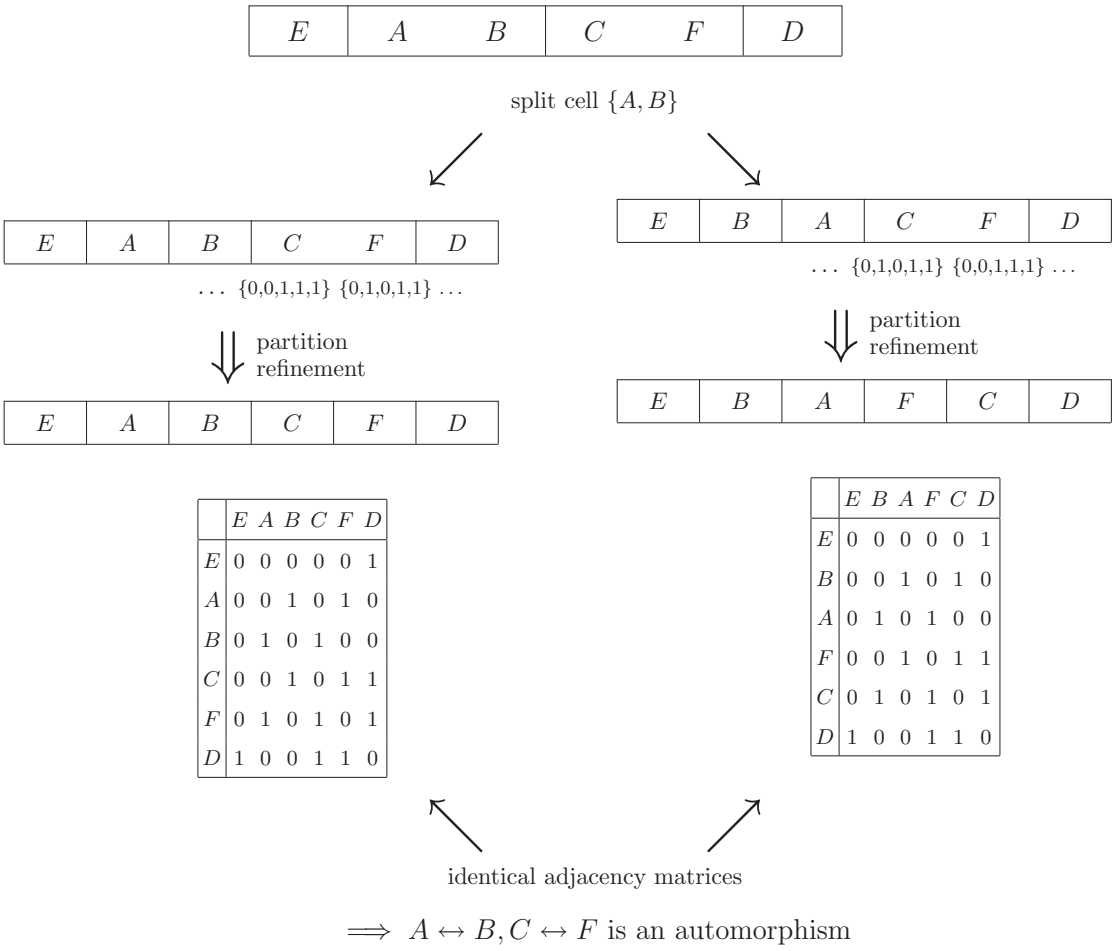


Fig. 11. Example of partition refinement in *nauty* for the graph in Fig. 7. The invariant used here is the degree of a vertex. At each step the invariant is computed and the partition is refined so that the value of the invariant for each vertex in the same cell is identical.

For example, for the graph in Fig. 7, Fig. 11 shows the steps in the partition refinement routine. In the first step, the degree of each vertex, $d(v, V)$, is computed and the vertex set is partitioned so that the vertices in the same cell have the same degree. The cells are then ordered by the degree of the vertices in the cells. This gives the partition $\{E, AB, CDF\}$. In the second step, the degree of each vertex with respect to each of the cells is computed, i.e., $\{d(v, V_1), d(v, V_2), d(v, V_3)\}$. In the example, all the members in cell V_3 do not have the same ordered set of values, so the cell is split into two. This partition refinement procedure continues until the cells can no more be split, which happens in the third iteration in this example.

Once no further refinement of a partition is possible, a search tree is constructed by splitting the non-singleton subsets in the partition. Each branch of the tree corresponds to one choice of the non-singleton subset and an ordering of the new subsets



	E	A	B	C	F	D
E	0	0	0	0	0	1
A	0	0	1	0	1	0
B	0	1	0	1	0	0
C	0	0	1	0	1	1
F	0	1	0	1	0	1
D	1	0	0	1	1	0

	E	B	A	F	C	D
E	0	0	0	0	0	1
B	0	0	1	0	1	0
A	0	1	0	1	0	0
F	0	0	1	0	1	1
C	0	1	0	1	0	1
D	1	0	0	1	1	0

Fig. 12. Search tree for finding automorphisms for the graph in Fig. 7. The final partition in Fig. 11 is split. Here only the $\{A, B\}$ split is considered.

after splitting. The new partitions are then refined using the vertex invariant and further split and refined continuously until discrete partitions are obtained. A discrete partition is a partition that has only singleton subsets. Each discrete partition is a (ordered) labeling of the graph. The automorphisms of the graph are obtained by comparing the graphs (by their adjacency matrices) with two different labelings. One of these labelings is chosen as the canonical labeling for the graph by *nauty*. The choice is made based on a complicated scheme, see McKay (1981) for details.

For the graph in Fig. 7, the search tree is constructed by starting from the partition $\{E, AB, CF, D\}$ that cannot be further refined. Fig. 12 shows the two branches of the search tree after splitting the cell $\{AB\}$. After splitting the cell, the partitions are again refined using the degree invariant $d(v, V_i)$ (as shown in the example in Fig. 11). In this example the discrete partitions are obtained in a single step of partition refinement for each of the two branches considered. The figure does not show splitting the cell $\{CF\}$ which is also considered by the algorithm. The adjacency matrices of the discrete partitions in Fig. 12 are identical, implying that the function mapping the two ordered labelings is an automorphism, i.e., the relabeling $\{A \leftrightarrow B, C \leftrightarrow F\}$ is an automorphism of this graph as it does not alter the graph.

The search tree implemented in *nauty* is a depth-first tree which includes an efficient implementation of the partition refinement routine. The canonical label chosen by *nauty* basically corresponds to the smallest automorphism under certain ordering of the automorphisms of the graph. The sorting criteria, although involved, is relatively less expensive to compute. The algorithm further gains by including certain pruning mechanisms for reducing the search tree.

The algorithm *nauty* is known to take exponential running time, in the number of vertices, in the worst case (Kocay, 1996), which 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 Alg. III.1 requires $O(n \cdot |S|)$ or $O(n \cdot |T|)$ running time to transform a design to a graph). Therefore, we expect our isomorphism check to also be very efficient for most 2-level regular fractional factorial designs.

It is essential to note that, similar to Lin and Sitter (2008), for comparing two designs, our graph based isomorphism check involves running an expensive computation for each design and then comparing the pairs of canonical labels (which is

computationally cheap). Therefore, for removing isomorphs from a collection of m designs, our graph based isomorphism check requires only m expensive runs compared to $\frac{m(m-1)}{2}$ runs, in the worst case, required by the other methods. Thus our graph based check is also an attractive choice for use in a design catalog generation algorithm. But we expect our graph based isomorphism check to outperform Lin and Sitter (2008). This is because Lin and Sitter (2008)'s eigenvalue based check, given the defining contrast subgroup, requires computing the eigenvalues of an exponentially large number of matrices; if there are p word pattern matrices of a design then eigenvalues of $2^p - 1$ matrices will be computed. The use of vertex invariants will usually leads to much smaller than exponentially large leaf nodes (discrete partitions) in the search tree. Further computing vertex invariants is much faster the eigenvalue computations, which, in general, require $O(m^3)$ floating point operations for an $m \times m$ matrix (Calvetti *et al.*, 2002).

III.5. Summary

In this chapter we presented a new approach for solving the design isomorphism problem for 2-level regular fractional factorial problems. The approach is based on modeling the designs as graphs and then solving the graph isomorphism problem. In Chapter VI we will see that this new method works much better than the existing algorithms. Another good property of this approach is that it is extensible to other classes of designs. We only need to find a graph representation of these designs that gives a one-to-one map from the designs to the graphs (this property is necessary for a result like Theorem III.1 to hold). In Chapter V we will see such an extension for the case of 2-level regular fractional factorial split-plot designs.

Since we have stressed much on the computational difficulty of the design iso-

morphism problem, it is essential that we comment on its computational complexity. Given the proven relationship (in Theorem III.1) between design isomorphism and graph isomorphism we will discuss in terms of the complexity of the graph isomorphism problem. The graph isomorphism problem has a 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 isomorphism problem for bipartite graphs has also been found to be computationally equivalent to the graph isomorphism problem (Zemlyachenko *et al.*, 1985). The graph construction for a design, as given in Algorithm III.1, has time complexity $O(n \cdot |S|)$ (all steps other than *Step 4* of the construction procedure 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 more harder than the graph isomorphism problem.

CHAPTER IV

GENERATING NON-ISOMORPHIC CATALOGS OF 2-LEVEL REGULAR FRACTIONAL FACTORIAL DESIGNS

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 (necessary and sufficient) isomorphism check. The total number of 2^{n-k} designs is given by $\binom{(2^{n-k}-1)-(n-k)}{k}$. So, for example, the total number of 32-run designs with 5, 6, 7, ... factors is 1, 26, 325, 2,600, 14,950, 65,780, 230,230, 657,800, 1,562,275, 3,124,550, 5,311,735, ... So, even if we have a very efficient isomorphism check, the problem of generating design catalogs is computationally hard.

In this chapter we describe algorithms that try to find all non-isomorphic designs in a complete catalog without comparing all the designs for isomorphism. The existing methods in literature for achieving this are described in Section IV.2. But before that, in Section IV.1 we describe the sequential approach for generating complete design catalogs on which the methods in Section IV.2 improve upon. Together, the refined sequential algorithm for generating non-isomorphic catalogs is presented in Section IV.3. Exploiting the graph models developed for fractional factorial designs in Section III.2, we present a new method for further reducing the computational effort in the generation algorithm in Section IV.4 and the final algorithm in Section IV.5.

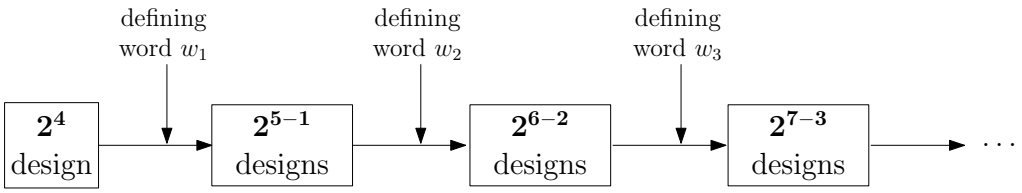


Fig. 13. Sequential generation of 16-run designs. Starting with the full factorial design, a larger (child) design is generated by adding a defining word to the smaller (parent) design.

IV.1. Sequential generation of design catalogs

Fractional factorial designs are typically generated in a sequential manner. Fig. 13 shows the sequential generation of 16-run designs. First the (only) 2^4 full factorial design is picked. A defining word w_1 is added to this design to construct a 2^{5-1} design. Then another defining word w_2 is added to construct a 6-factor design and the process continues until the design with the desired number of factors has been constructed. To construct the collection of all the 2^{n-k} designs, all possible choices of the defining words are considered at each step of the sequential procedure. All possible choices of the defining words are obtained by considering all combinations of two or more factors in the full factorial design, in the first step of the sequential procedure. Therefore, for the 16-run designs, all the defining words are obtained by adding the new factor (e.g., G for 7-factor designs) to the words in the set $\{AB, AC, AD, BC, BD, CD, ABC, ABD, ACD, BCD, ABCD\}$. We will call such a set the set of *candidate defining words*. Words of length less than two are ignored from this set because the resulting designs will then be of resolution less than three.

There are two ways in which the computational burden can be reduced from the above method. One way is to consider only a subset of the entire subset of fractional

factorial designs at each step of the sequential generation procedure (Chen *et al.*, 1993, Lin and Sitter, 2008, Xu, 2005). We will call this (sub)set of designs as the set of *intermediate designs*. The set of non-isomorphic designs can then be obtained from the set of intermediate designs by using an isomorphism check on this set. We will discuss more details on this in Sections IV.2 and IV.4.

The other method to speed up the computations is by using a necessary isomorphism check. Chen *et al.* (1993), Lin and Sitter (2008), Xu (2005) decompose the intermediate set of designs into smaller subsets, using a necessary condition (e.g., word length pattern (Chen *et al.*, 1993) or eigenvalue criterion (Lin and Sitter, 2008) among others), such that two designs belonging to different subsets are non-isomorphic to each other. Therefore, only the designs within the smaller subsets need to be compared using a necessary and sufficient isomorphism check. Chen *et al.* (1993) and Xu (2005) used an exhaustive relabeling check to do this. Lin and Sitter (2008) used Clark and Dean (2001)'s condition as the necessary and sufficient condition in one version of their generation algorithm, and their eigenvalue check in another version. Both the versions generated the same list of designs but the second version of the algorithm was much faster.

IV.2. Reducing intermediate designs

To reduce the size of the intermediate set of designs, two approaches have been proposed in literature – one by Chen *et al.* (1993) and another by Bingham and Sitter (1999a).

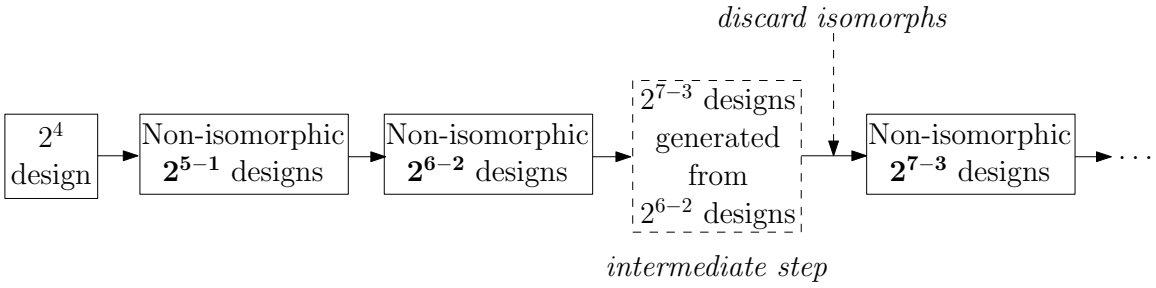


Fig. 14. Chen *et al.* (1993)'s sequential generation procedure. Intermediate designs are constructed only from the non-isomorphic designs in the preceding stage.

IV.2.1. Chen *et al.* (1993)'s modified sequential generation

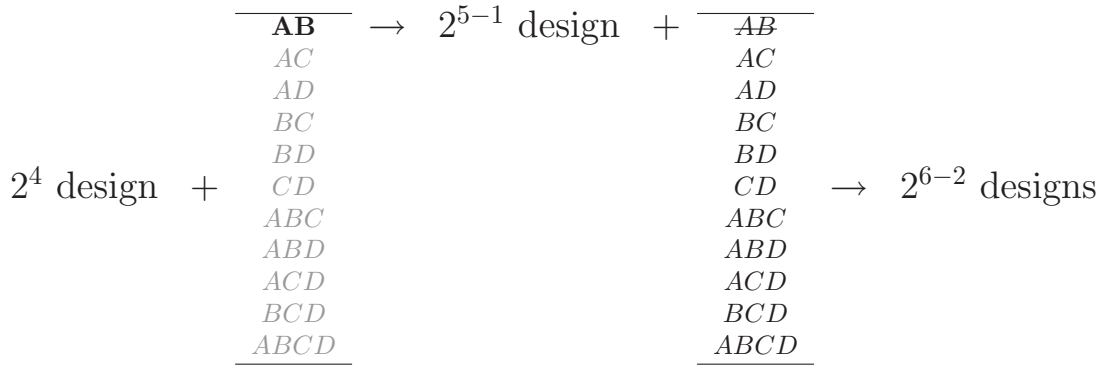
Let $D^0_{n,k}$ denote the set of all intermediate 2^{n-k} designs constructed using all the $2^{(n-1)-(k-1)}$ designs, and let $D_{n,k}$ denote the set of non-isomorphic 2^{n-k} designs (obtained from $D^0_{n,k}$ using some isomorphism check). Now, consider the set $D^+_{n,k}$ of designs constructed in a way similar to $D^0_{n,k}$ designs but by only using the $D_{n-1,k-1}$ designs, i.e., the set of all non-isomorphic $2^{(n-1)-(k-1)}$ designs. Clearly, $D^+_{n,k} \subset D^0_{n,k}$. Chen *et al.* (1993) showed that $D^+_{n,k} \supset D_{n,k}$. Therefore, we can reduce the number of intermediate designs in the sequential generation procedure by constructing intermediate designs by only using non-isomorphic (parent) designs at each stage. Fig. 14 shows the updated sequential generation procedure.

IV.2.2. Bingham and Sitter (1999a)'s orderly approach

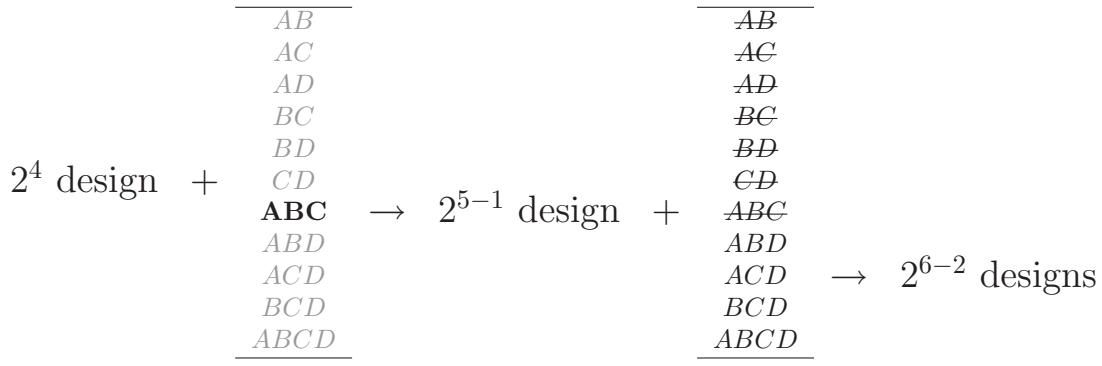
Bingham and Sitter (1999a) further reduced the size of the set of intermediate designs by constructing the designs at each stage of the sequential generation procedure in an orderly manner. They first sort the set of candidate defining words. Let $\mathcal{C}$ denote this ordered set, ordered first by word lengths and then by lexicographic ordering to break the ties. Then, intermediate designs are constructed by adding a candidate defining

word to each design d in $D_{n-1,k-1}$, the set of all non-isomorphic $2^{(n-1)-(k-1)}$ designs. Those candidate designs for which the last added defining word (when constructing the $2^{(n-1)-(k-1)}$ design from a $2^{(n-2)-(k-2)}$ design) lies before the newly added defining word, in $\mathcal{C}$, are not allowed. These candidate designs make up the set $D^{+}_{n,k}$.

Suppose we want to construct the catalog of 2^{6-2} fractional factorial designs. According to the sequential generation procedure described in Section IV.1, we first start with the 2^4 design and construct 2^{5-1} designs. The ordered set of candidate defining words, $\mathcal{C}$, here is $\{AB, AC, AD, BC, BD, CD, ABC, ABD, ACD, BCD, ABCD\}$. We construct the 5-factor designs by considering each of the candidate defining words in the order listed in $\mathcal{C}$. To construct the 6-factor designs we again add each of the words in $\mathcal{C}$ to each of the 5-factor designs. Fig. 15 shows an example constructing 6-factor designs by selecting a different 5-factor design in the first step. In Fig. 15(a) the 5-factor design is constructed by using the defining word ABE. We can now construct 6-factor designs by using each of the words in $\mathcal{C}$ except AB. In Fig. 15(b) the 5-factor design is constructed using the defining word $ABCE$ but, according to Bingham and Sitter (1999a)'s rule, we may use only the candidate words below ABC in $\mathcal{C}$ to construct the 6-factor designs. Suppose we instead use the word ABF to construct the 6-factor design in Fig. 15(b). Then it can be seen that this design, with defining contrast subgroup $\{ABF, ABCE, CEF\}$, is isomorphic (under the relabeling $E \leftrightarrow F$) to the design with defining contrast subgroup $\{ABE, ABCF, CEF\}$ that will be chosen in Fig. 15(a) (according to Bingham and Sitter (1999a)'s rule). Bingham and Sitter (1999a)'s rule thus avoids such obviously isomorphic duplicates in the intermediate set by allowing only one of the two designs to be picked.



(a) 6-factor designs constructed from the 5-factor design with defining word ABE



(b) 6-factor designs constructed from the 5-factor design with defining word $ABCE$

Fig. 15. Example of Bingham and Sitter (1999a)'s orderly design reduction procedure. (a) and (b) show two (of possible 11) different 2^{5-1} designs selected by choosing candidate defining words, ABE and $ABCE$, respectively. 2^{6-2} designs are constructed from these 2^{5-1} designs by adding defining words to them. Choices of candidate defining words not permitted by Bingham and Sitter (1999a)'s rule are stricken out. So only four 2^{6-2} designs can be constructed in (b).

IV.3. Basic algorithm for generating non-isomorphic design catalogs

Combining the sequential generation procedure of Section IV.1 and the techniques for reducing intermediate designs of Section IV.2, we get a design generation algorithm

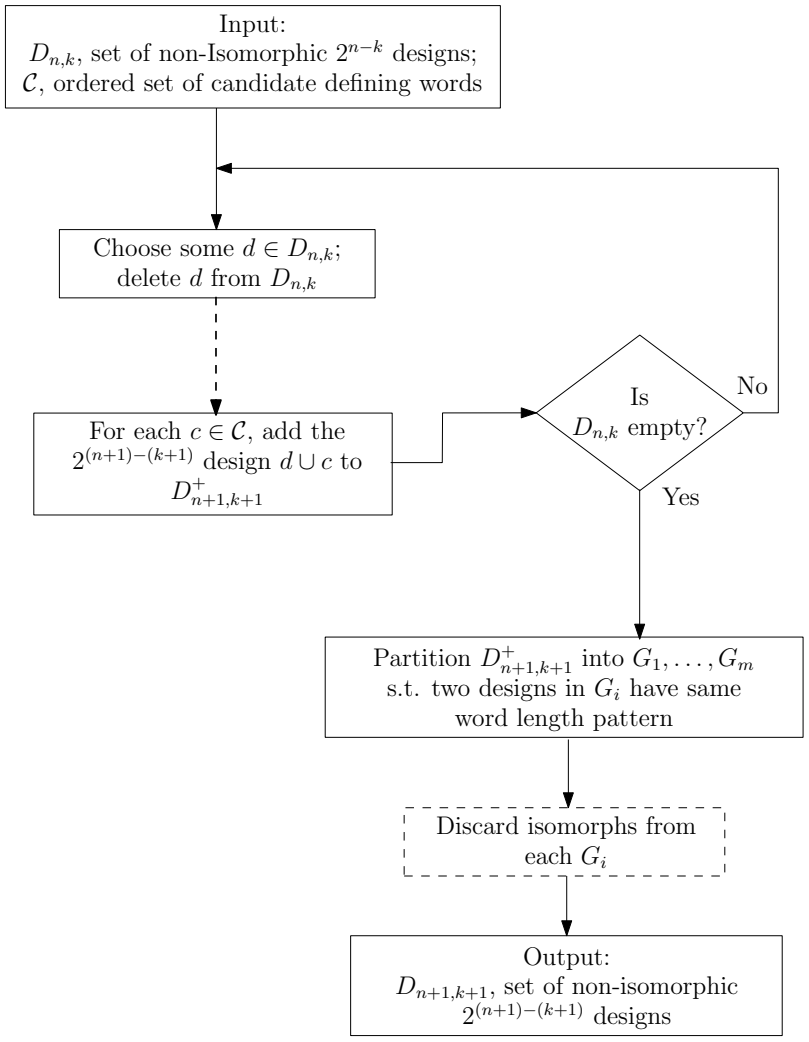


Fig. 16. Basic algorithm for generating the catalog of non-isomorphic 2^{n-k} designs (Lin and Sitter, 2008) from the set of non-isomorphic $2^{(n-1)-(k-1)}$ designs. Dashed-line box highlights the use of a different isomorphism check from Lin and Sitter (2008). Dashed line indicates the location where the new candidate defining reduction method of Section IV.4 will come in.

similar to Lin and Sitter (2008). Fig. 16 shows a schematic representation of the generation algorithm. It also highlights the differences between Lin and Sitter (2008)'s algorithm and the final algorithm proposed in Section IV.5.

We generate the set of non-isomorphic 2^{n-k} designs in a recursive manner as

described in Section IV.1. 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 (see Section IV.2.1). The basic algorithm proceeds by first constructing the ordered set of all candidate defining words $\mathcal{C}$. Then, candidate designs are constructed by adding a candidate defining word to each design d in $D_{n-1,k-1}$, the set of all non-isomorphic $2^{(n-1)-(k-1)}$ designs, according to the procedure in Section IV.2.2. These candidate designs make up the intermediate set $D^+_{n,k}$. 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 $D_{n,k}$, the set of non-isomorphic 2^{n-k} designs.

Since this basic algorithm described above is similar to that in Lin and Sitter (2008) and Bingham and Sitter (1999a), 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 $D^+_{n,k}$ is smaller than the set of all possible 2^{n-k} designs. It seems obvious that the smaller the set $D^+_{n,k}$ is the faster the algorithm is going to work. A method for further reducing $D^+_{n,k}$ is described in the next section. It's location in the final algorithm, relative to the algorithm in Fig. 16 is highlighted by the dashed arrow.

IV.4. New candidate defining word reduction method

In this section, we extend an idea suggested by McKay (1998), which proposes an algorithm for generating non-isomorphic graphs, to reduce the candidate defining words in $\mathcal{C}$. This will further reduce the number of intermediate designs generated in $D^{+}_{n,k}$. 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 IV.1. An automorphism of a 2-level regular fractional factorial design $d \equiv \{n, S\}$ is a relabeling of factor labels of d, such that the design obtained after relabeling is identically d.

Fig. 17(a) shows the graph representation of a 2^{6-2} design with defining contrast subgroup $\{ABE,\ ACF,\ BCEF\}$. Fig. 17(b) shows the graph obtained after the relabeling $B \leftrightarrow C$ and $E \leftrightarrow F$. Clearly, the two graphs, and hence the designs, are identical. Therefore, the relabeling $B \leftrightarrow C$ and $E \leftrightarrow F$ is an automorphism of the 2^{6-2} design in Fig. 17.

Theorem IV.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. Since $d \equiv \{n, S\}$, we have $d \cup c_1 = \{n+1, \{S, c_1S\}\}$ and $d \cup c_2 = \{n+1, \{S, c_2S\}\}$, where $\{S, c_iS\}$ 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 $(c_1S)^{\alpha} = c_2S$ to prove that $(d \cup c_1)^{\alpha} = d \cup c_2$.
Let $w \in S$, then, since $(c_1S)^{\alpha} = c_2S$, we have $(c_1w)^{\alpha} = c_1^{\alpha}w^{\alpha} = c_2w^{\alpha} \in c_2S$.

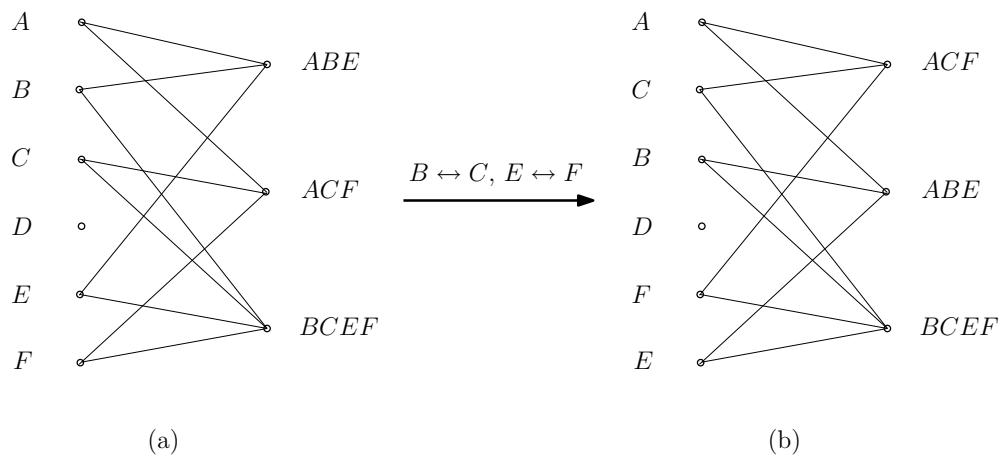


Fig. 17. Automorphism of a 2^{6-2} design. (a) is the graph representation of the 2^{6-2} design with defining contrast subgroup $\{ABE, ACF, BCEF\}$. The relabeling $B \leftrightarrow C$ and $E \leftrightarrow F$ is an automorphism of this design as the graph in (b) is identical to that in (a).

Since, the choice of $w \in S$ is arbitrary, we have $(c_1 S)^\alpha = c_2 S$. □

As an example, consider the 2^{6-2} design in Fig. 17. Among the many different possible defining words, consider the two 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. 17, BDG and CDG are clearly isomorphic to each other. The designs obtained from adding BDG (Fig. 8) and CDG are isomorphic to each other.

The result in Theorem IV.2 allows us to reduce the candidate defining words in $\mathcal{C}$ to $\mathcal{C}'$ 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 $\mathcal{C}$ to $\mathcal{C}'$ by keeping only non-isomorphic defining words. For obtaining the automorphisms of a design, we compute the auto-

morphisms of the corresponding graph representation, which we obtain using *nauty* (as described in Section III.4.2). It should be noted that the computation of the automorphism group with *nauty* does not incur any extra computational expense.

IV.5. Final algorithm for generating design catalogs

Algorithm IV.1 summarizes our design generation algorithm by combining the graph based candidate defining word reduction method of Section IV.4 with the basic algorithm described in Section IV.3.

Algorithm IV.1. *Generating non-isomorphic* $2^{(n+1)-(k+1)}$ *designs from* 2^{n-k} *designs*

Input: $D_{n,k}$, set of all non-isomorphic 2^{n-k} designs

Step 1. Construct all possible $2^a - 1$ words, except I, from the first $a = n - k$ factors, and order them by their word lengths breaking ties with lexicographic ordering. Call this ordered set $\mathcal{C}$.

Step 2. For each design $d \in D_{n,k}$

(a) Find the set $\mathcal{C}'$, of unique defining words, under the action of the automorphisms of d on $\mathcal{C}$.

(b) Construct a set of $2^{(n+1)-(k+1)}$ designs by adding to d a defining word $c \in \mathcal{C}'$, where c lies below the last added word in d in the set $\mathcal{C}$.

Step 3. Combining all the designs constructed for each d, form the set $D^+_{n+1,k+1}$, the set of intermediate designs.

Step 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.

Step 5. Use the graph based isomorphism check of Section III.3 to compare designs within each subset $G_i, i = 1, \ldots, m$, to remove isomorphs from each subset.

Step 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 *Step 5.* of Alg. IV.1, 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 IV.3. *The algorithm IV.1 generates the complete set of non-isomorphic* $2^{(n+1)-(k+1)}$ *designs.*

Proof. The result follows from the basic algorithm and Theorem IV.2. □

IV.6. Summary

This chapter presented the sequential generation procedure for efficiently generating catalogs of 2-level regular fractional factorial designs. We developed a new method for improving the efficiency of the sequential generation algorithm by reducing the number of designs actually compared using an isomorphism check. The methods developed in this and the preceding chapter appear in Shrivastava and Ding (2010), which has been accepted for publication.

Similar to the graph based isomorphism check of Chapter III, this graph based

method is also extensible to other classes of designs. In the next chapter we will see one such extension for the case of 2-level regular fractional factorial split-plot designs.

CHAPTER V

GENERATING CATALOGS OF FRACTIONAL FACTORIAL SPLIT-PLOT DESIGNS

In the last two chapters we have presented a new graph based method for generating catalogs of non-isomorphic 2-level regular fractional factorial designs. Although we described the methods in the context of a specific class of designs, namely 2-level regular fractional factorial designs, the methods are extensible to other classes of designs. As a case in point, in this chapter, we extend these methods to the class of 2-level regular fractional factorial split-plot designs. We extend both the results – the graph based isomorphism check and the candidate defining word reduction method for improving the catalog generation algorithm.

Fractional factorial split-plot designs are a practical design option for an experimenter when complete randomization of the runs of a fractional factorial experiment is not possible (see, for example, Gregory and Taam (1996), Kowalski and Potcner (2003)). They have especially been recommended for robust product design over Taguchi's inner and outer arrays as a more economical and efficient option (Bingham and Sitter, 2003, Box and Jones, 1992).

V.1. Preliminaries on fractional factorial split-plot designs

Consider the leaf spring experiment described in Section II.1.1. Running a completely randomized fractional factorial design requires that the furnace temperature (A), in the heating stage, be changed frequently, possibly after each run. This, although possible, is impractical as it increases both the duration and the cost of performing the experiment. Changing the heating time (B) after every run was also found to

	A	B	C	D	E
1	1	1	0	0	0
2	1	1	1	0	0
3	1	1	1	0	1
4	1	1	0	0	1
5	1	1	1	1	0
6	1	1	1	1	1
7	1	1	0	1	0
8	1	1	0	1	1
9	0	1	0	0	0
10	0	1	0	1	1
11	0	1	1	1	0
12	0	1	0	0	1
13	0	1	1	0	0
14	0	1	0	1	0
15	0	1	1	1	1
16	0	1	1	0	1
17	0	0	0	1	0
18	0	0	0	1	1
19	0	0	0	0	1
20	0	0	0	0	0
21	0	0	1	0	1
22	0	0	1	1	0
23	0	0	1	1	1
24	0	0	1	0	0
25	1	0	0	0	0
26	1	0	1	1	1
27	1	0	0	1	1
28	1	0	1	0	0
29	1	0	1	0	1
30	1	0	0	0	1
31	1	0	0	1	0
32	1	0	1	1	0

Fig. 18. A 32-run factorial split-plot experimental plan with 2 whole plot factors A and B, and 3 sub-plot factors C, D and E.

increase the operational costs significantly. Thus, it would be better if we put some restrictions on the randomization of the runs in the 2^5 design, when selecting the experimental plan, and not just choose from any of the 32! arrangements (assuming a single replicate) of the 2^5 full factorial design.

Under these constraints, one possible ordering in the experimental plan is the arrangement of the runs of the 2^5 design as listed in Fig. 2, i.e., the canonical ordering. This plan first fixes the furnace temperature and heating time, and then considers

all combinations (2^3) of the other factors, namely transfer time (C), hold-down time (D) and quench oil temperature (E); for example, in the first 8 runs A and B are set to their level 0s and all possible combinations ($2^3 = 8$) of the settings of C, D and E are considered. Other possible experimental plans can be obtained by randomizing the order in which factors A and B are assigned levels, i.e., in $2^2! = 4!$ ways, and/or randomizing the order of the settings of C, D and E (in 8! ways) for each fixed setting of A and B. Thus, the number of possible choices of the experimental plan are $2^2! \times (2^3!)^{2^2} = 4! \times (8!)^4$. Fig. 18 shows one such experimental plan.

Such experimental designs that have restrictions on randomization are called *split-plot designs*. The difficult to change factors, A and B in the above example, are called *whole plot factors* and the remaining (relatively) easy to change factors, C, D and E in the leaf spring example, are called *sub-plot factors*. Although the split-plot design discussed in the example above appears identical to the full factorial design, this is a different design as the restricted randomization leads to an altogether different statistical model. The split-plot design can, though, be seen as a cross product of two factorial designs, one constructed for the whole plot factors and the other for sub-plot factors.

V.1.1. Regular fractional factorial split-plot designs

In the split-plot design above, we used a full factorial design for both the whole plot factors (2^2) and the sub-plot factors (2^3). Instead, we may use a regular fractional factorial design for either the whole plot factors or the sub-plot factors. A design constructed in this way is called a *regular fractional factorial split-plot design*. Such a design is generally denoted as $2^{(n_1+n_2)-(k_1+k_2)}$, where n_1 and n_2 are the number of whole plot and sub-plot factors, respectively, and k_1 and k_2 are the levels of fractionation in the whole plot and sub-plot, respectively. For example, using the defining

	A	B	C	D	E
1	0	0	0	0	0
2	0	0	0	1	1
3	0	0	1	0	1
4	0	0	1	1	0
5	0	1	0	0	0
6	0	1	0	1	1
7	0	1	1	0	1
8	0	1	1	1	0
9	1	0	0	0	0
10	1	0	0	1	1
11	1	0	1	0	1
12	1	0	1	1	0
13	1	1	0	0	0
14	1	1	0	1	1
15	1	1	1	0	1
16	1	1	1	1	0

Fig. 19. A $2^{(2+3)-(0+1)}$ fractional factorial split-plot design. A and B are the whole plot factors, and C, D and E are sub-plot factors. The defining relation is $I = CDE$.

relation $E = CD$, we may construct a $2^{(2+3)-(0+1)}$ design as shown in Fig. 19.

Bingham and Sitter (1999a) highlight some important properties that the defining relations in regular fractional factorial designs obey. Firstly, defining words of whole plot fractions should not contain any sub-plot factors. This ensures that the whole plot factors can be randomized independently of the sub-plot factors. Secondly, defining words of sub-plot fractions may contain whole plot factors, but should have at least two sub-plot factors. If instead, the defining word contains only one sub-plot factor, then this effectively makes this sub-plot factor a whole plot factor. For example, if in the above example we choose $E = AB$ as a defining relation then the defining word ABE has only one sub-plot factor. Fig. 3 gives the design matrix for this design. As can be observed, whenever A and B are set at particular levels the setting of E is also fixed, effectively making E a whole plot factor. The design in Fig. 19 instead allows randomizing the whole plot factors first, independent of the sub-plot factors, and then randomizing the sub-plot factor settings for each whole

plot setting.

V.1.2. Properties of regular fractional factorial split-plot designs

As the difference between regular fractional factorial designs and regular fractional factorial split-plot designs is only in the randomization structure, many concepts applicable to regular fractional factorial designs have straightforward extensions. These include defining contrast subgroup, treatment combination subgroup, word length pattern, design resolution and aberration (Huang *et al.*, 1998). These objects are constructed or computed by simply considering the split-plot design as a regular fractional factorial design.

V.1.3. Representations of regular fractional factorial split-plot designs

Similar to the regular fractional factorial designs, given the number of whole plot factors n_1 and sub-plot factors n_2, regular fractional factorial split-plot designs can be uniquely represented by either the defining contrast subgroup (S) or the treatment combination subgroup (T). We will denote the defining contrast subgroup representation by the triplet $\{n_1, n_2, S\}$ and the treatment combination subgroup representation by $\{n_1, n_2, T\}$.

In the remainder of the chapter, we will only consider 2-level regular fractional factorial designs, and, for brevity, will simply refer to them as split-plot designs.

V.2. The design isomorphism problem

Analogous to regular fractional factorial designs, we have the following definition of design isomorphism for split-plot designs.

Definition V.1. Two regular fractional factorial split-plot design matrices are called

isomorphic to each other if one can be obtained from the other by some relabeling of the whole plot factor labels, sub-plot factor labels, level labels of factors and row labels.

Note that the difference between Definition V.1 and Definition II.1 (in Section II.2) is that in the case of split-plot a whole plot factor cannot be relabeled to a sub-plot factor and vice versa. That is, only factor label permutations are allowed only within the whole plot and sub-plot factor sets and not between. Fig. 20 gives an example of two $2^{(3-1)+(4-2)}$ design matrices that are isomorphic to each other under the relabeling $A \leftrightarrow B$, $d \leftrightarrow e$. If, say, factors B and C in this example were sub-plot factors, then the two designs would not have been isomorphic to each other; as the relabeling under the relabeling $A \leftrightarrow B$ would not have been permitted.

Since a split-plot design matrix is uniquely defined by its defining contrast sub-

	A	B	C	d	e	f	g
1	0	0	0	0	0	0	0
2	0	0	0	0	1	1	0
3	0	0	0	1	0	1	1
4	0	0	0	1	1	0	1
5	0	1	1	0	0	0	1
6	0	1	1	0	1	1	1
7	0	1	1	1	0	1	0
8	0	1	1	1	1	0	0
9	1	0	1	0	0	0	0
10	1	0	1	0	1	1	0
11	1	0	1	1	0	1	1
12	1	0	1	1	1	0	1
13	1	1	0	0	0	0	1
14	1	1	0	0	1	1	1
15	1	1	0	1	0	1	0
16	1	1	0	1	1	0	0

(a) Defining words: $\{ABC$, def, $Bdg\}$

	A	B	C	d	e	f	g
1	0	0	0	0	0	0	0
2	0	0	0	0	1	1	1
3	0	0	0	1	0	1	0
4	0	0	0	1	1	0	1
5	0	1	1	0	0	0	0
6	0	1	1	0	1	1	1
7	0	1	1	1	0	1	0
8	0	1	1	1	1	0	1
9	1	0	1	0	0	0	1
10	1	0	1	0	1	1	0
11	1	0	1	1	0	1	1
12	1	0	1	1	1	0	0
13	1	1	0	0	0	0	1
14	1	1	0	0	1	1	0
15	1	1	0	1	0	1	1
16	1	1	0	1	1	0	0

(b) Defining words: $\{ABC$, def, $Aeg\}$

Fig. 20. Two isomorphic $2^{(3-1)+(4-2)}$ split-plot designs. A, B and C are whole plot factors, and d, e, f and g are sub-plot factors. The two designs are isomorphic under the relabeling $A \leftrightarrow B$, $d \leftrightarrow e$.

group, similar to Proposition II.2 and Corollary II.3, we have Proposition V.2 and Corollary V.3. We skip the proofs of these as they are straightforward extensions.

Proposition V.2. *Two 2-level regular fractional split-plot designs, $d_1 \equiv \{n_1, n_2, S_1\}$ and $d_2 \equiv \{n_1, n_2, S_2\}$, where S_1, S_2 are defining contrast subgroups, 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 whole plot factor labels and sub-plot factor labels, and reordering of words.*

Corollary V.3. *Two 2-level regular fractional split-plot designs, $d_1 \equiv \{n_1, n_2, T_1\}$ and $d_2 \equiv \{n_1, n_2, T_2\}$, where T_1, T_2 are treatment combination subgroups, are isomorphic to each other if and only if one of T_1 or T_2 can be obtained from the other by some permutation of whole plot factor labels and sub-plot factor labels, and reordering of words.*

V.2.1. Isomorphism testing of two split-plot designs

Although, the problem of generating catalogs of split-plot designs has recently been of considerable interest (Bingham and Mukerjee, 2006, Bingham *et al.*, 2004, Bingham and Sitter, 1999a, 2001, 2003, Huang *et al.*, 1998, Mukerjee and Fang, 2002), there has been a lack of isomorphism checks developed specifically for split-plot designs. Bingham and Sitter (1999a) extended Chen *et al.* (1993)'s exhaustive relabeling based (necessary and sufficient) isomorphism to test two split-plot designs for isomorphism. Instead of considering all permutations of factor labels when comparing two split-plot designs, they consider only the permutations that preserve the split-plot structure (i.e., do not relabel whole plot factor to sub-plot factor). But this approach, as can be imagined, is highly inefficient, especially as the number of factors increase.

As a necessary isomorphism check, any of the necessary or necessary and sufficient

isomorphism checks for regular fractional factorial designs can be directly used. This is so because if two split-plot designs are isomorphic, then the corresponding regular fractional factorial designs, obtained from the split-plot designs by ignoring the whole plot and sub-plot differentiation, are also isomorphic. For example, in Fig. 20 if we ignore the split-plot structure then the 2^{7-3} designs obtained would still be isomorphic under the same relabeling map. The converse of this, though, is not true in general, as it may involve exchanging the labels of a whole plot factor with a sub-plot factor.

V.2.2. Extension of graph based isomorphism check

In this section we extend the necessary and sufficient graph based isomorphism of Chapter III to split-plot designs. We first provide a colored graph representation of a split-plot design and then show the equivalence between testing these graphs for isomorphism and the design isomorphism problem.

V.2.2.1. Split-plot designs as graphs

Here, we provide a vertex-colored graph representation of a 2-level regular fractional factorial split-plot design. A vertex-colored graph (henceforth colored graph) $G(V, E, c)$ is a graph in which each vertex v in the set V is associated with a color, given by $c(v)$.

Algorithm V.1. *Construction of colored graph $G(V, E, c)$ for design $d \equiv \{n_1, n_2, S\}$*

Input: design $d \equiv \{n_1, n_2, S\}$

Step 1. Start with an empty graph with no vertices, i.e., $V = \phi$ (and $E = \phi$).

Step 2. For each whole plot factor in the design d, add a vertex in V and associate it with color c_w, i.e., add vertices $v_{w1}, \ldots, v_{wn_1}$ in V and set $c(v_{wi}) = c_w$ $\forall\, i \in \{1, \ldots, n_1\}$.

Step 3. For each sub-plot factor in the design d, add a vertex in V and associate it with color c_s, i.e., add vertices $v_{s1}, \ldots, v_{sn_2}$ in V and set $c(v_{si}) = c_s \forall i \in \{1, \ldots, n_2\}$.

Step 4. For each word in the defining contrast subgroup S, except I, add a vertex in V and associate it with color c_g, i.e., add vertices $v_{g1}, \ldots, v_{g(|S|-1)}$ in V and set $c(v_{gi}) = c_g \forall i \in \{1, \ldots, |S|-1\}$, where $|S|$ denotes the cardinality of set S.

Step 5. For each word in S, except I, add edges between the vertex (in $\{v_{g1}, \ldots, v_{g(|S|-1)}\}$) corresponding to the word, and the vertices (in $\{v_{w1}, \ldots, v_{wn_1}\} \cup \{v_{s1}, \ldots, v_{wn_2}\}$) corresponding to the factors in the word.

The graph constructed by Algorithm V.1 has three colors – c_w, c_s and c_g corresponding to the whole plot factors, sub-plot factors and the words in the defining contrast subgroup.

Fig. 21 shows the colored graph representation of the $2^{(3-1)+(4-2)}$ design shown in Fig. 20(a), with defining contrast subgroup $\{I,\ ABC,\ Bdg,\ def,\ ACdg,\ Befg,$

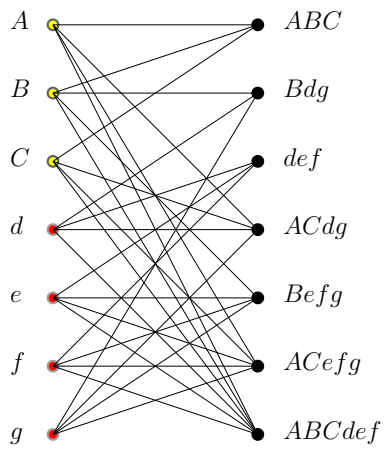


Fig. 21. Colored graph for the $2^{(3-1)+(4-2)}$ design in Fig. 20(a). Vertices corresponding to whole plot factors (A, B, C), sub-plot factors (d, e, f, g) and words in defining contrast subgroup have separate colors.

$ACefg$, $ABCdef\}$. In the figure, the vertex Bdg is connected by edges to vertices B, d, and g, and vertices B, d and Bdg have different colors.

We also have an alternative graph representation of the split-plot design using the treatment combination subgroup, T, of the design (due to Corollary V.3). The alternative graph representation for a design $\{n_1, n_2, T\}$ is obtained by following Algorithm V.1, but with S replaced by T. Similar to the fractional factorial design case, the alternative graph representation will a smaller graph whenever $n < 2k$, where $n = n_1 + n_2$ and $k = k_1 + k_2$.

V.2.2.2. Split-plot design isomorphism and colored graph isomorphism

The split-plot design isomorphism problem can be translated to the problem of checking isomorphism between the corresponding colored graph representations of the two designs. The relabelings of the factors of the split-plot design (in Proposition V.2) then correspond to the permutations of vertex labels that preserve the vertex colorings and the vertex adjacencies in the graph. This is the same as the colored graph isomorphism problem.

Two colored graphs are called isomorphic to each other if there exists a vertex-adjacency and color preserving relabeling of the vertices of one graph that makes it identical to the other. A color-preserving relabeling means a vertex, e.g., v_1, can be relabeled to another label, say, v_1', if and only if both v_1 and v_1' have the same color. Fig. 22 illustrates colored graph isomorphism with graphs with two colors. Ignoring the vertex colorings, graphs in Fig. 22(a) and (c) are isomorphic to each other under the relabeling $B \leftrightarrow F$ and $C \leftrightarrow E$. But, since vertex B (or C) does not have the same color as vertex F (or E) in Fig. 22(c), this vertex permutation is not permitted.

Theorem V.4. *Two 2-level regular fractional factorial split-plot designs,*

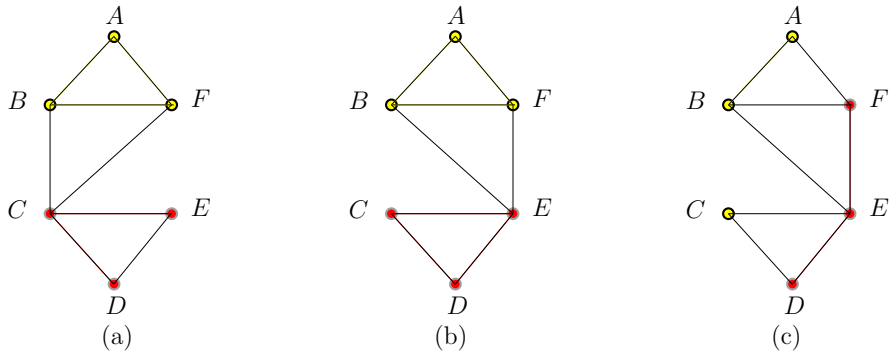


Fig. 22. Example of colored graph isomorphism. (a) and (b) are isomorphic to each other under the relabeling $B \leftrightarrow F$ and $C \leftrightarrow E$. But (a) and (c) are not isomorphic as there is no color-preserving relabeling; B and F have different colors in (c)

$d_1 \equiv \{n_1, n_2, S_1\}$ and $d_2 \equiv \{n_1, n_2, S_2\}$, where n_1, n_2 are the number of whole plot and sub-plot factors, respectively, and S_1, S_2 are defining contrast subgroups, with graph representations $G_1(V_1, E_1, c)$ and $G_2(V_2, E_2, c)$, respectively, are isomorphic to each other if and only if G_1 and G_2 are isomorphic to each other.

Proof. Let $F_{w,i}$ and $F_{s,i}$ denote the set of whole plot and sub-plot factors, respectively, in design d_i, $i = 1, 2$. Also, let $f_{i,j}$ denote some (i^{th}) factor in design d_j, $j = 1, 2$. Without loss of generality we assume that the first n_1 vertices in the graphs correspond to the whole plot factors, the next n_2 vertices correspond to the sub-plot factors and the remaining vertices correspond to the defining contrast subgroup.

First, assume that d_1 and d_2 are isomorphic.

Then $\exists$ a permutation (or relabeling) α of factor labels such that $S_1^\alpha = S_2$, i.e. S_1 is isomorphic to S_2 under the action of α, and if factor $f_{i,1}^\alpha = f_{i,2}$ (i.e., $f_{i,1} \mapsto f_{i,2}$ under α) then either $f_{i,1} \in F_{w,1}$ and $f_{i,2} \in F_{w,2}$, or $f_{i,1} \in F_{s,1}$ and $f_{i,2} \in F_{s,2}$, $i = 1, \ldots, n_1 + n_2$. The second observation simply means that the permutation α preserves the split-plot

structure of the design. Further, this means that the permutation α preserves the color of the vertices in the corresponding graphs.

Consider some word $w_1 = f_{1,1} \cdots f_{m,1} \in S_1$. Then $\exists$ a word $w_2 = f_{1,2} \cdots f_{m,2} \in S_2$ such that $w_1^\alpha = w_2$ and $f_{i,1}^\alpha = f_{i,2}, i \in \{1, \ldots, n_1 + n_2\}$ ($\because S_1^\alpha = S_2$). Therefore, the edges in G_1^α are of the form $\{f_{1,1}^\alpha, w_1^\alpha\} \equiv \{f_{1,2}, w_2\}$, which is an edge in G_2. Therefore G_1 and G_2 are isomorphic under α.

Now, assume that G_1 and G_2 are isomorphic, with $\{v_{i,1}, i = 1, \ldots, n_1\}^\alpha = \{v_{i,2}, i = 1, \ldots, n_1\}$, $\{v_{i,1}, i = n_1 + 1, \ldots, n_1 + n_2\}^\beta = \{v_{i,2}, i = n_1 + 1, \ldots, n_1 + n_2\}$ and $\{v_{i,1}, i = n_1 + n_2 + 1, \ldots, |V|\}^\gamma = \{v_{i,2}, i = n_1 + n_2 + 1, \ldots, |V|\}$.

The color preserving permutations α and β imply that the whole plot factors in d_1 are only relabeled to whole plot factors in d_2, and the same holds for sub-plot factors. Thus, we only need to now check that the S_1 is isomorphic to S_2.

Let $w_1 = f_{1,1} \cdots f_{m,1} \in S_1$. Let $v_{g1} \in V_1$ correspond to w_1, and $v_{1,1}, \ldots, v_{m,1} \in V_1$ correspond to $f_{1,1}, \ldots, f_{m,1}$, respectively. Similarly, let $w_2 = f_{1,2} \cdots f_{m,2} \in S_2$, $v_2 \in V_2$ correspond to w_2, and $v_{1,2}, \ldots, v_{m,2}$ correspond to $f_{1,2}, \ldots, f_{m,2}$, respectively. Since, $G_1^{\alpha\beta\gamma} = G_2$, we have for edges $\{v_{1,1}^{\alpha\beta}, v_1^\gamma\} = \{v_{1,2}, v_2\}$. Since the choice of $v_{1,1}$ was arbitrary in the last statement, we have, for the corresponding words, $w_1^\gamma = w_2$. 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. □

Corollary V.5. *Two 2-level regular fractional factorial split-plot designs, $d_1 \equiv \{n_1, n_2, T_1\}$ and $d_2 \equiv \{n_1, n_2, T_2\}$, where n_1, n_2 are the number of whole plot and sub-plot factors, respectively, and T_1, T_2 are treatment combination subgroups, with graph representations $G_1(V_1, E_1, c)$ and $G_2(V_2, E_2, c)$, respectively, are isomorphic to each other if and only if G_1 and G_2 are isomorphic to each other.*

Proof. Follows from Theorem V.4 and Corollary V.3 □

Theorem V.4 and Corollary V.5 give a *necessary and sufficient condition* for checking the isomorphism between two split-plot designs by solving the colored graph isomorphism problem. For solving the colored graph isomorphism problem we use canonical labeling algorithm, *nauty*, described in Section III.4.2.

V.3. Generating non-isomorphic catalogs of split-plot designs

The split-plot designs can be generated sequentially (or recursively) in a way similar to that described in Chapter IV. That is, we can construct an $(n+1)$-factor design from an n-factor design by adding a defining word to the smaller design. But the split-plot structure of the design adds some complications to this recursive method. The new defining word could correspond either to a new whole plot factor or a sub-plot factor. If the n-factor design was a $2^{(n_1-k_1)-(n_2-k_2)}$ design then the new design would then be either a $2^{(n_1+1-k_1-1)-(n_2-k_2)}$ or a $2^{(n_1-k_1)-(n_2+1-k_2-1)}$ design, depending on the new factor.

Fig. 23 shows the sequential generation of 16-run designs starting with a 2^{2+2} full factorial design. Depending on whether the defining word w_1 corresponds to a whole plot factor or a sub-plot factor a $2^{(3-1)+2}$ or a $2^{2+(3-1)}$ design may be generated. Going further one step we get a $2^{(4-2)+2}$, $2^{(3-1)+(3-1}$, $2^{(3-1)+(3-1)}$ or $2^{2+(4-2)}$ design. The two $2^{(3-1)+(3-1)}$ designs in second stage may be equivalent (if the same two candidate defining words are used in opposite order). The branch corresponding to the second $2^{(3-1)+(3-1)}$ design may be discarded as it will generate duplicate designs. To avoid such duplicate branches, we use the rule that no defining word corresponding to a whole plot factor is added to a design that contains a defining word corresponding to

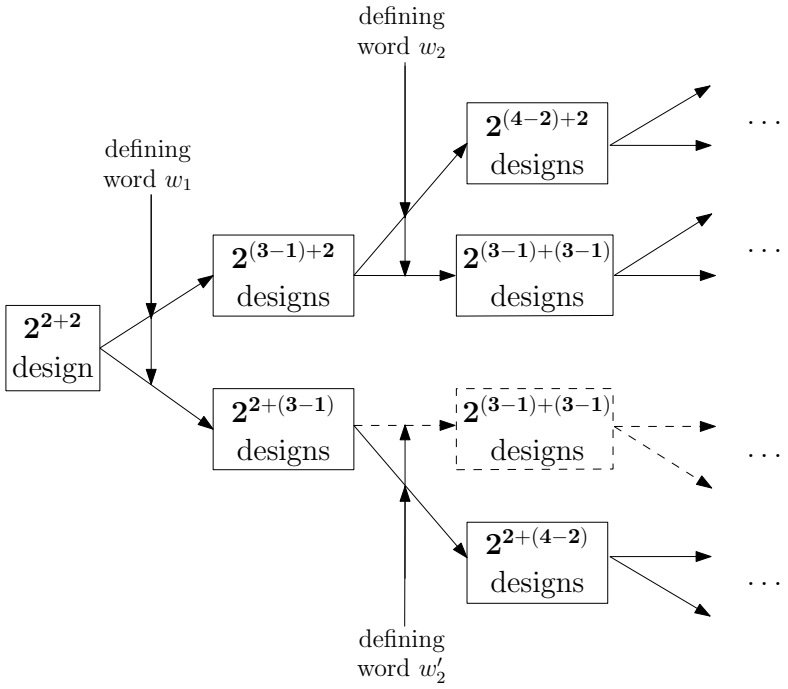


Fig. 23. Sequential generation of 16-run split-plot designs. Starting with the full factorial design, a larger (child) design is generated by adding a defining word to the smaller (parent) design.

a sub-plot factor.

Essentially, a $2^{(n_1-k_1)-(n_2-k_2)}$ is generated by starting from $2^{a_1+a_2}$ design, where $a_1 = n_1-k_1$ and $a_2 = n_2-k_2$, then recursively adding defining words corresponding to whole plot factors until $2^{(n_1-k_1)-a_2}$ designs are obtained. Then defining words corresponding to sub-plot factors are added recursively to this design until $2^{(n_1-k_1)-(n_2-k_2)}$ designs are obtained. The defining words are chosen based on the constraints noted in Section V.1.1.

Bingham and Sitter (1999a) used the sequential generation approach described above along with the intermediate design reduction methods described in Section IV.2 for generating catalogs of non-isomorphic split-plot designs. At each step of the recur-

sive procedure, they construct child designs by adding defining words (in all possible ways) to non-isomorphic parent designs only, according to Section IV.2.1. Also, some isomorphic designs are avoided by using the orderly approach of Section IV.2.2.

We use the approach of Bingham and Sitter (1999a), described above, for generating non-isomorphic design catalogs but further reduce the intermediate designs by using the candidate defining word reduction method and the graph based isomorphism check.

V.3.1. Candidate defining word reduction method

In this section, we extend the candidate defining word reduction method of Section IV.4 to split-plot designs. We first extend the concept of automorphisms split-plot designs.

Definition V.6. An automorphism of a 2-level regular fractional factorial split-plot design $d \equiv \{n_1, n_2, S\}$ is a split-plot structure preserving-relabeling of factor labels of d, such that the design obtained after relabeling is identically d.

Fig. 24(a) shows the graph representation of a $2^{(3-1)+(3-1)}$ design with defining contrast subgroup $\{ABC, def, ABCdef\}$. Fig. 24(b) shows the graph obtained after the relabeling $A \leftrightarrow B$ and $d \leftrightarrow e$. Clearly, the two graphs, and hence the designs, are identical. Therefore, the relabeling $A \leftrightarrow B$ and $d \leftrightarrow f$ is an automorphism of the $2^{(3-1)+(3-1)}$ design in Fig. 24.

Theorem V.7. *Suppose $d \equiv \{n_1, n_2, S\}$ is a parent split-plot 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 split-plot 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.*

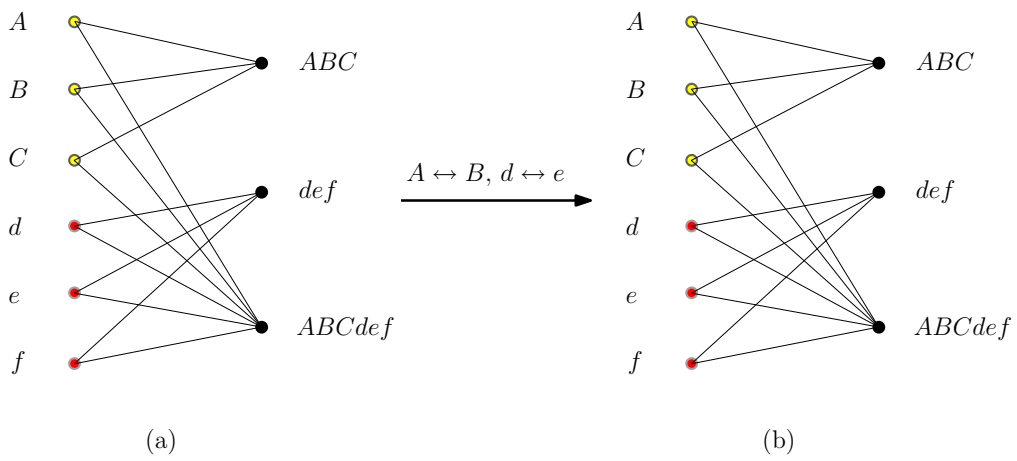


Fig. 24. Automorphism of a $2^{(3-1)+(3-1)}$ design. (a) is the graph representation of the 2^{6-2} design with defining contrast subgroup $\{ABC,\ def,\ ABCdef\}$. The relabeling $A \leftrightarrow B$ and $d \leftrightarrow e$ is an automorphism of this design as the graph in (b) is identical to that in (a).

Proof. Since $d \equiv \{n_1, n_2, S\}$, we have either $d \cup c_1 = \{n_1 + 1, n_2, \{S, c_1S\}\}$ and $d \cup c_2 = \{n_1 + 1, n_2, \{S, c_2S\}\}$, or $d \cup c_1 = \{n_1, n_2 + 1, \{S, c_1S\}\}$ and $d \cup c_2 = \{n_1, n_2 + 1, \{S, c_2S\}\}$, where $\{S, c_iS\}$ is the defining contrast subgroup of $d \cup c_i$, $i = 1, 2$.

In either case, since α is an automorphism of d, we only need to show that $(c_1S)^\alpha = c_2S$ to prove that $(d \cup c_1)^\alpha = d \cup c_2$.

Let $w \in S$, then, since $(c_1S)^\alpha = c_2S$, we have $(c_1w)^\alpha = c_1^\alpha w^\alpha = c_2w^\alpha \in c_2S$.

Since, the choice of $w \in S$ is arbitrary, we have $(c_1S)^\alpha = c_2S$. □

Theorem V.7 provides an extension of the candidate defining word reduction method to split-plot designs that works identically to the way the reduction method works for fractional factorial designs. For example, consider the two defining words Bdg and Aeg that may be added to the $2^{(3-1)+(3-1)}$ design in Fig. 24 to construct

$2^{(3-1)+(4-2)}$ designs. These words are isomorphic to each other under the relabeling $A \leftrightarrow B$ and $d \leftrightarrow e$, which is an automorphism of the parent design. The designs obtained from adding Bdg and Aeg, shown in Fig. 20, are also isomorphic to each other.

V.3.2. Final algorithm

Algorithm V.2 summarizes our split-plot design generation algorithm by combining the graph based candidate defining word reduction method of Section V.3.1 with the sequential algorithm of Bingham and Sitter (1999a). We describe here only the procedure for adding a sub-plot factor to the design. The procedure for adding a whole plot factor, i.e., constructing $2^{(n_1+1-k_1-1)+n_2}$ designs from $2^{(n_1-k_1)+n_2}$ designs, is identical to using Algorithm IV.1 for generating $2^{(n_1+1)-(k_1+1)}$ fractional factorial designs from $2^{(n_1-k_1)}$ fractional factorial designs.

Algorithm V.2. *Generating non-isomorphic* $2^{(n_1-k_1)+(n_2+1-k_2-1)}$ *designs from* $2^{(n_1-k_1)+(n_2-k_2)}$ *designs*

Input: D_{n_1,n_2,k_1,k_2}, set of all non-isomorphic $2^{(n_1-k_1)+(n_2-k_2)}$ designs

Step 1. Construct all possible 2^a-1 words, except I, from the first $a = n_1+n_2-k_1-k_2$ factors, ignoring words containing only whole plot factors. Order these by their word lengths breaking ties with lexicographic ordering. Call this ordered set $\mathcal{C}$.

Step 2. For each design $d \in D_{n_1,n_2,k_1,k_2}$

(a) Find the set $\mathcal{C}'$, of unique defining words, under the action of the automorphisms of d on $\mathcal{C}$.

(b) Construct a set of $2^{(n_1-k_1)+(n_2+1-k_2-1)}$ designs by adding to d a defining

word $c \in \mathcal{C}'$, where c lies below the last added word in d in the set $\mathcal{C}$.

Step 3. Combining all the designs constructed for each d, form the set $D^{+}_{n_1,n_2+1,k_1,k_2+1}$, the set of intermediate designs.

Step 4. Partition the set $D^{+}_{n_1,n_2+1,k_1,k_2+1}$ 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.

Step 5. Use the graph based isomorphism check of Section V.2.2 to compare designs within each subset $G_i, i = 1, \ldots, m$, to remove isomorphs from each subset.

Step 6. Collect all the remaining designs (in these subsets) in D_{n_1,n_2+1,k_1,k_2+1}, the set of non-isomorphic $2^{(n_1-k_1)+(n_2+1-k_2-1)}$ designs.

As in the case of regular fractional factorial designs, in step *Step 5.* of Alg. V.2, we construct the graphs for the designs either from their defining contrast subgroups or the treatment combination subgroups depending on whether $n_1+n_2 \geq 2\times(k_1+k_2)$ or not. Since the designs are generated recursively, starting from the full factorial design, the first few iterations (while $n_1 + n_2 \geq 2 \times (k_1 + k_2)$) use defining contrast subgroup to construct the graph. Once $n_1 + n_2 < 2 \times (k_1 + k_2)$ (so that $|S| > |T|$, the treatment combination subgroup is used to construct the graph.

Theorem V.8. *The algorithm V.2 generates the complete set of non-isomorphic* $2^{(n_1-k_1)+(n_2+1-k_2-1)}$ *designs.*

Proof. The result follows from the Bingham and Sitter (1999a)'s sequential algorithm and Theorem V.7. □

V.4. Summary

In this chapter we extended the isomorphism check and the efficient generation algorithm, developed in Chapters III and IV to 2-level regular fractional factorial split-plot designs. The extensions were based on the vertex-colored graph representations of these split-plot designs. The results extended in this chapter are included in Shrivastava (2009), soon to be submitted for publication.

It may be noted that this graph representation can be easily extended to split-split plot designs (Montgomery, 2000). Unlike split-plot designs where the factors are divided into two sets, the split-split plot designs may have more than two sets. A vertex colored graph can again be used by assigning a different color to each of the sets of factors. In Chapter VII we will discuss such extensions to other classes of designs.

CHAPTER VI

RESULTS

In this chapter we show the effectiveness of the algorithms that have been developed in this dissertation. We present the design catalogs that we generated using these algorithms, and the comparisons of the computational efficiency of our proposed algorithms with existing algorithms.

VI.1. 2-level regular fractional factorial designs

For generating catalogs of non-isomorphic 2-level regular fractional factorial designs, the best results have so far been reported by 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. Even designs are those in which the length of all words in the defining contrast subgroup is even. Using Algorithm IV.1 we were able to generate all the designs generated by Lin and Sitter (2008). Additionally, we could generate all of 1024-run (resolution ≥ 6), 2048-run (resolution ≥ 7) and 4096-run (resolution ≥ 8) designs. Table 4 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).

Appendix A lists the complete catalog of non-isomorphic fractional factorial designs of run sizes 1024, 2048 and 4096. For 1024-run designs, we only list the best two designs according to the minimum aberration criteria (in Appendix A) as the total number of these designs is very large. The other generated catalogs have been omitted as they have been generated previously in literature.

Table 4. 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	1,794	623	15	92	24	9	6
14	1	128	4,579	1,535	11	282	47	7	7
15	1	144	11,635	3,522	6	1,011	98	7	4
16	*	145	29,091	7,500	1	4,019	185	7	5
17	*	129	†	14,438	1	13,759	380	3	5
18	*	113	†	25,064	*	†	919	2	2
19	*	91	†	†	*	†	1,701	1	1
20	*	67	†	†	*	†	1,682	1	1
19	*	91	†	†	*	†	1,701	1	1
20	*	67	†	†	*	†	1,682	1	1
21	*	50	†	†	*	†	739	1	1
22	*	34	†	†	*	†	128	1	1

Table 4 Continued

n	Run Size (Resolution $\geq r$)								
	16(3)	32(3)	64(3)	128(4)	256(5)	512(5)	1024(6)	2048(7)	4096(8)
23	*	21	†	†	*	†	8	1	1
24	*	14	†	†	*	†	1	*	1
25	*	9	†	†	*	†	*	*	*
26	*	5	†	†	*	†	*	*	*
27	*	3	†	†	*	†	*	*	*
28	*	2	†	†	*	†	*	*	*
29	*	1	†	†	*	†	*	*	*
30	*	1	†	†	*	†	*	*	*
31	*	1	†	†	*	†	*	*	*
32	*	0	†	†	*	†	*	*	*

*no such designs exist.

†our implementation of no existing algorithms, including ours, returns a valid result for this problem size.

VI.1.1. Computational efficiency

In this section we compare the computational efficiency of our proposed algorithms. We first compare our isomorphism check with the three isomorphism checks described in detail in Section II.3, which we believe are the most efficient ones in literature. The three methods are Clark and Dean (2001)'s algorithm for defining contrast subgroup, DeseqCS, Section II.3.1, Ma *et al.* (2001)'s algorithm, MaCD2, (Section II.3.2) and Lin and Sitter (2008)'s eigenvalue check, EigVal, (Section II.3.3).

Our implementation of DeseqCS is based on the variant of Clark and Dean (2001)'s isomorphism check built on the defining contrast subgroup representations of regular fractional factorial designs. This method was described in Section II.3.1. We expect this variant to perform better than the original algorithm as the new method exploits the structure of regular fractional factorial designs.

In our implementation of Lin and Sitter (2008)'s eigenvalue check, we use 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, for matrix computations. We have used these libraries with the intent to have an efficient implementation of the eigenvalue check. But since the implementations may not be the most efficient, we will not be interested in small differences in performance in our comparisons. In the implementation of the eigenvalue check, on issue needs to be addressed. The issue is that the method 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.

To compare these isomorphism checks we implemented the basic algorithm described in Section IV.3 using each of the four isomorphism checks – DeseqCS, MaCD2, EigVal and our graph based isomorphism check, GBAnoR of Chapter III. Tables 5, 6 and 7 show the cumulative run times for generating 128-run, 256-run and 512-run de-

signs, respectively. The tables compare five algorithms – DeseqCS, MaCD2, EigVal, GBAnoR and GBA. GBA includes, both, our graph based isomorphism check and the candidate defining word reduction procedure presented in Section IV.4, i.e., it is the implementation of Algorithm IV.1. All the computations were done on a Windows Server 2003 R2 Standard x64 edition with an Intel Xeon 3GHz processor and 16 GB RAM. All the algorithms were programmed in C++ and built as 32-bit applications with the Microsoft Visual C++ 8.0 compiler.

The cumulative run times in Tables 5, 6 and 7 include the time needed to gen-

Table 5. Comparison of cumulative CPU time (in seconds) for generating 128 run ($R \geq 4$) designs.

$n-k$	DeseqCS	MaCD2	EigVal	GBAnoR	GBA
8-1	0.078	0.297	0.062	0.000	0.000
9-2	0.484	1.609	0.249	0.015	0.000
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	14,322.500	1,539.800‡	523.646	5.390	3.531
14-7	†	6,808.460‡	3,290.970	25.765	18.484
15-8	†	27,747.500‡	21,401.300	73.719	57.219
16-9	†	†	9.3 days§	211.362	175.752

†the problem size is too large for our implementation of the corresponding algorithm to give valid results.

‡MaCD2 did not detect all non-isomorphic designs in this case.

§our implementation of EigVal could not handle this problem size; the values reported are from Lin and Sitter (2008);

erate 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 DeseqCS, EigVal may be performing better because it makes only one expensive computation for each of the designs in the intermediate set of designs, whereas DeseqCS needs to compare pairs of designs. But MaCD2 also requires only one costly computation per design. The better performance of EigVal over MaCD2 suggests that a method, such as EigVal, developed specifically for 2-level regular designs is probably able to exploit the structure of the designs much better than methods developed to cater to the general class of designs. Nevertheless, EigVal appears to be the fastest isomorphism check before our check. Also, in our runs we did not find a single case where Lin and Sitter (2008)'s sufficiency conjecture fails.

Table 6. Comparison of cumulative CPU time (in seconds) for generating 256 run ($R \geq 5$) designs.

$n-k$	DeseqCS	MaCD2	EigVal	GBAnoR	GBA
9-1	0.156	2.250	0.046	0.015	0.000
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	110.110	6.609	0.203	0.093
13-5	179.75	260.050	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	1,025.160	492.990	42.858	1.656	1.203
17-9	1,025.340	493.178	43.061	1.843	1.296

Table 7. Comparison of cumulative CPU time (in seconds) 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	0.140
13-4	912.046	3,863.610‡	126.341	2.265	0.750
14-5	13,683.700	19,168.600‡	750.344	11.047	5.453
15-6	†	89,653.800‡	5,119.450	57.219	38.641
16-7	†	100 hours‡	30 hours§	320.910	271.534
17-8	†	†	12 days§	1,877.180	1,796.540

†the problem size is too large for our implementation of the correspond ing algorithm to give valid results.

‡MaCD2 did not detect all non-isomorphic designs in this case.

§our implementation of EigVal could not handle this problem size; the values reported are from Lin and Sitter (2008);

Compared to EigVal, for $k \geq 3$, the run times for GBAnoR are smaller by over 95% for the 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 is much less but is still impressive. For $k \geq 3$, the run times for GBA are between 30 – 80% of the run times for GBAnoR. Compared to EigVal, the total reduction in

run times is over 98% in most cases. Note that EigVal is the fastest isomorphism check before us. The generation of 2^{17-8} (512-run) designs with resolution ≥ 5 took about 12 days with EigVal (Lin and Sitter, 2008), whereas it took about 30 mins with GBA. The further reduction in run time, over GBAnoR, due to candidate reduction method was about 2.5 minutes (150 seconds).

The improvement due to our candidate defining word reduction method is better reflected in Table 8. It compares the number of designs in the intermediate set from which the non-isomorphic set is obtained by using some isomorphism check. The number of designs left in $D^{+}_{n,k}$ is about 4000 – 8000 fewer for 128-run designs with

Table 8. Number of designs in intermediate set, $D^{+}_{n,k}$, before discarding isomorphs.

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	2,063	496
4	1,273	703	480	206	4,739	1,497
5	3,346	2,026	453	267	11,077	5,731
6	7,560	4,952	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	†	†

†our implementation of no existing algorithms, including ours, returns a valid result for this problem size.

$k \geq 7$, when candidate word 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.

VI.2. 2-level regular fractional factorial split-plot designs

For 2-level regular split-plot designs, only non-isomorphic minimum aberration designs for 16-run (Bingham and Sitter, 1999a) and 32-run designs up to 13 factors (Bingham and Sitter, 2001) have been reported. The reason for reporting minimum aberration designs and not the entire catalog is perhaps because the number of non-isomorphic designs is usually much larger for split-plot designs than for regular fractional factorial designs. For this reason, we only present the set of non-isomorphic minimum aberration designs in this dissertation.

Using the algorithms developed in Chapter V, we could generate all the designs reported in literature earlier. Additionally we could generate 64-run (resolution ≥ 3) designs up to 13 factors, 128-run (resolution ≥ 4) designs up to 13 factors, 256-run (resolution ≥ 5) designs up to 17 factors, 512-run (resolution ≥ 5) designs up to 18 factors, 1024-run (resolution ≥ 6) designs up to 16 factors, 2048-run (resolution ≥ 7) up to 23 factors, and 4096-run (resolution ≥ 8) designs up to 24 factors. Catalogs of non-isomorphic minimum aberration designs, with run size up to 128, among these are listed in Appendix B. The remaining designs are available from the author (and for download at `http://ise.tamu.edu/metrology`, click on Publications, as supplement to Shrivastava and Ding (2010)). The computation times for generating these design catalogs is given in Table 9. Since there are neither any computational results available in literature nor any other (efficient) isomorphism check available for 2-level

Table 9. Computation times for generating catalogs of non-isomorphic minimum aberration 2-level regular fractional factorial split-plot designs.

Run-size	Resolution	Largest $n_1 + n_2$	CPU time (in mins)
32	3	20	13.05
64	3	13	16.07
128	4	13	12.16
256	5	17	11.37
512	5	18	10.65
1024	6	16	24.55 hours
2048	7	23	3.42 hours
4096	8	24	7.65 hours

regular fractional factorial split-plot designs, we do not present any computational comparisons.

Table 10 lists some selected 4096-run minimum aberration designs with 20 factors, i.e., $n_1 + n_2 = 20$. The whole plot factors are denoted by uppercase letters and sub-plot factors are denoted by lowercase letters. It may be noted that the minimum aberration designs are not unique. For example, there are four $2^{(11-1)+(9-7)}$ non-isomorphic minimum aberration designs.

Table 10. Selected 20-factor, 4096-run minimum aberration 2-level regular fractional factorial split-plot designs with resolution ≥ 8

$n_1.n_2.k_1.k_2$	defining words	word length pattern
1.19.0.8	$Abcdefgm$, $Abcdhijn$, $Abefhiko$, $Aceghjkp$, $Adfgijkq$, $bceghilr$, $Abfghjls$, $bdefijlt$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]

Table 10 Continued

$n_1.n_2.k_1.k_2$	defining words	word length pattern
2.18.0.8	$ABcdefgm$, $ABcdhijn$, $ABefhiko$, $Aceghjkp$, $Adfgijkq$, $Bceghilr$, $ABfghjls$, $Bdefijlt$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]
2.18.0.8	$ABcdefgm$, $ABcdhijn$, $ABefhiko$, $Aceghjkp$, $Adfgijkq$, $Bceghilr$, $ABfghjls$, $defghklt$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]
3.17.0.8	$ABCdefgm$, $ABCdhijn$, $ABefhiko$, $ACeghjkp$, $Adfgijkq$, $BCeghilr$, $ABfghjls$, $Bdefijlt$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]
3.17.0.8	$ABCdefgm$, $ABCdhijn$, $ABefhiko$, $ACeghjkp$, $Adfgijkq$, $BCeghilr$, $ABfghjls$, $defghklt$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]
3.17.0.8	$ABCdefgm$, $ABCdhijn$, $ABefhiko$, $ACeghjkp$, $Adfgijkq$, $BCeghilr$, $ABdghkls$, $BCdfiklt$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]
11.9.0.8	$ABCDEFlm$, $ABCGHIln$, $ADEGHJlo$, $BDFGIJlp$, $CEFHIJlq$, $CDFGHKlr$, $AEFGIKls$, $BDEHIKlt$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]
11.9.1.7	$ABCDEFGM$, $ABCDHIkn$, $ABEFHJko$, $ACEGIJkp$, $ACEFHIlq$, $BCDFHJlr$, $BEFGIJls$, $BCEGHklt$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]
11.9.1.7	$ABCDEFGM$, $ABCDHIkn$, $ABEFHJko$, $ACEGIJkp$, $ACEFHIlq$, $BCDFHJlr$, $BCEGHkls$, $CDFGIklt$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]
11.9.1.7	$ABCDEFGM$, $ABCDHIkn$, $ABEFHJko$, $ACEFHIlp$, $BCDEHJlq$, $ADEGHklr$, $ABFGIkls$, $CEFGJklt$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]
11.9.1.7	$ABCDEFGM$, $ABCDHIkn$, $ABEFHJko$, $ACEFHIlp$, $BCDEHJlq$, $ADEGHklr$, $ABFGIkls$, $DFHIJklt$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]
12.8.1.7	$ABCDEFGM$, $ABCDHIln$, $ABEFHJlo$, $ACEGIJlp$, $BCEGHKlq$, $ABFGIKlr$, $DEFGJKls$, $CFHIJKlt$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]
12.8.1.7	$ABCDEFGHIJKM$, $ABCDEFln$, $ABCGHIlo$, $ADEGHJlp$, $BDFGIJlq$, $CEFHIJlr$, $CDFGHKls$, $AEFGIKlt$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]
12.8.2.6	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIko$, $ACEGHJkp$, $ADFGIJkq$, $BCEGHIlr$, $ABFGHJls$, $BDEFIJlt$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]
12.8.2.6	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIko$, $ACEGHJkp$, $ADFGIJkq$, $BCEGHIlr$, $ABFGHJls$, $DEFGHklt$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]
12.8.2.6	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIko$, $ACEGHJkp$, $ADFGIJkq$, $BCEGHIlr$, $ABDGHkls$, $BCDFIklt$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]

Table 10 Continued

$n_1.n_2.k_1.k_2$	defining words	word length pattern
12.8.2.6	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIko$, $ACEGHJkp$, $BCEGHIlq$, $ABFGHJlr$, $DEFGHkls$, $ACDEIklt$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]
13.7.2.6	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIlo$, $ACEGHJlp$, $ADFGIJlq$, $BCDEHKlr$, $ABCGIKls$, $BEFGJKlt$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]
14.6.3.5	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$, $ACEGHIlp$, $BCEFHJlq$, $CDFGIJlr$, $ACDFHKls$, $ABDGIKlt$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]
15.5.4.4	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$, $ACEGHJKP$, $BCEGHIlq$, $ABFGHJlr$, $BDEFIJls$, $DEFGHKlt$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]
16.4.5.3	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$, $ACEGHJKP$, $ADFGIJKQ$, $BCEGHIlr$, $ABFGHJls$, $BDEFIJlt$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]

CHAPTER VII

CONCLUSION

In this chapter we first summarize the proposed methods in this dissertation and highlight the contributions. We then provide some thoughts on future work, including immediate extensions of the current work and some related problems.

VII.1. Summary

In this dissertation, we develop a new efficient approach for listing non-isomorphic catalogs of fractional factorial designs. We develop, both, a necessary and sufficient isomorphism check for testing two designs for isomorphism, and a new method for improving the efficiency of the sequential catalog generation algorithm.

We develop a new necessary and sufficient check for testing the isomorphism of two 2-level fractional factorial designs based on a bipartite 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 indicate that our proposed isomorphism check runs significantly faster than the existing checks in literature, including Lin and Sitter (2008)'s (necessary and conjectured sufficient) eigenvalue check, Clark and Dean (2001)'s (necessary and sufficient) check 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 number of factors, 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 $\frac{m(m-1)}{2}$ 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 is faster than Clark and Dean (2001)'s method.

Our graph representation also allows us to extend results in non-isomorphic graph generation literature to the non-isomorphic design generation problem. Using results from the graph isomorphism literature we improve the existing design generation algorithm of Lin and Sitter (2008) by further 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 the ability to handle large designs compared to Lin and Sitter (2008). Fig. 25 shows the largest size design catalogs that were generated over the years, since the problem was first proposed by Draper and Mitchell (1967). The figure includes only those publications that specifically generated non-isomorphic design catalogs. It is evident that the contribution of our method in increasing the capability to handle larger designs is significant.

Further, we showed that the graph based methods developed for 2-level regular fractional factorial designs are extensible to 2-level regular fractional factorial split-plot designs, certifying the extensibility of the graph based approach. This allows us to generate much larger split-plot design catalogs than those existing in current

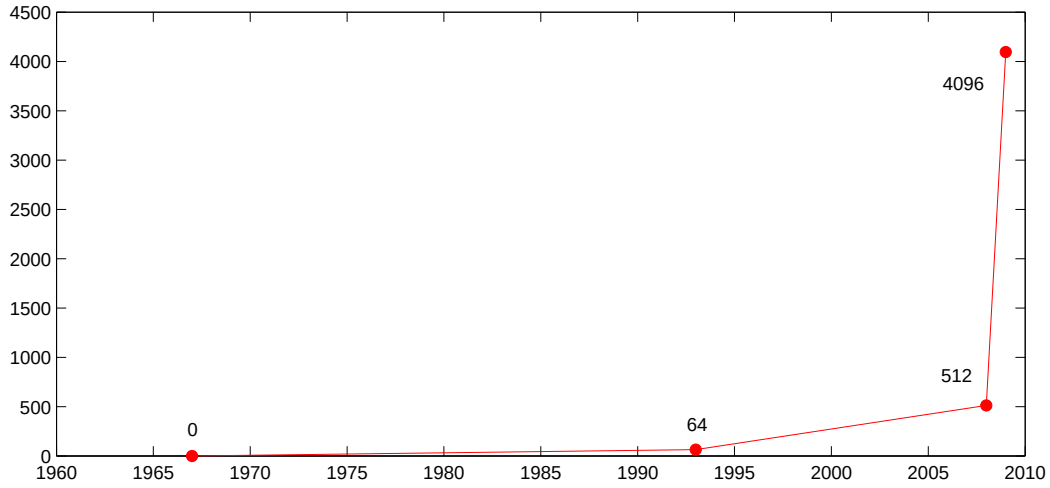


Fig. 25. History of size of 2-level regular fractional factorial non-isomorphic design catalogs generated in literature. Draper and Mitchell (1967) proposed the problem but could not generate complete non-isomorphic catalogs. Chen *et al.* (1993) generated up to 64-run size catalogs and Lin and Sitter (2008) generated up to 512-run size catalogs. We generated 4096-run size designs with our proposed method.

literature. We have been able to generate split-plot designs up to 4096 runs. This is a significant improvement over the 32-run size design catalogs provided by Bingham and Sitter (2001).

VII.2. Extensions to other design classes

In the previous section (and also earlier in this dissertation), we have mentioned about the extensible nature of our proposed framework for generating non-isomorphic designs. The 2-level split-plot designs do provide some support to this argument. In this section, we will look at some more general classes of fractional factorial designs to which our methods can be extended. We provide graph representations of the designs

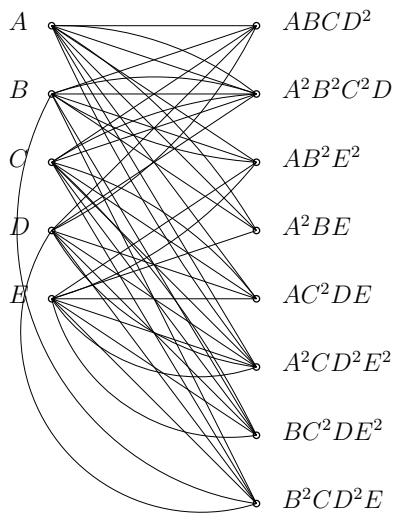


Fig. 26. Bipartite multigraph for the 3^{5-2} design with defining contrast subgroup $\{I,\ ABCD^2,\ A^2B^2C^2D,\ AB^2E^2,\ A^2BE,\ AC^2DE,\ A^2CD^2E^2,\ BC^2DE^2,\ B^2CD^2E\}$. Vertices on the left, set V_a, correspond to factors, and vertices on the right, set V_b, correspond to words in the defining contrast subgroup. Multiple edges denote the more than one levels of each factor.

that may lead to results similar to Theorem III.1.

VII.2.1. Multi-level and mixed-level regular fractional factorial designs

Multi-level regular fractional factorial designs are designs where all factors have the same number of levels, s, but $s > 2$. Mixed-level designs are those whose factors may have different number of levels. Since these designs are very similar to the 2-level regular fractional factorial designs, we can use similar graph representations for these. We will discuss the case of multi-level designs here by considering a 3-level design. The case for large s is almost identical but gets messier.

Multi-level regular designs can be represented as graphs by a natural extension of Algorithm III.1. For example, consider the 3^{5-2} design given by the defining contrast

subgroup $\{I,\ ABCD^2,\ A^2B^2C^2D,\ AB^2E^2,\ A^2BE,\ AC^2DE,\ A^2CD^2E^2,\ BC^2DE^2,\ B^2CD^2E\}$. We can construct the graph for this design by following the same recipe as given in Algorithm III.1, but with the additional instruction in *Step 4* of the algorithm that if a word has a squared term for a factor then we add two edges between the vertex for the word and the vertex for the factor. The graph so obtained is a bipartite multigraph. Fig. 26 shows the multigraph for the 3^{5-2} design considered above. Notice that there are two edges between the vertices $ABCD^2$ and D.

For the case of mixed-level designs, we can use the same method as that for multi-level designs to construct the graphs. Additionally, we color the vertices so that vertices corresponding to factors with the same number of levels have the same color. This gives us a colored multigraph.

Since the algorithms for constructing the graphs for multi-level and mixed level designs follow similar steps to those in Algorithm III.1 (and are hence one-to-one maps), we expect that proving the equivalence between the isomorphism problem for these classes and the corresponding graph classes should be possible. One difficulty in implementing this method is that we are not aware of any good implementations of graph isomorphism algorithms that can handle multi-level designs. *nauty* does not have this desired capability in its existing implementation, although the original algorithm in McKay (1981) is capable of handling multigraphs.

The extension of the result in Theorem IV.2 to the case of multi-level and mixed-level regular designs also seems possible. This is because the sequential algorithms for 2-level designs and multi-level designs are very similar. As at each step the new design is created by adding a defining word to the smaller designs.

	A	B	C	D
1	0	0	0	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0

Fig. 27. A 4-factor 5-run 2-level non-regular fractional factorial design.

VII.2.2. Non-regular designs

Since non-regular designs are by definition are those designs that do not have the special structure of regular designs (given by the defining words), the extension of results in this case is difficult. We provide a graph representation here that was suggested in McKay (2007) for problem of testing the isotopy of two matrices.

Consider the 4-factor, 5-run, 2-level non-regular fractional factorial design in

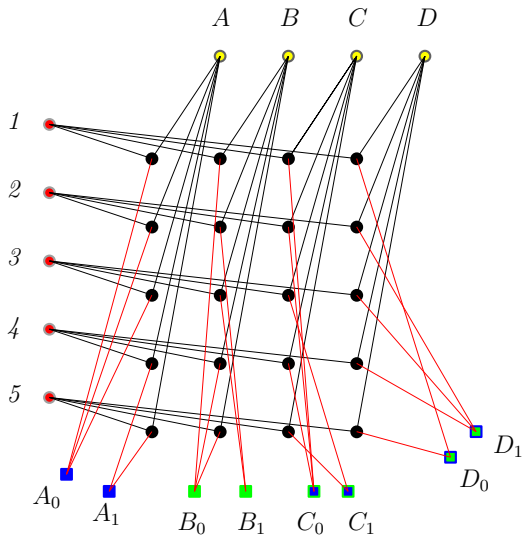


Fig. 28. A vertex-colored graph representation for the non-regular fractional factorial design in Fig. 27. The edges have been colored only to improve the readability.

Fig. 27. A graph representation for this design is shown in Fig. 28. The graph is constructed by adding a vertex for each cell in the design matrix, each factor in the design, each run in the design, and each of the level labels of each factor in the designs. This is a vertex-colored graph, with the factors having one color, run labels having another color and the pairs of levels of each factor having different sets of colors. That is, the colors of, say, level '0' of factors A and B are different.

The above representation seems one possible way of constructing the graphs, but it is not clear if this representation will allow proving the equivalence between the two isomorphism problems, and further lead to a necessary and sufficient isomorphism check for non-regular designs. The hypothesis remains to be verified. But if proven true, it may result in a fast isomorphism check for non-regular designs.

The example above considers a 2-level design, but it is easy to see that this graph representation can be extended to multi-level designs. For mixed-level designs, further partitioning of the factors into factors with same number of levels, as in the previous section, may be done.

VII.3. Related problems

In this section we discuss some problems that may be able to benefit from the methods and approaches developed in this dissertation.

VII.3.1. Constructing catalogs of optimal experimental designs

Usually it is desired that the complete set of all designs that are optimal based on some criterion be available. Examples of such criteria for regular fractional factorial designs are maximum resolution, minimum aberration, clear main effects and strongly clear effects.

The current approach to constructing these designs is to generate the entire catalog of designs and store it (off-line generation), and then use the optimality criteria to choose the optimal design(s) from this stored collection. It seems more desirable to instead be able to generate the collection of optimal designs in real-time. This could be done if the choice of the optimality criterion can be included as a constraint in the sequential generation algorithm. As this will in turn lead to a smaller intermediate set of designs at each step of the generation algorithm, it may be possible to explore much larger size designs than can be generated for off-line use.

It should be noted that among the criteria mentioned above, the resolution criterion can be used as a constraint in reducing the size of the intermediate set (see Section IV.2.1). But such results are not available for other criteria. In particular, it does not seem possible to get the collection of minimum aberration designs without generating the entire collection of designs (see also Bingham and Sitter (1999a)). But it is worth investigating what properties an optimality criteria should have for it to be usable as a constraint in the design generation procedure.

VII.3.2. Using experimenter's requirements as constraints in design generation

This problem is somewhat similar to the previous one. We again are interested in real-time generation of design catalogs by including constraints on the generation procedure. But here we want to take the experimenter's requirements as inputs. By requirements we mean the statistical effects that need to estimated from the subsequent analysis of experiment data. The question to be answered is how these requirements can be modeled so that they can be incorporated in the sequential design generation procedure in an efficient manner.

It can be imagined that some simple requirements can be incorporated easily in

the generation scheme. For example, if all the main effects and two-factor interaction effects need to be estimated then basically the requirement is that of a resolution IV design. But for a complex set of requirements, which cannot be translated to the design resolution, it is not clear if the generation algorithm can be sped up by using some constraints. For example if in a 7-factor experiment we are interested in estimating all main effects, three two factor interactions (AB, BC, CD) and a three factor interaction (AEF), then we need at least a resolution IV design, but not all resolution IV designs will be feasible solutions (e.g., any design with the word $ABCD$ in the defining contrast subgroup is infeasible).

Since these requirements can be complex, the possible solution seems to be through modeling these experimenter's requirements as graphs. Modeling these requirements as graphs, and also modeling the statistical capabilities of fractional factorial designs has been studied before. Some of these approaches include Taguchi's requirements graph and approaches for representing aliasing relationships of designs (Sun and Wu, 1994, Wu and Chen, 1992) as graphs. But these representations cannot usually be used for representing high order interactions, and also they are not compatible with our graph representations of the designs.

The problem therefore is to devise new graph models, to capture these requirements, that can be easily incorporated in our design generation algorithm. For the practitioner, this could allow generating the smaller feasible set of designs in real-time, and may be even larger size designs than those available in off-line catalogs.

VII.3.3. Graph models for complicated engineering system designs

Fig. 29 shows a schematic of a phone quality testing system (Shrivastava *et al.*, 2006). The setup shown is used for testing multiple phones at a time. Each phone is loaded on one of the fixtures in the system, and a computer controller executes

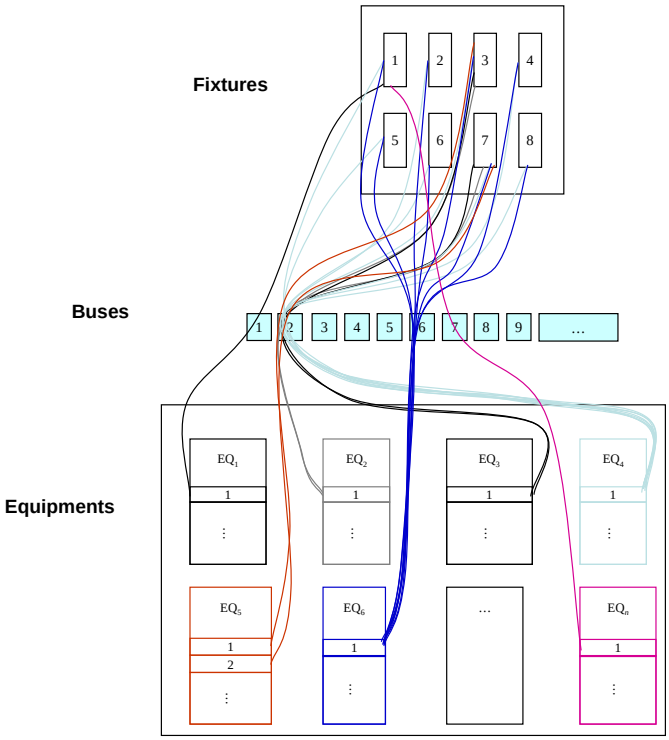


Fig. 29. A complicated phone quality testing system.

a number of tests using the equipments in the system. There are multiple units of each equipment available in the system, and these are shared by the fixtures. The communication between the equipments and the fixtures is through communication buses. Due to a limited number of buses, not every unit of every equipment can be used by each fixture, but only a select few can be assigned to each fixture. The assignments also include the specific bus which may be used by a fixture for accessing an equipment unit. The problem of designing this system is then deciding how should the different equipment units be allocated to the different fixtures, and further which communication buses can be used for each of these communications.

The system design problem described above, although it appears quite different, has a lot of similarity to the design generation problem. Since the different units of

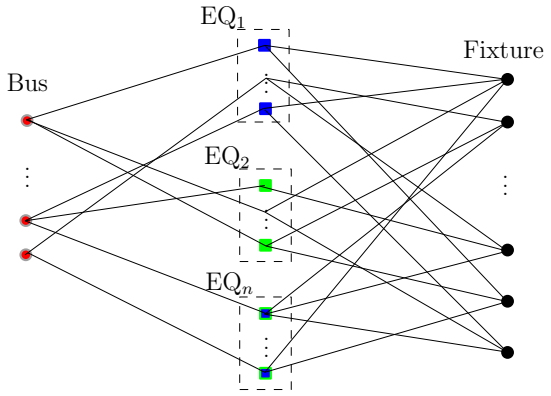


Fig. 30. A graph representation of the phone quality testing system in Fig. 29.

an equipment are identical to each other, all the buses are identical to each other and all the fixtures are identical to each other, a lot of the possible system designs or feasible solutions are isomorphic to each other.

Fig. 30 shows a vertex-colored graph representation of a possible design for the phone quality testing system. Each feasible solution for the design problem can be represented as a vertex-colored graph, and thus all non-isomorphic feasible solutions are non-isomorphic graphs. These solutions can be generated by a sequential procedure (McKay, 1998), similar to the sequential generation of designs. But the problem of finding the best design is an optimization problem. Solving this optimization is a challenging question since most optimization techniques require some notion of distance between solution points, but it is not clear how such a distance measure can be defined.

This optimization problem can also be related to the problem of finding optimal designs. An efficient solution to either of the problems may, thus, benefit the other problem.

VII.4. Conclusion

This dissertation contributes to both the research and practice of design of experiments. It provides a new approach to modeling experimental designs as graphs, thereby providing new opportunities for developing efficient methods for constructing experimental designs. This is, to the best of our knowledge, the first approach in modeling fractional factorial designs as graphs for the purpose of constructing them efficiently. For the practitioners, we provide complete design catalogs that were not available before.

To summarize, the following our the contributions of this research:

1. A new necessary and sufficient check for 2-level regular fractional factorial design isomorphism.

2. A generation algorithm that can generate catalogs of non-isomorphic 2-level regular fractional factorial designs much faster than any of the previous methods.

3. A unified framework customizable to different classes of fractional factorial designs.

4. Catalogs of up to 4096-run non-isomorphic 2-level regular fractional factorial and 2-level regular fractional factorial split-plot designs, not available earlier in literature.

REFERENCES

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

Angelopoulos, P., Evangelaras, H., Koukouvinos, C. and Lappas, E. (2007) An effective step-down algorithm for the construction and the identification of nonisomorphic orthogonal arrays. *Metrika*, **66**(2), 139–149.

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

Balaban, A.T. (1985) Applications of graph theory in chemistry. *Journal of Chemical Information and Computer Sciences*, **25**(3), 334–343.

Bingham, D. and Mukerjee, R. (2006) Detailed wordlength pattern of regular fractional factorial split-plot designs in terms of complementary sets. *Discrete Mathematics*, **306**(14), 1522 – 1533, r.C. Bose Centennial Symposium on Discrete Mathmematics and Applications.

Bingham, D., Schoen, E.D. and Sitter, R.R. (2004) Designing fractional factorial split-plot experiments with few whole-plot factors. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, **53**(2), 325–339.

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

Bingham, D. and Sitter, R.R. (1999b) Some theoretical results for fractional factorial split-plot designs. *The Annals of Statistics*, **27**(4), 1240–1255.

Bingham, D. and Sitter, R.R. (2001) Design issues in fractional factorial split-plot experiments. *Journal of Quality Technology*, **33**(1), 2–15.

Bingham, D. and Sitter, R.R. (2003) Fractional factorial split-plot designs for robust parameter experiments. *Technometrics*, **45**(1), 80–89.

Bisgaard, S. (1992) Industrial use of statistically designed experiments: case study references and some historical anecdotes. *Quality Engineering*, **4**(4), 547–562.

Bose, R. (1963) Strongly regular graphs, partial geometries and partially balanced designs. *Pacific Journal of Mathematics*, **13**(2), 389–419.

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

Box, G.E.P. and Hunter, J.S. (1961b) The 2^{k-p} fractional factorial designs Part II. *Technometrics*, **3**(4), 449–458.

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

Box, G.E.P. and Jones, S. (1992) Split-plot designs for robust product experimentation. *Journal of Applied Statistics*, **19**(1), 3–26.

Butler, N.A. (2004) Construction of Two-Level Split-Lot Fractional Factorial Designs for Multistage Processes. *Technometrics*, **46**(4), 445–451.

Calvetti, D., Kim, S.M. and Reichel, L. (2002) The restarted QR-algorithm for eigenvalue computation of structured matrices. *Journal of Computational and Applied Mathematics*, **149**(2), 415–422.

Cameron, P. and Mary, Q. (2004) Automorphisms of graphs, in *Topics in Algebraic Graph Theory*, edited by Beineke, L.W., Wilson, R.J. and Cameron, P.J., number 102 in Encyclopedia of Mathematics and its Applications, Cambridge University Press, New York, NY, pp. 137–155.

Cameron, P. and van Lint, J. (1980) *Graph Theory, Coding Theory and Block Designs*, number 43 in London Mathematical Society Lecture Note Series, Cambridge University Press, New York, NY.

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. and Lin, D.K.J. (1991) On the identity relationships of 2^{k-p} designs. *Journal of Statistical Planning and Inference*, **28**(1), 95–98.

Chen, J., Sun, D.X. and 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, C.S. (1981) Maximizing the total number of spanning trees in a graph: two related problems in graph theory and optimum design theory. *Journal of Combinatorial Theory, Series B*, **31**(2), 240–248.

Cheng, S.W. and 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. and Dean, A.M. (2001) Equivalence of fractional factorial designs. *Statistica Sinica*, **11**, 537–547.

Colbourn, M. and Colbourn, C. (1981) Concerning the complexity of deciding isomorphism of block designs. *Discrete Applied Mathematics*, **3**(3), 155–162.

Cook, W., Cunningham, W., Pulleyblank, W. and Schrijver, A. (1998) *Combinatorial Optimization*, Wiley, New York.

Daniel, C. (1962) Sequences of Fractional Replicates in the 2^{p-q} Series. *Journal of the American Statistical Association*, **57**(298), 403–429.

Diestel, R. (2005) *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*, third edition, Springer-Verlag, New York, NY.

Draper, N.R. and Mitchell, T.J. (1967) The construction of saturated 2^{k-p}_R designs. *The Annals of Mathematical Statistics*, **38**(4), 1110–1126.

Draper, N.R. and 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.

Fang, K.T., Lin, D.K.J., Winker, P. and Zhang, Y. (2000) Uniform design: theory and application. *Technometrics*, **42**(3), 237–248.

Fortin, S. (1996) The graph isomorphism problem, Technical Report TR 96–20, Department of Computer Science, The University of Alberta, Edmonton, Alberta, Canada.

Fries, A. and Hunter, W.G. (1980) Minimum aberration 2^{k-p} designs. *Technometrics*, **22**(4), 601–608.

Gregory, W.L. and Taam, W. (1996) A split-plot experiment in an automobile windshield fracture resistance test. *Quality & Reliability Engineering International*, **12**(2), 79–87.

Hayes, B. (2000a) Graph theory in practice: Part I. *American Scientist*, **88**(1), 9–13.

Hayes, B. (2000b) Graph theory in practice: Part II. *American Scientist*, **88**(2), 104–109.

Hedayat, A.S., Sloane, N.J.A. and Stufken, J. (1999) *Orthogonal Arrays: Theory and Applications*, Springer Series in Statistics, Springer-Verlag, New York.

Huang, P., Chen, D. and Voelkel, J.O. (1998) Minimum-aberration two-Level split-plot designs. *Technometrics*, **40**(4), 314–326.

Ilzarbe, L., lvarez, M.J., Viles, E. and Tanco, M. (2008) Practical applications of design of experiments in the field of engineering: a bibliographical review. *Quality and Reliability Engineering International*, **24**(4), 417–428.

Kaski, P. (2002) Isomorph-free exhaustive generation of combinatorial designs, Research Report HUT-TCS-A70, Laboratory for Theoretical Computer Science, Department of Computer Science and Engineering, Helsinki University of Technology, Finland.

Katsaounis, T. and 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 *Computational and Constructive Design Theory*, edited by Wallis, W.D., volume 368, Kluwer Academic Publishers, Dordrecht, The Netherlands, pp. 135–175.

Kowalski, S. and Potcner, K. (2003) How to recognize a split-plot experiment. *Quality Progress*, **36**(11), 60–66.

Kulahci, M. and Bisgaard, S. (2005) The use of Plackett-Burman designs to construct split-plot designs. *Technometrics*, **47**(4), 495–501.

Lawson, C., Hanson, R., Kincaid, D. and Krogh, F. (1979) Basic linear algebra subprograms for Fortran usage. *ACM Transactions on Mathematical Software (TOMS)*, **5**(3), 308–323.

Lin, C.D. and 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. and Lin, D.K.J. (2001) On the isomorphism of fractional factorial designs. *Journal of Complexity*, **17**(1), 86–97.

Mason, O. and Verwoerd, M. (2007) Graph theory and networks in biology. *IET Systems Biology*, **1**(2), 89–119.

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, Canberra, ACT, 0200, Australia.

McKay, B.D. (2007) *nauty User's Guide (Version 2.4)*, Computer Science Department, Australian National University, Canberra, ACT, 0200, Australia.

McLeod, R.G. and Brewster, J.F. (2004) The design of blocked fractional factorial split-plot experiments. *Technometrics*, **46**(2), 135–146.

Mee, R.W. (2004) Efficient two-level designs for estimating main effects and two-factor interactions. *Journal of Quality Technology*, **36**(4), 400–412.

Mee, R.W. and Bates, R.L. (1998) Split-lot designs: experiments for multistage batch processes. *Technometrics*, **40**(2), 127–140.

Montgomery, D.C. (2000) *Design and Analysis of Experiments*, fifth edition, John Wiley & Sons, New York, NY.

Mukerjee, R. and Fang, K.T. (2002) Fractional factorial split-plot designs with minimum aberration and maximum estimation capacity. *Statistica Sinica*, **12**, 885–903.

Plackett, R.L. and Burman, J.P. (1946) The design of optimum multifactorial experiments. *Biometrika*, **33**(4), 305–325.

Prvan, T. and Street, D.J. (2002) An annotated bibliography of application papers using certain classes of fractional factorial and related designs. *Journal of Statistical Planning and Inference*, **106**(1-2), 245–269.

Rao, C.R. (1947) Factorial experiments derivable from combinatorial arrangements of arrays. *Supplement to the Journal of the Royal Statistical Society*, **9**(1), 128–139.

Read, R.C. and Corneil, D.G. (1977) The graph isomorphism disease. *Journal of Graph Theory*, **1**(1), 339–363.

Robinson, D.J.S. (1995) *A Course in the Theory of Groups*, volume 80 of *Graduate Texts in Mathematics*, second edition, Springer-Verlag, New York, NY.

Rotman, J.J. (1995) *An Introduction to the Theory of Groups*, volume 148 of *Graduate Texts in Mathematics*, fourth edition, Springer-Verlag, New York, NY.

Roy, R.K. (2001) *Design of Experiments Using The Taguchi Approach: 16 Steps to Product and Process Improvement*, Wiley, New York.

Schoen, E. and Nguyen, M. (2007) Enumeration and classification of orthogonal arrays, Research paper 2007:21, University of Antwerp, Belgium, Faculty of Applied Economics, submitted to *Technometrics*.

Shrivastava, A.K. (2009) Graph based isomorph-free generation of two-level regular fractional factorial split-plot designs, Working paper, Department of Industrial and Systems Engineering, Texas A&M University, College Station, TX.

Shrivastava, A.K. and Ding, Y. (2010) Graph based isomorph-free generation of two-level regular fractional factorial designs. *Journal of Statistical Planning and Inference*, **140**(1), 169–179.

Shrivastava, A.K., Ding, Y., Coody, J., Niu, F. and Ceglarek, D. (2006) Modeling, analysis and design of complex quality testing systems using a hierarchical simulation framework, in *Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control*, Fort Lauderdale, FL, pp. 691–696.

Stimming, C. (2007) *Lapack++ 2.5.2*, Hamburg, Germany.

Stufken, J. and Tang, B. (2007) Complete enumeration of two-level orthogonal arrays of strength d with $d + 2$ constraints. *Annals of Statistics*, **35**(2), 793–814.

Sun, D.X., Li, W. and 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 & Statistics, Stony Brook University, New York, NY.

Sun, D.X. and Wu, C.F.J. (1994) Interaction graphs for three-level fractional factorial designs. *Journal of Quality Technology*, **26**(4), 297–307.

Tang, B. and Deng, L.Y. (1999) Minimum G_2-aberration for nonregular fractional factorial designs. *The Annals of Statistics*, **27**(6), 1914–1926.

Wu, C.F.J. and Chen, Y. (1992) A graph-aided method for planning two-level experiments when certain interactions are important. *Technometrics*, **34**(2), 162–175.

Wu, C.F.J. and Hamada, M. (2000) *Experiments: Planning, Analysis, and Parameter Design Optimization*, John Wiley & Sons, New York, NY.

Xu, H. (2005) A catalogue of three-level regular fractional factorial designs. *Metrika*, **62**(2), 259–281.

Yang, Y.H. and Speed, T. (2002) Design issues for cDNA microarray experiments. *Nature Reviews Genetics*, **3**(8), 579–588.

Zemlyachenko, V., Korneenko, N. and Tyshkevich, R. (1985) Graph isomorphism problem. *Journal of Mathematical Sciences*, **29**(4), 1426–1481.

Zhu, Y. and Zeng, P. (2005) On the coset pattern matrices and minimum m-aberration of 2^{n-p} designs. *Statistica Sinica*, **15**(3), 717.

APPENDIX A

LIST OF NON-ISOMORPHIC 2-LEVEL REGULAR FRACTIONAL FACTORIAL DESIGNS

This appendix gives tables of some good 1024-run (Table 11) and all of 2048-run (Table 12) and 4096-run (Table 13) designs. The defining words for each of the designs are given. Additionally, the word length patterns of the designs are given. Note that the first element of the word length pattern is a_R, where R is the resolution of the design.

Table 11. Best two 1024-run 2-level regular fractional factorial designs with resolution ≥ 6 by minimum aberration criterion

$n-k.x$	defining words	word length pattern
11-1.1	$ABCDEFGHIJK$	[0, 0, 0, 0, 0, 1]
11-1.2	$ABCDEFGHIK$	[0, 0, 0, 0, 1, 0]
12-2.1	$ABCDEFGK$, $ABCDHIJL$	[0, 0, 3, 0, 0, 0, 0]
12-2.2	$ABCDEFK$, $ABCGHIJL$	[0, 1, 1, 1, 0, 0, 0]
13-3.1	$ABCDEFK$, $ABCGHIL$, $ADEGHJM$	[0, 4, 3, 0, 0, 0, 0, 0]
13-3.2	$ABCDEK$, $ABCFGHL$, $ADFGIJM$	[1, 3, 2, 1, 0, 0, 0, 0]
14-4.1	$ABCDEFK$, $ABCGHIL$, $ADEGHJM$, $BDFGIJN$	[0, 8, 7, 0, 0, 0, 0, 0, 0]
14-4.2	$ABCDEK$, $ABFGHL$, $ACDFGIM$, $CEFHIJN$	[2, 6, 5, 2, 0, 0, 0, 0, 0]
15-5.1	$ABCDEFK$, $ABCGHIL$, $ADEGHJM$, $BDFGIJN$, $CEFHIJO$	[0, 15, 15, 0, 0, 0, 0, 0, 0, 1]
15-5.2	$ABCDEK$, $ABFGHL$, $ACDFGIM$, $BCDFGJN$, $CEFHIJO$	[3, 12, 11, 4, 0, 0, 0, 0, 1, 0]
16-6.1	$ABCDEK$, $ABFGHL$, $ACFIJM$, $BCDFGIN$, $BCEFHJO$, $ABCDEFGHIJP$	[6, 25, 15, 0, 10, 6, 0, 0, 0, 1, 0]
16-6.2	$ABCDEK$, $ABFGHL$, $CDFGIM$, $CEFHJN$, $ADEFHIO$, $ADEFGJP$	[8, 24, 13, 0, 8, 8, 2, 0, 0, 0, 0]
17-7.1	$ABCDEK$, $ABFGHL$, $CDFGIM$, $CEFHJN$, $ADEFHIO$, $ADEFGJP$, $ABDGIJQ$	[12, 41, 25, 0, 20, 22, 6, 0, 0, 1, 0, 0]
17-7.2	$ABCDEK$, $ABFGHL$, $CDFGIM$, $CEFHJN$, $AEGIJO$, $ADEFHIP$, $BCEGHIQ$	[13, 40, 25, 0, 18, 24, 6, 0, 1, 0, 0, 0]

Table 11 Continued

$n-k.x$	defining words	word length pattern
18-8.1	$ABCDEK$, $ABFGHL$, $CDFGIM$, $CEFHJN$, $AEGIJO$, $ADEFHIP$, $BCEGHIQ$, $BDEFGJR$	[19, 66, 45, 0, 42, 60, 18, 0, 3, 2, 0, 0, 0]
18-8.2	$ABCDEK$, $ABFGHL$, $CDFGIM$, $CEFHJN$, $AEGIJO$, $BDHIJP$, $ADEFHIQ$, $BCEGHIR$	[20, 64, 46, 0, 40, 64, 16, 0, 4, 0, 1, 0, 0]
19-9.1	$ABCDEK$, $ABFGHL$, $CDFGIM$, $CEFHJN$, $AEGIJO$, $BDHIJP$, $ADEFHIQ$, $BCEGHIR$, $BDEFGJS$	[28, 104, 78, 0, 88, 144, 48, 0, 12, 8, 1, 0, 0, 0]
19-9.2	$ABCDEK$, $ABCFGL$, $ABDFHM$, $ACDFIN$, $AEGHIO$, $BDEGJP$, $CFGHJQ$, $DEFHIJR$, $BCDFGIJS$	[46, 56, 81, 72, 81, 72, 46, 56, 0, 0, 0, 0, 1, 0]
20-10.1	$ABCDEK$, $ABFGHL$, $CDFGIM$, $CEFHJN$, $AEGIJO$, $BDHIJP$, $ADEFHIQ$, $BCEGHIR$, $BDEFGJS$, $ACDGHJT$	[40, 160, 130, 0, 176, 320, 120, 0, 40, 32, 5, 0, 0, 0, 0]
20-10.2	$ABCDEK$, $ABCFGL$, $ABDFHM$, $ACDFIN$, $AEGHIO$, $BCDFJP$, $CEGHJQ$, $DEGIJR$, $EFHIJS$, $BGHIJT$	[90, 0, 255, 0, 332, 0, 255, 0, 90, 0, 0, 0, 0, 0, 1]
21-11.1	$ABCDEK$, $ABFGHL$, $CDFGIM$, $CEFHJN$, $AEGIJO$, $BDHIJP$, $ADEFHIQ$, $BCEGHIR$, $BDEFGJS$, $ACDGHJT$, $ABCFIJU$	[56, 240, 210, 0, 336, 672, 280, 0, 120, 112, 21, 0, 0, 0, 0, 0]
21-11.2	$ABCDEK$, $ABCFGL$, $ABDFHM$, $ACEGHN$, $BCEFIO$, $CDEGIP$, $ABEHIQ$, $BDEFJR$, $ABGHJS$, $FGHIJT$, $ACDFGIJU$	[128, 0, 410, 0, 608, 0, 680, 0, 160, 0, 61, 0, 0, 0, 0, 0]
22-12.1	$ABCDEK$, $ABFGHL$, $CDFGIM$, $CEFHJN$, $AEGIJO$, $BDHIJP$, $ADEFHIQ$, $BCEGHIR$, $BDEFGJS$, $ACDGHJT$, $ABCFIJU$, $ABCDEFGHIJV$	[77, 352, 330, 0, 616, 1344, 616, 0, 330, 352, 77, 0, 0, 0, 0, 0, 1]
22-12.2	$ABCDEK$, $ABCFGL$, $ABDFHM$, $ACEGHN$, $ACDFIO$, $BCEGIP$, $BCDFJQ$, $BDEGJR$, $DEFHJS$, $AFGIJT$, $ACDGHIJU$, $ABCDEFHIJV$	[183, 0, 600, 0, 1233, 0, 1324, 0, 585, 0, 155, 0, 15, 0, 0, 0, 0]
23-13.1	$ABCDEK$, $ABCFGL$, $ABDFHM$, $ACEGHN$, $ACDFIO$, $BCEGIP$, $BCDFJQ$, $ABEGJR$, $ADHIJS$, $CGHIJT$, $BCDEFHIU$, $ABCDGHIV$, $ACEFGIJW$	[251, 0, 899, 0, 2235, 0, 2697, 0, 1545, 0, 496, 0, 65, 0, 3, 0, 0, 0]
23-13.2	$ABCDEK$, $ABCFGL$, $ABDFHM$, $ACEGHN$, $ACDFIO$, $BCEGIP$, $BCDFJQ$, $ABEGJR$, $ADHIJS$, $BGHIJT$, $BCDEFHIU$, $BCEFGHJV$, $ABCDGIJW$	[252, 0, 890, 0, 2268, 0, 2632, 0, 1620, 0, 445, 0, 84, 0, 0, 0, 0, 0]
24-14.1	$ABCDEK$, $ABCFGL$, $ABDFHM$, $ACEGHN$, $ACDFIO$, $BCEGIP$, $BCDFJQ$, $ABEGJR$, $ADHIJS$, $BGHIJT$, $BCDEFHIU$, $BCEFGHJV$, $ABCDGIJW$, $ABEFHIJX$	[336, 0, 1335, 0, 3888, 0, 5264, 0, 3888, 0, 1335, 0, 336, 0, 0, 0, 0, 0, 1]

Table 12. All 2048-run 2-level regular fractional factorial designs with resolution ≥ 7

$n-k.x$	defining words	word length pattern
12-1.1	$ABCDEFL$	[1, 0, 0, 0, 0, 0]
12-1.2	$ABCDEFGL$	[0, 1, 0, 0, 0, 0]
12-1.3	$ABCDEFGHL$	[0, 0, 1, 0, 0, 0]
12-1.4	$ABCDEFGHIL$	[0, 0, 0, 1, 0, 0]
12-1.5	$ABCDEFGHIJL$	[0, 0, 0, 0, 1, 0]
12-1.6	$ABCDEFGHIJKL$	[0, 0, 0, 0, 0, 1]
13-2.1	$ABCDEFL$, $ABCGHIM$	[2, 1, 0, 0, 0, 0, 0]
13-2.2	$ABCDEFL$, $ABGHIJM$	[2, 0, 0, 1, 0, 0, 0]
13-2.3	$ABCDEFL$, $AGHIJKM$	[2, 0, 0, 0, 0, 1, 0]
13-2.4	$ABCDEFL$, $ABCGHIJM$	[1, 1, 1, 0, 0, 0, 0]
13-2.5	$ABCDEFL$, $ABGHIJKM$	[1, 1, 0, 0, 1, 0, 0]
13-2.6	$ABCDEFL$, $ABCGHIJKM$	[1, 0, 1, 1, 0, 0, 0]
13-2.7	$ABCDEFGL$, $ABCDHIJM$	[0, 3, 0, 0, 0, 0, 0]
13-2.8	$ABCDEFGL$, $ABCHIJKM$	[0, 2, 0, 1, 0, 0, 0]
13-2.9	$ABCDEFGL$, $ABCDHIJKM$	[0, 1, 2, 0, 0, 0, 0]
14-3.1	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$	[4, 3, 0, 0, 0, 0, 0, 0]
14-3.2	$ABCDEFL$, $ABCGHIM$, $ABDGJKN$	[3, 3, 0, 0, 1, 0, 0, 0]
14-3.3	$ABCDEFL$, $ABCGHIM$, $ADEGJKN$	[3, 2, 1, 1, 0, 0, 0, 0]
14-3.4	$ABCDEFL$, $ABCGHIM$, $DEFGJKN$	[4, 2, 0, 0, 0, 1, 0, 0]
14-3.5	$ABCDEFL$, $ABCGHIM$, $DEGHJKN$	[4, 1, 0, 2, 0, 0, 0, 0]
14-3.6	$ABCDEFL$, $ABCGHIM$, $ADEGHJKN$	[2, 3, 2, 0, 0, 0, 0, 0]
14-3.7	$ABCDEFGL$, $ABCDHIJM$, $ABEFHIKN$	[0, 7, 0, 0, 0, 0, 0, 0]
15-4.1	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $BDFGIJO$	[8, 7, 0, 0, 0, 0, 0, 0, 0]
15-4.2	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $BDFGHKO$	[7, 6, 0, 0, 1, 1, 0, 0, 0]
15-4.3	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $ADFGIKO$	[6, 7, 0, 0, 2, 0, 0, 0, 0]
15-4.4	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $BDFGIKO$	[6, 5, 2, 2, 0, 0, 0, 0, 0]
15-4.5	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $BCDEGHKO$	[7, 7, 0, 0, 0, 0, 0, 0, 1]
15-4.6	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $BDFGIJKO$	[4, 7, 4, 0, 0, 0, 0, 0, 0]
15-4.7	$ABCDEFGL$, $ABCDHIJM$, $ABEFHIKN$, $ACEGHJKO$	[0, 15, 0, 0, 0, 0, 0, 0, 0]
16-5.1	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $BDFGIJO$, $CEFHIJP$	[15, 15, 0, 0, 0, 0, 0, 0, 1, 0]
16-5.2	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $BDFGIJO$, $CDFGHKP$	[12, 13, 0, 0, 4, 2, 0, 0, 0, 0]

Table 12 Continued

$n-k.x$	defining words	word length pattern
16-5.3	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $BDFGIJO$, $CEFHIKP$	[11, 11, 4, 4, 0, 0, 0, 0, 1, 0]
16-5.4	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $BDFGIJO$, $CEFHIJKP$	[8, 14, 8, 0, 0, 0, 0, 0, 0, 1]
16-5.5	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $ADFGIKO$, $ABFHJKP$	[10, 15, 0, 0, 6, 0, 0, 0, 0, 0]
16-5.6	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $BCDEGHKO$, $BDFGIJKP$	[7, 15, 8, 0, 0, 0, 0, 0, 1, 0]
16-5.7	$ABCDEFGL$, $ABCDHIJM$, $ABEFHIKN$, $ACEGHJKO$, $ADFGIJKP$	[0, 30, 0, 0, 0, 0, 0, 0, 0, 1]
17-6.1	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $BDFGIJO$, $CEFHIJP$, $CDFGHKQ$	[21, 25, 0, 0, 10, 6, 0, 0, 1, 0, 0]
17-6.2	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $BDFGIJO$, $CDFGHKP$, $AEFGIKQ$	[20, 25, 0, 0, 12, 6, 0, 0, 0, 0, 0]
17-6.3	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $ADFGIKO$, $ABFHJKP$, $ACEIJKQ$	[16, 30, 0, 0, 16, 0, 0, 0, 0, 1, 0]
18-7.1	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $BDFGIJO$, $CEFHIJP$, $CDFGHKQ$, $AEFGIKR$	[33, 45, 0, 0, 30, 18, 0, 0, 1, 0, 0, 0]
18-7.2	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $BDFGIJO$, $CDFGHKP$, $AEFGIKQ$, $BCEGJKR$	[32, 46, 0, 0, 32, 16, 0, 0, 0, 1, 0, 0]
19-8.1	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $BDFGIJO$, $CEFHIJP$, $CDFGHKQ$, $AEFGIKR$, $BDEHIKS$	[52, 78, 0, 0, 72, 48, 0, 0, 4, 1, 0, 0, 0]
20-9.1	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $BDFGIJO$, $CEFHIJP$, $CDFGHKQ$, $AEFGIKR$, $BDEHIKS$, $BCEGJKT$	[80, 130, 0, 0, 160, 120, 0, 0, 16, 5, 0, 0, 0, 0]
21-10.1	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $BDFGIJO$, $CEFHIJP$, $CDFGHKQ$, $AEFGIKR$, $BDEHIKS$, $BCEGJKT$, $ABFHJKU$	[120, 210, 0, 0, 336, 280, 0, 0, 56, 21, 0, 0, 0, 0, 0]
22-11.1	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $BDFGIJO$, $CEFHIJP$, $CDFGHKQ$, $AEFGIKR$, $BDEHIKS$, $BCEGJKT$, $ABFHJKU$, $ACDIJKV$	[176, 330, 0, 0, 672, 616, 0, 0, 176, 77, 0, 0, 0, 0, 0, 0]
23-12.1	$ABCDEFL$, $ABCGHIM$, $ADEGHJN$, $BDFGIJO$, $CEFHIJP$, $CDFGHKQ$, $AEFGIKR$, $BDEHIKS$, $BCEGJKT$, $ABFHJKU$, $ACDIJKV$, $ABCDEFGHIJKW$	[253, 506, 0, 0, 1288, 1288, 0, 0, 506, 253, 0, 0, 0, 0, 0, 1]

Table 13. All 4096-run 2-level regular fractional factorial designs with resolution ≥ 8

$n-k.x$	defining words	word length pattern
13-1.1	$ABCDEFGM$	[1, 0, 0, 0, 0, 0]
13-1.2	$ABCDEFGHM$	[0, 1, 0, 0, 0, 0]
13-1.3	$ABCDEFGHIM$	[0, 0, 1, 0, 0, 0]
13-1.4	$ABCDEFGHIJM$	[0, 0, 0, 1, 0, 0]
13-1.5	$ABCDEFGHIJKM$	[0, 0, 0, 0, 1, 0]
13-1.6	$ABCDEFGHIJKLM$	[0, 0, 0, 0, 0, 1]
14-2.1	$ABCDEFGM$, $ABCDHIJN$	[3, 0, 0, 0, 0, 0, 0]
14-2.2	$ABCDEFGM$, $ABCHIJKN$	[2, 0, 1, 0, 0, 0, 0]
14-2.3	$ABCDEFGM$, $ABHIJKLN$	[2, 0, 0, 0, 1, 0, 0]
14-2.4	$ABCDEFGM$, $ABCDHIJKN$	[1, 2, 0, 0, 0, 0, 0]
14-2.5	$ABCDEFGM$, $ABCHIJKLN$	[1, 1, 0, 1, 0, 0, 0]
14-2.6	$ABCDEFGM$, $ABCDHIJKLN$	[1, 0, 2, 0, 0, 0, 0]
14-2.7	$ABCDEFGHM$, $ABCDIJKLN$	[0, 2, 1, 0, 0, 0, 0]
15-3.1	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$	[7, 0, 0, 0, 0, 0, 0, 0]
15-3.2	$ABCDEFGM$, $ABCDHIJN$, $ABCEHKLO$	[6, 0, 0, 0, 1, 0, 0, 0]
15-3.3	$ABCDEFGM$, $ABCDHIJN$, $ABEFHKLO$	[5, 0, 2, 0, 0, 0, 0, 0]
15-3.4	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKLO$	[3, 4, 0, 0, 0, 0, 0, 0]
16-4.1	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$, $ACEGHJKP$	[15, 0, 0, 0, 0, 0, 0, 0, 0]
16-4.2	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$, $CDEFHILP$	[14, 0, 0, 0, 0, 0, 0, 0, 1]
16-4.3	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$, $ACEGHILP$	[13, 0, 0, 0, 2, 0, 0, 0, 0]
16-4.4	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$, $ACEGHJLP$	[11, 0, 4, 0, 0, 0, 0, 0, 0]
16-4.5	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$, $ACEGHJKLP$	[7, 8, 0, 0, 0, 0, 0, 0, 0]
17-5.1	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$, $ACEGHJKP$, $ADFGIJKQ$	[30, 0, 0, 0, 0, 0, 0, 0, 1, 0]
17-5.2	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$, $ACEGHJKP$, $CDEFHILQ$	[22, 0, 8, 0, 0, 0, 0, 0, 1, 0]
17-5.3	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$, $ACEGHJKP$, $BCEGHILQ$	[25, 0, 0, 0, 6, 0, 0, 0, 0, 0]
17-5.4	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$, $ACEGHJKP$, $ADFGIJKLQ$	[15, 15, 0, 0, 0, 0, 0, 0, 0, 1]
17-5.5	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$, $CDEFHILP$, $ACEGHJKLQ$	[14, 16, 0, 0, 0, 0, 0, 0, 1, 0]
18-6.1	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$, $ACEGHJKP$, $ADFGIJKQ$, $BCEGHILR$	[46, 0, 0, 0, 16, 0, 0, 0, 1, 0, 0]

Table 13 Continued

$n-k.x$	defining words	word length pattern
18-6.2	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$, $ACEGHJKP$, $BCEGHILQ$, $ABFGHJLR$	[45, 0, 0, 0, 18, 0, 0, 0, 0, 0, 0]
19-7.1	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$, $ACEGHJKP$, $ADFGIJKQ$, $BCEGHILR$, $ABFGHJLS$	[78, 0, 0, 0, 48, 0, 0, 0, 1, 0, 0, 0]
20-8.1	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$, $ACEGHJKP$, $ADFGIJKQ$, $BCEGHILR$, $ABFGHJLS$, $BDEFIJLT$	[130, 0, 0, 0, 120, 0, 0, 0, 5, 0, 0, 0, 0]
21-9.1	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$, $ACEGHJKP$, $ADFGIJKQ$, $BCEGHILR$, $ABFGHJLS$, $BDEFIJLT$, $DEFGHKLU$	[210, 0, 0, 0, 280, 0, 0, 0, 21, 0, 0, 0, 0, 0]
22-10.1	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$, $ACEGHJKP$, $ADFGIJKQ$, $BCEGHILR$, $ABFGHJLS$, $BDEFIJLT$, $DEFGHKLU$, $ACDEIKLV$	[330, 0, 0, 0, 616, 0, 0, 0, 77, 0, 0, 0, 0, 0, 0]
23-11.1	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$, $ACEGHJKP$, $ADFGIJKQ$, $BCEGHILR$, $ABFGHJLS$, $BDEFIJLT$, $DEFGHKLU$, $ACDEIKLV$, $BCDGJKLW$	[506, 0, 0, 0, 1288, 0, 0, 0, 253, 0, 0, 0, 0, 0, 0, 0]
24-12.1	$ABCDEFGM$, $ABCDHIJN$, $ABEFHIKO$, $ACEGHJKP$, $ADFGIJKQ$, $BCEGHILR$, $ABFGHJLS$, $BDEFIJLT$, $DEFGHKLU$, $ACDEIKLV$, $BCDGJKLW$, $CFHIJKLX$	[759, 0, 0, 0, 2576, 0, 0, 0, 759, 0, 0, 0, 0, 0, 0, 1]

APPENDIX B

LIST OF NON-ISOMORPHIC 2-LEVEL MINIMUM ABERRATION FRACTIONAL FACTORIAL SPLIT-PLOT DESIGNS

This appendix gives tables of minimum aberration designs of 32 (Table 14), 64 (Table 15) and 128 (Table 16). The whole plot factors are denoted by uppercase letters and the sub-plot factors by lowercase letters, in the tables. The defining words for each of the designs are given. Additionally, the word length patterns of the designs are given. Note that the first element of the word length pattern is a_R, where R is the resolution of the design.

Table 14. 32-run minimum aberration 2-level regular fractional factorial split-plot designs with resolution ≥ 3

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
6	1.5.0.1	$Abcdef$	[0, 0, 0, 1]
6	2.4.0.1	$ABcdef$	[0, 0, 0, 1]
6	3.3.0.1	$ABCdef$	[0, 0, 0, 1]
6	4.2.0.1	$ABCDef$	[0, 0, 0, 1]
7	1.6.0.2	$Abcf$, $Abdeg$	[0, 1, 2, 0, 0]
7	1.6.0.2	$bcdf$, $Abceg$	[0, 1, 2, 0, 0]
7	2.5.0.2	$ABcf$, $ABdeg$	[0, 1, 2, 0, 0]
7	2.5.0.2	$ABcf$, $Acdeg$	[0, 1, 2, 0, 0]
7	2.5.0.2	$Acdf$, $ABceg$	[0, 1, 2, 0, 0]
7	2.5.0.2	$cdef$, $ABcdg$	[0, 1, 2, 0, 0]
7	3.4.0.2	$ABdf$, $ABCeg$	[0, 1, 2, 0, 0]
7	3.4.0.2	$ABdf$, $ACdeg$	[0, 1, 2, 0, 0]
7	3.4.0.2	$Adef$, $ABCdg$	[0, 1, 2, 0, 0]
7	3.4.0.2	$ABCdf$, $ABCeg$	[0, 1, 2, 0, 0]
7	3.4.1.1	ABF, $Acdeg$	[1, 0, 1, 1, 0]
7	4.3.0.2	$ABef$, $ACDeg$	[0, 1, 2, 0, 0]
7	4.3.1.1	$ABCF$, $ABdeg$	[0, 1, 2, 0, 0]
7	5.2.1.1	$ABCF$, $ABDeg$	[0, 1, 2, 0, 0]
7	5.2.1.1	$ABCDF$, $ABeg$	[0, 1, 2, 0, 0]

Table 14 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
8	1.7.0.3	$Abcf$, $Abdg$, $Acdeh$	[0, 3, 4, 0, 0, 0]
8	1.7.0.3	$bcdf$, $bceg$, $Abdeh$	[0, 3, 4, 0, 0, 0]
8	2.6.0.3	$ABcf$, $ABdg$, $Acdeh$	[0, 3, 4, 0, 0, 0]
8	2.6.0.3	$ABcf$, $Acdg$, $ABdeh$	[0, 3, 4, 0, 0, 0]
8	2.6.0.3	$Acdf$, $Aceg$, $ABdeh$	[0, 3, 4, 0, 0, 0]
8	2.6.0.3	$cdef$, $ABcdg$, $ABceh$	[0, 3, 4, 0, 0, 0]
8	3.5.0.3	$ABdf$, $ACdg$, $ABCeh$	[0, 3, 4, 0, 0, 0]
8	3.5.0.3	$ABdf$, $ACdg$, $BCdeh$	[0, 3, 4, 0, 0, 0]
8	3.5.0.3	$ABdf$, $ABeg$, $ACdeh$	[0, 3, 4, 0, 0, 0]
8	3.5.0.3	$ABdf$, $Adeg$, $ABCeh$	[0, 3, 4, 0, 0, 0]
8	3.5.0.3	$Adef$, $ABCdg$, $ABCeh$	[0, 3, 4, 0, 0, 0]
8	3.5.1.2	ABF, $Acdg$, $Bceh$	[1, 2, 3, 1, 0, 0]
8	4.4.0.3	$ABef$, $ACeg$, $BCDeh$	[0, 3, 4, 0, 0, 0]
8	4.4.0.3	$ABef$, $ACDeg$, $BCDeh$	[0, 3, 4, 0, 0, 0]
8	4.4.1.2	$ABCF$, $ABdg$, $ACdeh$	[0, 3, 4, 0, 0, 0]
8	5.3.1.2	$ABCF$, $ABeg$, $ACDeh$	[0, 3, 4, 0, 0, 0]
8	5.3.1.2	$ABCDF$, $ABeg$, $ACeh$	[0, 3, 4, 0, 0, 0]
8	5.3.2.1	ABF, ACG, $BCdeh$	[2, 1, 2, 2, 0, 0]
8	6.2.2.1	$ABCF$, $ABDG$, $ACDeh$	[0, 3, 4, 0, 0, 0]
9	1.8.0.4	$Abcf$, $Abdg$, $Abeh$, $Acdei$	[0, 6, 8, 0, 0, 1, 0]
9	1.8.0.4	$bcdf$, $bceg$, $Abdeh$, $Acdei$	[0, 6, 8, 0, 0, 1, 0]
9	2.7.0.4	$ABcf$, $ABdg$, $ABeh$, $Acdei$	[0, 6, 8, 0, 0, 1, 0]
9	2.7.0.4	$ABcf$, $Acdg$, $Aceh$, $ABdei$	[0, 6, 8, 0, 0, 1, 0]
9	2.7.0.4	$Acdf$, $Aceg$, $ABdeh$, $Bcdei$	[0, 6, 8, 0, 0, 1, 0]
9	3.6.0.4	$ABdf$, $ACdg$, $Adeh$, $ABCei$	[0, 6, 8, 0, 0, 1, 0]
9	3.6.0.4	$ABdf$, $ABeg$, $ACdeh$, $BCdei$	[0, 6, 8, 0, 0, 1, 0]
9	3.6.0.4	$ABdf$, $Adeg$, $ABCeh$, $BCdei$	[0, 6, 8, 0, 0, 1, 0]
9	3.6.1.3	ABF, $Acdg$, $Aceh$, $Bdei$	[1, 5, 6, 2, 1, 0, 0]
9	4.5.0.4	$ABef$, $ACeg$, $ADeh$, $BCDei$	[0, 6, 8, 0, 0, 1, 0]
9	4.5.1.3	$ABCF$, $ABdg$, $ABeh$, $ACdei$	[0, 6, 8, 0, 0, 1, 0]
9	5.4.1.3	$ABCF$, $ABeg$, $ACDeh$, $BCDei$	[0, 6, 8, 0, 0, 1, 0]
9	5.4.1.3	$ABCDF$, $ABeg$, $ACeh$, $ADei$	[0, 6, 8, 0, 0, 1, 0]
9	5.4.2.2	ABF, ACG, $BCdh$, $Adei$	[2, 4, 6, 2, 0, 1, 0]
9	6.3.2.2	$ABCF$, $ABDG$, $ABeh$, $ACDei$	[0, 6, 8, 0, 0, 1, 0]
9	6.3.3.1	ABF, ACG, BCH, $ABCdei$	[4, 3, 3, 4, 0, 0, 1]
9	7.2.3.1	$ABCF$, $ABDG$, $ACDH$, $BCDei$	[0, 7, 7, 0, 0, 0, 1]
10	1.9.0.5	$Abcf$, $Abdg$, $Abeh$, $Acdei$, $bcdej$	[0, 10, 16, 0, 0, 5, 0, 0]
10	2.8.0.5	$ABcf$, $ABdg$, $ABeh$, $Acdei$, $Bcdej$	[0, 10, 16, 0, 0, 5, 0, 0]
10	2.8.0.5	$ABcf$, $Acdg$, $Aceh$, $ABdei$, $Bcdej$	[0, 10, 16, 0, 0, 5, 0, 0]
10	3.7.0.5	$ABdf$, $ACdg$, $Adeh$, $ABCei$, $BCdej$	[0, 10, 16, 0, 0, 5, 0, 0]
10	3.7.1.4	ABF, $Acdg$, $Aceh$, $Adei$, $Bcdej$	[1, 10, 11, 4, 3, 1, 1, 0]

Table 14 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
10	4.6.0.5	$ABef$, $ACeg$, $BDeh$, $CDei$, $ABCDej$	[0, 15, 0, 15, 0, 0, 0, 1]
10	4.6.1.4	$ABCF$, $ABdg$, $ABeh$, $ACdei$, $BCdej$	[0, 10, 16, 0, 0, 5, 0, 0]
10	5.5.1.4	$ABCDF$, $ABeg$, $ACeh$, $ADei$, $BCDej$	[0, 10, 16, 0, 0, 5, 0, 0]
10	5.5.2.3	ABF, ACG, $BCdh$, $BCei$, $Adej$	[2, 8, 12, 4, 2, 3, 0, 0]
10	6.4.2.3	$ABCF$, $ABDG$, $ABeh$, $ACDei$, $BCDej$	[0, 10, 16, 0, 0, 5, 0, 0]
10	6.4.3.2	ABF, ACG, BCH, $Adei$, $ABCdj$	[4, 8, 8, 4, 4, 3, 0, 0]
10	7.3.3.2	$ABCF$, $ABDG$, $ACDH$, $ABei$, $ACej$	[0, 16, 0, 12, 0, 3, 0, 0]
10	7.3.4.1	ABF, ACG, BCH, $ABCI$, $Adej$	[7, 8, 3, 4, 5, 3, 1, 0]
10	8.2.4.1	$ABCF$, $ABDG$, $ACDH$, $BCDI$, $ABej$	[0, 18, 0, 8, 0, 5, 0, 0]
11	1.10.0.6	$Abcf$, $Abdg$, $Acdh$, $Abei$, $Acej$, $Adek$	[0, 25, 0, 27, 0, 10, 0, 1, 0]
11	1.10.0.6	$Abcf$, $Abdg$, $Acdh$, $Abei$, $Acej$, $bdek$	[0, 25, 0, 27, 0, 10, 0, 1, 0]
11	2.9.0.6	$ABcf$, $ABdg$, $Acdh$, $ABei$, $Acej$, $Adek$	[0, 25, 0, 27, 0, 10, 0, 1, 0]
11	2.9.0.6	$ABcf$, $ABdg$, $Acdh$, $ABei$, $Acej$, $cdek$	[0, 25, 0, 27, 0, 10, 0, 1, 0]
11	2.9.0.6	$ABcf$, $ABdg$, $Acdh$, $Acei$, $Adej$, $ABcdek$	[0, 25, 0, 27, 0, 10, 0, 1, 0]
11	3.8.0.6	$ABdf$, $ACdg$, $BCdh$, $ABei$, $ACej$, $Adek$	[0, 25, 0, 27, 0, 10, 0, 1, 0]
11	3.8.0.6	$ABdf$, $ACdg$, $BCdh$, $ABei$, $Adej$, $Cdek$	[0, 25, 0, 27, 0, 10, 0, 1, 0]
11	3.8.0.6	$ABdf$, $ACdg$, $ABeh$, $ACei$, $Adej$, $ABCdek$	[0, 25, 0, 27, 0, 10, 0, 1, 0]
11	3.8.1.5	ABF, Acg, $Bcdh$, $Bcei$, $Adej$, $ABcdek$	[2, 14, 22, 8, 6, 9, 2, 0, 0]
11	4.7.0.6	$ABef$, $ACeg$, $BCeh$, $ADei$, $BDej$, $CDek$	[0, 25, 0, 27, 0, 10, 0, 1, 0]
11	4.7.1.5	$ABCF$, $ABdg$, $ACdh$, $ABei$, $ACej$, $Adek$	[0, 25, 0, 27, 0, 10, 0, 1, 0]
11	4.7.1.5	$ABCF$, $ABdg$, $ACdh$, $ABei$, $Adej$, $Cdek$	[0, 25, 0, 27, 0, 10, 0, 1, 0]
11	5.6.1.5	$ABCF$, $ABeg$, $ACeh$, $ADei$, $BDej$, $CDek$	[0, 25, 0, 27, 0, 10, 0, 1, 0]

Table 14 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
11	5.6.2.4	ABF, ACG, $BCdh$, $BCei$, $Adej$, $ABCdek$	[2, 14, 22, 8, 6, 9, 2, 0, 0]
11	6.5.2.4	$ABCF$, $ABDG$, $ABeh$, $ACei$, $ADej$, $CDek$	[0, 25, 0, 27, 0, 10, 0, 1, 0]
11	6.5.3.3	ABF, ACG, BCH, $Adei$, $ABCdj$, $ABCek$	[4, 14, 16, 8, 12, 9, 0, 0, 0]
11	7.4.3.3	$ABCF$, $ABDG$, $ACDH$, $ABei$, $ACej$, $ADek$	[0, 25, 0, 27, 0, 10, 0, 1, 0]
11	7.4.4.2	ABF, ACG, BCH, $ABCI$, Adj, $Bdek$	[8, 12, 10, 12, 12, 7, 2, 0, 0]
11	8.3.4.2	$ABCF$, $ABDG$, $ACDH$, $BCDI$, $ABej$, $ACek$	[0, 26, 0, 24, 0, 13, 0, 0, 0]
11	9.2.5.1	ABF, ACG, ADH, $BCDI$, $ABCDJ$, $BCek$	[4, 18, 12, 8, 12, 5, 4, 0, 0]
12	1.11.0.7	$Abcf$, $Abdg$, $Acdh$, $bcdi$, $Abej$, $Acek$, $Adel$	[0, 38, 0, 52, 0, 33, 0, 4, 0, 0]
12	1.11.0.7	$Abcf$, $Abdg$, $Acdh$, $Abei$, $Acej$, $bdek$, $cdel$	[0, 38, 0, 52, 0, 33, 0, 4, 0, 0]
12	2.10.0.7	$ABcf$, $ABdg$, $Acdh$, $Bcdi$, $ABej$, $Acek$, $Adel$	[0, 38, 0, 52, 0, 33, 0, 4, 0, 0]
12	2.10.0.7	$ABcf$, $ABdg$, $Acdh$, $Bcdi$, $Acej$, $Adek$, $ABcdel$	[0, 38, 0, 52, 0, 33, 0, 4, 0, 0]
12	2.10.0.7	$ABcf$, $ABdg$, $Acdh$, $ABei$, $Acej$, $Adek$, $cdel$	[0, 38, 0, 52, 0, 33, 0, 4, 0, 0]
12	2.10.0.7	$ABcf$, $ABdg$, $Acdh$, $ABei$, $Acej$, $cdek$, $ABcdel$	[0, 38, 0, 52, 0, 33, 0, 4, 0, 0]
12	3.9.0.7	$ABdf$, $ACdg$, $BCdh$, $ABei$, $ACej$, $BCek$, $Adel$	[0, 38, 0, 52, 0, 33, 0, 4, 0, 0]
12	3.9.0.7	$ABdf$, $ACdg$, $BCdh$, $ABei$, $ACej$, $Adek$, $Bdel$	[0, 38, 0, 52, 0, 33, 0, 4, 0, 0]
12	3.9.0.7	$ABdf$, $ACdg$, $ABeh$, $ACei$, $Adej$, $Bdek$, $ABCdel$	[0, 38, 0, 52, 0, 33, 0, 4, 0, 0]
12	3.9.1.6	ABF, Acg, Adh, $Bcdi$, $Bcej$, $Bdek$, $Acdel$	[3, 25, 23, 27, 25, 10, 13, 1, 0, 0]
12	4.8.0.7	$ABef$, $ACeg$, $BCeh$, $ADei$, $BDej$, $CDek$, $ABCDel$	[0, 38, 0, 52, 0, 33, 0, 4, 0, 0]
12	4.8.1.6	$ABCF$, $ABdg$, $ACdh$, $BCdi$, $ABej$, $ACek$, $Adel$	[0, 38, 0, 52, 0, 33, 0, 4, 0, 0]
12	4.8.1.6	$ABCF$, $ABdg$, $ACdh$, $BCdi$, $ABej$, $Adek$, $Cdel$	[0, 38, 0, 52, 0, 33, 0, 4, 0, 0]
12	4.8.1.6	$ABCF$, $ABdg$, $ACdh$, $ABei$, $ACej$, $Adek$, $Bdel$	[0, 38, 0, 52, 0, 33, 0, 4, 0, 0]

Table 14 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
12	5.7.1.6	$ABCF$, $ABeg$, $ACeh$, $BCei$, $ADej$, $BDek$, $CDel$	[0, 38, 0, 52, 0, 33, 0, 4, 0, 0]
12	5.7.2.5	ABF, ACG, Adh, $BCdi$, $BCej$, $Bdek$, $ACdel$	[3, 25, 23, 27, 25, 10, 13, 1, 0, 0]
12	6.6.2.5	$ABCF$, $ABDG$, $ABeh$, $ACei$, $BCej$, $ADek$, $CDel$	[0, 38, 0, 52, 0, 33, 0, 4, 0, 0]
12	6.6.3.4	ABF, ACG, BCH, $Adei$, $ABCdj$, $ABCek$, $BCdel$	[4, 23, 28, 16, 28, 23, 4, 0, 0, 1]
12	7.5.3.4	$ABCF$, $ABDG$, $ACDH$, $ABei$, $ACej$, $BCek$, $ADel$	[0, 38, 0, 52, 0, 33, 0, 4, 0, 0]
12	7.5.4.3	ABF, ACG, BCH, $ABCI$, Adj, Bek, $Cdel$	[9, 17, 21, 27, 27, 18, 7, 1, 0, 0]
12	8.4.4.3	$ABCF$, $ABDG$, $ACDH$, $BCDI$, $ABej$, $ACek$, $ADel$	[0, 38, 0, 52, 0, 33, 0, 4, 0, 0]
12	9.3.5.2	ABF, ACG, ADH, $BCDI$, $ABCDJ$, $BCek$, $BDel$	[4, 26, 20, 24, 28, 13, 12, 0, 0, 0]
12	10.2.6.1	ABF, ACG, BCH, ADI, $BCDJ$, $ABCDK$, $BDel$	[8, 22, 24, 20, 24, 17, 8, 4, 0, 0]
13	1.12.0.8	$Abcf$, $Abdg$, $Acdh$, $bcdi$, $Abej$, $Acek$, $bcel$, $Adem$	[0, 55, 0, 96, 0, 87, 0, 16, 0, 1, 0]
13	1.12.0.8	$Abcf$, $Abdg$, $Acdh$, $Abei$, $Acej$, $bdek$, $cdel$, $Abcdem$	[0, 55, 0, 96, 0, 87, 0, 16, 0, 1, 0]
13	2.11.0.8	$ABcf$, $ABdg$, $Acdh$, $Bcdi$, $ABej$, $Acek$, $Bcel$, $Adem$	[0, 55, 0, 96, 0, 87, 0, 16, 0, 1, 0]
13	2.11.0.8	$ABcf$, $ABdg$, $Acdh$, $Bcdi$, $ABej$, $Acek$, $Adel$, $cdem$	[0, 55, 0, 96, 0, 87, 0, 16, 0, 1, 0]
13	2.11.0.8	$ABcf$, $ABdg$, $Acdh$, $ABei$, $Acej$, $Adek$, $cdel$, $ABcdem$	[0, 55, 0, 96, 0, 87, 0, 16, 0, 1, 0]
13	3.10.0.8	$ABdf$, $ACdg$, $BCdh$, $ABei$, $ACej$, $BCek$, $Adel$, $Bdem$	[0, 55, 0, 96, 0, 87, 0, 16, 0, 1, 0]
13	3.10.0.8	$ABdf$, $ACdg$, $BCdh$, $ABei$, $ACej$, $Adek$, $Bdel$, $Cdem$	[0, 55, 0, 96, 0, 87, 0, 16, 0, 1, 0]
13	3.10.1.7	ABF, Acg, Adh, Aei, $Bcdj$, $Bcek$, $Bdel$, $cdem$	[4, 38, 32, 52, 56, 33, 32, 4, 4, 0, 0]
13	4.9.0.8	Aef, Beg, Ceh, Dei, $ABCej$, $ABDek$, $ACDel$, $BCDem$	[4, 38, 32, 52, 56, 33, 32, 4, 4, 0, 0]
13	4.9.1.7	$ABCF$, $ABdg$, $ACdh$, $BCdi$, $ABej$, $ACek$, $BCel$, $Adem$	[0, 55, 0, 96, 0, 87, 0, 16, 0, 1, 0]
13	4.9.1.7	$ABCF$, $ABdg$, $ACdh$, $BCdi$, $ABej$, $ACek$, $Adel$, $Bdem$	[0, 55, 0, 96, 0, 87, 0, 16, 0, 1, 0]
13	4.9.1.7	$ABCF$, $ABdg$, $ACdh$, $ABei$, $ACej$, $Adek$, $Bdel$, $Cdem$	[0, 55, 0, 96, 0, 87, 0, 16, 0, 1, 0]

Table 14 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
13	5.8.1.7	$ABCF$, $ABeg$, $ACeh$, $BCei$, $ADej$, $BDek$, $CDel$, $ABCDem$	[0, 55, 0, 96, 0, 87, 0, 16, 0, 1, 0]
13	5.8.2.6	ABF, ACG, Adh, Aei, $BCdj$, $BCek$, $Bdel$, $Cdem$	[4, 38, 32, 52, 56, 33, 32, 4, 4, 0, 0]
13	6.7.2.6	$ABCF$, $ABDG$, $ABeh$, $ACei$, $BCej$, $ADek$, $BDel$, $CDem$	[0, 55, 0, 96, 0, 87, 0, 16, 0, 1, 0]
13	6.7.3.5	ABF, ACG, BCH, Adi, $BCdj$, $Bdek$, $ABCel$, $ACdem$	[6, 31, 44, 40, 56, 47, 20, 8, 2, 1, 0]
13	7.6.3.5	$ABCF$, $ABDG$, $ACDH$, $ABei$, $ACej$, $BCek$, $ADel$, $BDem$	[0, 55, 0, 96, 0, 87, 0, 16, 0, 1, 0]
13	7.6.4.4	ABF, ACG, BCH, $ABCI$, Adj, Bek, $Cdel$, $ABCdem$	[10, 24, 39, 54, 54, 39, 24, 10, 0, 0, 1]
13	8.5.4.4	$ABCF$, $ABDG$, $ACDH$, $BCDI$, $ABej$, $ACek$, $BCel$, $ADem$	[0, 55, 0, 96, 0, 87, 0, 16, 0, 1, 0]
13	9.4.5.3	ABF, ACG, ADH, $BCDI$, $ABCDJ$, $BCek$, $BDel$, $ACDem$	[4, 38, 33, 52, 52, 33, 38, 4, 0, 0, 1]
13	10.3.6.2	ABF, ACG, BCH, ADI, $BCDJ$, $ABCDK$, $BDel$, $ABCem$	[8, 31, 40, 40, 56, 47, 24, 8, 0, 1, 0]
13	11.2.7.1	ABF, ACG, BCH, ADI, BDJ, $ACDK$, $BCDL$, $ABCDem$	[12, 30, 41, 44, 44, 41, 30, 12, 0, 0, 1]
14	1.13.0.9	$Abcf$, $Abdg$, $Acdh$, $bcdi$, $Abej$, $Acek$, $bcel$, $Adem$, $bden$	[0, 77, 0, 168, 0, 203, 0, 56, 0, 7, 0, 0]
14	2.12.0.9	$ABcf$, $ABdg$, $Acdh$, $Bcdi$, $ABej$, $Acek$, $Bcel$, $Adem$, $Bden$	[0, 77, 0, 168, 0, 203, 0, 56, 0, 7, 0, 0]
14	2.12.0.9	$ABcf$, $ABdg$, $Acdh$, $Bcdi$, $ABej$, $Acek$, $Bcel$, $Adem$, $cden$	[0, 77, 0, 168, 0, 203, 0, 56, 0, 7, 0, 0]
14	3.11.0.9	$ABdf$, $ACdg$, $BCdh$, $ABei$, $ACej$, $BCek$, $Adel$, $Bdem$, $Cden$	[0, 77, 0, 168, 0, 203, 0, 56, 0, 7, 0, 0]
14	3.11.1.8	ABF, Acg, Adh, Aei, $Bcdj$, $Bcek$, $Bdel$, $cdem$, $ABcdn$	[5, 55, 45, 96, 106, 87, 82, 16, 17, 1, 1, 0]
14	3.11.1.8	ABF, Acg, Adh, Aei, $Bcdj$, $Bcek$, $Bdel$, $cdem$, $Acden$	[5, 55, 45, 96, 106, 87, 82, 16, 17, 1, 1, 0]
14	4.10.0.9	Aef, Beg, Ceh, Dei, $ABCej$, $ABDek$, $ACDel$, $BCDem$, $ABCDen$	[8, 42, 64, 85, 112, 85, 64, 42, 8, 0, 0, 1]
14	4.10.1.8	$ABCF$, $ABdg$, $ACdh$, $BCdi$, $ABej$, $ACek$, $BCel$, $Adem$, $Bden$	[0, 77, 0, 168, 0, 203, 0, 56, 0, 7, 0, 0]
14	4.10.1.8	$ABCF$, $ABdg$, $ACdh$, $BCdi$, $ABej$, $ACek$, $Adel$, $Bdem$, $Cden$	[0, 77, 0, 168, 0, 203, 0, 56, 0, 7, 0, 0]
14	5.9.1.8	$ABCF$, Aeg, Beh, Cei, Dej, $ABCek$, $ABDel$, $ACDem$, $BCDen$	[5, 55, 45, 96, 106, 87, 82, 16, 17, 1, 1, 0]

Table 14 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
14	5.9.2.7	ABF, ACG, Adh, Aei, $BCdj$, $BCek$, $Bdel$, $Cdem$, $ABCdn$	[5, 55, 45, 96, 106, 87, 82, 16, 17, 1, 1, 0]
14	5.9.2.7	ABF, ACG, Adh, Aei, $BCdj$, $BCek$, $Bdel$, $Cdem$, $ABden$	[5, 55, 45, 96, 106, 87, 82, 16, 17, 1, 1, 0]
14	6.8.2.7	$ABCF$, $ABDG$, $ABeh$, $ACei$, $BCej$, $ADek$, $BDel$, $CDem$, $ABCDen$	[0, 77, 0, 168, 0, 203, 0, 56, 0, 7, 0, 0]
14	6.8.3.6	ABF, ACG, BCH, Adi, Aej, $BCdk$, $BCel$, $Bdem$, $ACden$	[8, 43, 64, 80, 112, 95, 64, 32, 8, 5, 0, 0]
14	7.7.3.6	$ABCF$, $ABDG$, $ACDH$, $ABei$, $ACej$, $BCek$, $ADel$, $BDem$, $CDen$	[0, 77, 0, 168, 0, 203, 0, 56, 0, 7, 0, 0]
14	7.7.4.5	ABF, ACG, BCH, $ABCI$, Adj, Bdk, Cel, $ABdem$, $ABCden$	[12, 35, 64, 88, 104, 103, 64, 24, 12, 5, 0, 0]
14	8.6.4.5	$ABCF$, $ABDG$, $ACDH$, $BCDI$, $ABej$, $ACek$, $BCel$, $ADem$, $BDen$	[0, 77, 0, 168, 0, 203, 0, 56, 0, 7, 0, 0]
14	9.5.5.4	ABF, ACG, ADH, $BCDI$, $ABCDJ$, Aek, $BCel$, $BDem$, $CDen$	[5, 55, 45, 96, 106, 87, 82, 16, 17, 1, 1, 0]
14	10.4.6.3	ABF, ACG, BCH, ADI, $BCDJ$, $ABCDK$, $BDel$, $ABCem$, $ACDen$	[8, 45, 64, 72, 112, 107, 64, 24, 8, 7, 0, 0]
14	11.3.7.2	ABF, ACG, BCH, ADI, BDJ, $ACDK$, $BCDL$, $CDem$, $ABCen$	[12, 41, 64, 72, 104, 115, 64, 24, 12, 3, 0, 0]
14	12.2.8.1	ABF, ACG, BCH, ADI, BDJ, $ACDK$, $BCDL$, $ABCDM$, $CDen$	[16, 45, 64, 72, 96, 107, 64, 24, 16, 7, 0, 0]
15	1.14.0.10	$Abcf$, $Abdg$, $Acdh$, $bcdi$, $Abej$, $Acek$, $bcel$, $Adem$, $bden$, $cdeo$	[0, 105, 0, 280, 0, 435, 0, 168, 0, 35, 0, 0, 0]
15	2.13.0.10	$ABcf$, $ABdg$, $Acdh$, $Bcdi$, $ABej$, $Acek$, $Bcel$, $Adem$, $Bden$, $cdeo$	[0, 105, 0, 280, 0, 435, 0, 168, 0, 35, 0, 0, 0]
15	3.12.0.10	$ABdf$, $ACdg$, $BCdh$, $ABei$, $ACej$, $BCek$, $Adel$, $Bdem$, $Cden$, $ABCdeo$	[0, 105, 0, 280, 0, 435, 0, 168, 0, 35, 0, 0, 0]
15	3.12.1.9	ABF, Acg, Adh, Aei, $Bcdj$, $Bcek$, $Bdel$, $cdem$, $ABcdn$, $ABceo$	[6, 77, 62, 168, 188, 203, 188, 56, 62, 7, 6, 0, 0]
15	4.11.0.10	Aef, Beg, Ceh, $ADei$, $BDej$, $CDek$, $ABCel$, $ABDem$, $ACDen$, $BCDeo$	[12, 49, 108, 144, 176, 219, 176, 80, 36, 19, 4, 0, 0]
15	4.11.1.9	$ABCF$, $ABdg$, $ACdh$, $BCdi$, $ABej$, $ACek$, $BCel$, $Adem$, $Bden$, $Cdeo$	[0, 105, 0, 280, 0, 435, 0, 168, 0, 35, 0, 0, 0]
15	5.10.1.9	$ABCF$, Aeg, Beh, Cei, Dej, $ADek$, $ABCel$, $ABDem$, $ACDen$, $BCDeo$	[10, 60, 90, 141, 212, 193, 164, 98, 34, 18, 2, 1, 0]

Table 14 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
15	5.10.2.8	ABF, ACG, Adh, Aei, $BCdj$, $BCek$, $Bdel$, $Cdem$, $ABCdn$, $ABCeo$	[6, 77, 62, 168, 188, 203, 188, 56, 62, 7, 6, 0, 0]
15	5.10.2.8	ABF, ACG, Adh, Aei, $BCdj$, $BCek$, $Bdel$, $Cdem$, $ABCdn$, $ABdeo$	[6, 77, 62, 168, 188, 203, 188, 56, 62, 7, 6, 0, 0]
15	6.9.2.8	$ABCF$, $ABDG$, Aeh, Bei, Cej, Dek, $ABCel$, $ABDem$, $ACDen$, $BCDeo$	[6, 77, 62, 168, 188, 203, 188, 56, 62, 7, 6, 0, 0]
15	6.9.3.7	ABF, ACG, BCH, Adi, Aej, $BCdk$, $BCel$, $Bdem$, $Cden$, $ABdeo$	[10, 60, 90, 141, 212, 193, 164, 98, 34, 18, 2, 1, 0]
15	7.8.3.7	$ABCF$, $ABDG$, $ACDH$, $ABei$, $ACej$, $BCek$, $ADel$, $BDem$, $CDen$, $ABCDeo$	[0, 105, 0, 280, 0, 435, 0, 168, 0, 35, 0, 0, 0]
15	7.8.4.6	ABF, ACG, BCH, $ABCI$, Adj, Bdk, Ael, Bem, $Cden$, $ACdeo$	[14, 51, 92, 144, 212, 207, 144, 96, 46, 13, 4, 0, 0]
15	8.7.4.6	$ABCF$, $ABDG$, $ACDH$, $BCDI$, $ABej$, $ACek$, $BCel$, $ADem$, $BDen$, $CDeo$	[0, 105, 0, 280, 0, 435, 0, 168, 0, 35, 0, 0, 0]
15	9.6.5.5	ABF, ACG, ADH, $BCDI$, $ABCDJ$, Aek, $BCel$, $BDem$, $CDen$, $ABCeo$	[6, 77, 62, 168, 188, 203, 188, 56, 62, 7, 6, 0, 0]
15	10.5.6.4	ABF, ACG, BCH, ADI, $BCDJ$, $ABCDK$, Ael, $BCem$, $BDen$, $ACDeo$	[10, 61, 90, 136, 212, 203, 164, 88, 34, 23, 2, 0, 0]
15	11.4.7.3	ABF, ACG, BCH, ADI, BDJ, $ACDK$, $BCDL$, $CDem$, $ABCen$, $ABDeo$	[12, 57, 100, 120, 200, 243, 152, 72, 44, 19, 4, 0, 0]
15	12.3.8.2	ABF, ACG, BCH, ADI, BDJ, $ACDK$, $BCDL$, $ABCDM$, $CDen$, $ABCeo$	[16, 57, 96, 120, 192, 243, 160, 72, 48, 19, 0, 0, 0]
15	13.2.9.1	ABF, ACG, BCH, ADI, BDJ, CDK, $ABCL$, $ABDM$, $ACDN$, $BCDeo$	[22, 61, 94, 136, 188, 203, 156, 88, 46, 23, 6, 0, 0]
16	1.15.0.11	$Abcf$, $Abdg$, $Acdh$, $bcdi$, $Abej$, $Acek$, $bcel$, $Adem$, $bden$, $cdeo$, $Abcdep$	[0, 140, 0, 448, 0, 870, 0, 448, 0, 140, 0, 0, 0, 1]
16	2.14.0.11	$ABcf$, $ABdg$, $Acdh$, $Bcdi$, $ABej$, $Acek$, $Bcel$, $Adem$, $Bden$, $cdeo$, $ABcdep$	[0, 140, 0, 448, 0, 870, 0, 448, 0, 140, 0, 0, 0, 1]

Table 14 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
16	3.13.0.11	*Adf*, *Bdg*, *Cdh*, *dei*, *ABej*, *ACek*, *BCel*, *ABCdm*, *ABden*, *ACdeo*, *BCdep*	[7, 105, 84, 280, 315, 435, 400, 168, 189, 35, 28, 0, 1, 0]
16	3.13.1.10	*ABF*, *Acg*, *Adh*, *Aei*, *Bcdj*, *Bcek*, *Bdel*, *cdem*, *ABcdn*, *ABceo*, *ABdep*	[7, 105, 84, 280, 315, 435, 400, 168, 189, 35, 28, 0, 1, 0]
16	4.12.0.11	*Aef*, *Beg*, *Ceh*, *Dei*, *ABej*, *ACek*, *BCel*, *ABDem*, *ACDen*, *BCDeo*, *ABCDep*	[16, 65, 148, 236, 336, 419, 376, 240, 128, 59, 20, 4, 0, 0]
16	4.12.0.11	*Aef*, *Beg*, *Ceh*, *ABei*, *ADej*, *BDek*, *CDel*, *ABCem*, *ACDen*, *BCDeo*, *ABCDep*	[16, 65, 148, 236, 336, 419, 376, 240, 128, 59, 20, 4, 0, 0]
16	4.12.1.10	*ABCF*, *ABdg*, *ACdh*, *BCdi*, *ABej*, *ACek*, *BCel*, *Adem*, *Bden*, *Cdeo*, *ABCdep*	[0, 140, 0, 448, 0, 870, 0, 448, 0, 140, 0, 0, 0, 1]
16	5.11.1.10	*ABCDF*, *Aeg*, *Beh*, *Cei*, *ADej*, *BDek*, *CDel*, *ABCem*, *ABDen*, *ACDeo*, *BCDep*	[15, 65, 156, 232, 315, 435, 400, 216, 117, 75, 20, 0, 1, 0]
16	5.11.2.9	*ABF*, *ACG*, *Adh*, *Aei*, *BCdj*, *BCek*, *Bdel*, *Cdem*, *ABCdn*, *ABCeo*, *ABdep*	[7, 105, 84, 280, 315, 435, 400, 168, 189, 35, 28, 0, 1, 0]
16	6.10.2.9	*ABCF*, *ABDG*, *Aeh*, *Bei*, *Cej*, *Dek*, *ACel*, *ABCem*, *ABDen*, *ACDeo*, *BCDep*	[12, 83, 124, 230, 376, 391, 376, 244, 124, 69, 12, 6, 0, 0]
16	6.10.3.8	*ABF*, *ACG*, *BCH*, *Adi*, *Aej*, *BCdk*, *BCel*, *Bdem*, *Cden*, *ABCdo*, *ABdep*	[12, 83, 124, 230, 376, 391, 376, 244, 124, 69, 12, 6, 0, 0]
16	7.9.3.8	*ABCF*, *ABDG*, *ACDH*, *Aei*, *Bej*, *Cek*, *Del*, *ABCem*, *ABDen*, *ACDeo*, *BCDep*	[7, 105, 84, 280, 315, 435, 400, 168, 189, 35, 28, 0, 1, 0]
16	7.9.4.7	*ABF*, *ACG*, *BCH*, *ABCI*, *Adj*, *Bdk*, *Ael*, *Bem*, *Cden*, *ACdeo*, *BCdep*	[16, 70, 135, 231, 373, 405, 342, 262, 138, 52, 19, 3, 1, 0]
16	8.8.4.7	*ABCF*, *ABDG*, *ACDH*, *BCDI*, *ABej*, *ACek*, *BCel*, *ADem*, *BDen*, *CDeo*, *ABCDep*	[0, 140, 0, 448, 0, 870, 0, 448, 0, 140, 0, 0, 0, 1]
16	9.7.5.6	*ABF*, *ACG*, *ADH*, *BCDI*, *ABCDJ*, *Aek*, *BCel*, *BDem*, *CDen*, *ABCeo*, *ABDep*	[7, 105, 84, 280, 315, 435, 400, 168, 189, 35, 28, 0, 1, 0]
16	10.6.6.5	*ABF*, *ACG*, *BCH*, *ADI*, *BCDJ*, *ABCDK*, *Ael*, *BCem*, *BDen*, *CDeo*, *ABDep*	[12, 83, 124, 230, 376, 391, 376, 244, 124, 69, 12, 6, 0, 0]

Table 14 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
16	11.5.7.4	ABF, ACG, BCH, ADI, BDJ, $ACDK$, $BCDL$, Cem, Den, $ABeo$, $ABCDep$	[15, 73, 140, 216, 363, 435, 352, 232, 133, 67, 20, 0, 1, 0]
16	12.4.8.3	ABF, ACG, BCH, ADI, BDJ, $ACDK$, $BCDL$, $ABCDM$, $CDen$, $ABCeo$, $ABDep$	[16, 76, 144, 192, 352, 486, 352, 192, 144, 76, 16, 0, 0, 1]
16	13.3.9.2	ABF, ACG, BCH, ADI, BDJ, CDK, $ABCL$, $ABDM$, $ACDN$, Aeo, $BCDep$	[23, 73, 132, 216, 347, 435, 368, 232, 141, 67, 12, 0, 1, 0]
16	14.2.10.1	ABF, ACG, BCH, ADI, BDJ, CDK, $ABCL$, $ABDM$, $ACDN$, $BCDO$, $ABCDep$	[28, 84, 140, 224, 344, 406, 344, 224, 140, 84, 28, 0, 0, 1]
17	1.16.0.12	Abf, Acg, Adh, Aei, $bcdj$, $bcek$, $bdel$, $cdem$, $Abcdn$, $Abceo$, $Abdep$, $Acdeq$	[8, 140, 112, 448, 504, 870, 800, 448, 504, 140, 112, 0, 8, 1, 0]
17	1.16.0.12	Abf, bcg, bdh, bei, $Acdj$, $Acek$, $Adel$, $cdem$, $Abcdn$, $Abceo$, $Abdep$, $bcdeq$	[8, 140, 112, 448, 504, 870, 800, 448, 504, 140, 112, 0, 8, 1, 0]
17	2.15.0.12	Acf, Bcg, cdh, cei, $ABdj$, $ABek$, $Adel$, $Bdem$, $ABcdn$, $ABceo$, $Acdep$, $Bcdeq$	[8, 140, 112, 448, 504, 870, 800, 448, 504, 140, 112, 0, 8, 1, 0]
17	3.14.0.12	Adf, Bdg, Cdh, Aei, Bej, Cek, $Adel$, $Bdem$, $Cden$, $ABCdo$, $ABCep$, $ABCdeq$	[14, 112, 168, 364, 630, 750, 800, 568, 378, 224, 56, 28, 2, 1, 0]
17	3.14.0.12	Adf, Bdg, Cdh, dei, $ABdj$, $ABek$, $ACel$, $BCem$, $ABCdn$, $ABdeo$, $ACdep$, $BCdeq$	[14, 112, 168, 364, 630, 750, 800, 568, 378, 224, 56, 28, 2, 1, 0]
17	3.14.1.11	ABF, Acg, Adh, Aei, $Bcdj$, $Bcek$, $Bdel$, $cdem$, $ABcdn$, $ABceo$, $ABdep$, $Acdeq$	[8, 140, 112, 448, 504, 870, 800, 448, 504, 140, 112, 0, 8, 1, 0]
17	4.13.0.12	Aef, Beg, Ceh, Dei, $ABej$, $ACek$, $BCel$, $ADem$, $BDen$, $ACDeo$, $BCDep$, $ABCDeq$	[20, 86, 202, 366, 594, 778, 772, 604, 376, 190, 82, 22, 2, 1, 0]
17	4.13.1.11	$ABCF$, Adg, Bdh, Cdi, dej, $ABek$, $ACel$, $BCem$, $ABCdn$, $ABdeo$, $ACdep$, $BCdeq$	[8, 140, 112, 448, 504, 870, 800, 448, 504, 140, 112, 0, 8, 1, 0]
17	5.12.1.11	$ABCF$, Aeg, Beh, Cei, $ABej$, $ADek$, $BDel$, $CDem$, $ABCen$, $ACDeo$, $BCDep$, $ABCDeq$	[20, 84, 208, 368, 572, 790, 800, 576, 364, 212, 80, 16, 4, 1, 0]
17	5.12.1.11	$ABCDF$, Aeg, Beh, Cei, Dej, $ABek$, $ACel$, $BCem$, $ADen$, $BDeo$, $CDep$, $ABCDeq$	[20, 84, 208, 368, 572, 790, 800, 576, 364, 212, 80, 16, 4, 1, 0]

Table 14 Continued

n_1+n_2	$n_1.n_2.k_1.k_2$	defining words	word length pattern
17	5.12.2.10	ABF, ACG, Adh, Aei, $BCdj$, $BCek$, $Bdel$, $Cdem$, $ABCdn$, $ABCeo$, $ABdep$, $ACdeq$	[8, 140, 112, 448, 504, 870, 800, 448, 504, 140, 112, 0, 8, 1, 0]
17	6.11.2.10	$ABCF$, $ABDG$, Aeh, Bei, Cej, Dek, $ACel$, $ADem$, $ABCen$, $ABDeo$, $ACDep$, $BCDeq$	[18, 95, 192, 354, 626, 767, 752, 620, 374, 193, 80, 18, 6, 0, 0]
17	6.11.3.9	ABF, ACG, BCH, Adi, Aej, $BCdk$, $BCel$, $Bdem$, $Cden$, $ABCdo$, $ABCep$, $ABdeq$	[14, 112, 168, 364, 630, 750, 800, 568, 378, 224, 56, 28, 2, 1, 0]
17	7.10.3.9	$ABCF$, $ABDG$, $ACDH$, Aei, Bej, Cek, Del, $ABem$, $ABCen$, $ABDeo$, $ACDep$, $BCDeq$	[14, 112, 168, 364, 630, 750, 800, 568, 378, 224, 56, 28, 2, 1, 0]
17	7.10.4.8	ABF, ACG, BCH, $ABCI$, Adj, Bdk, Ael, Bem, $ABdn$, $Cdeo$, $ACdep$, $BCdeq$	[19, 95, 186, 354, 641, 767, 732, 620, 389, 193, 74, 18, 7, 0, 0]
17	8.9.4.8	$ABCF$, $ABDG$, $ACDH$, $BCDI$, Aej, Bek, Cel, Dem, $ABCen$, $ABDeo$, $ACDep$, $BCDeq$	[8, 140, 112, 448, 504, 870, 800, 448, 504, 140, 112, 0, 8, 1, 0]
17	9.8.5.7	ABF, ACG, ADH, $BCDI$, $ABCDJ$, Aek, $BCel$, $BDem$, $CDen$, $ABCeo$, $ABDep$, $ACDeq$	[8, 140, 112, 448, 504, 870, 800, 448, 504, 140, 112, 0, 8, 1, 0]
17	10.7.6.6	ABF, ACG, BCH, ADI, $BCDJ$, $ABCDK$, Ael, $BCem$, $BDen$, $CDeo$, $ABCep$, $ABDeq$	[14, 112, 168, 364, 630, 750, 800, 568, 378, 224, 56, 28, 2, 1, 0]
17	11.6.7.5	ABF, ACG, BCH, ADI, BDJ, $ACDK$, $BCDL$, Cem, Den, $ABCeo$, $ABDep$, $ABCDeq$	[18, 95, 193, 354, 620, 767, 767, 620, 354, 193, 95, 18, 0, 0, 1]
17	12.5.8.4	ABF, ACG, BCH, ADI, BDJ, $ACDK$, $BCDL$, $ABCDM$, Aen, $BCeo$, $BDep$, $ACDeq$	[20, 92, 200, 336, 604, 838, 752, 544, 396, 220, 72, 16, 4, 1, 0]
17	13.4.9.3	ABF, ACG, BCH, ADI, BDJ, CDK, $ABCL$, $ABDM$, $ACDN$, Aeo, $BCDep$, $ABCDeq$	[24, 92, 192, 336, 600, 838, 768, 544, 392, 220, 64, 16, 8, 1, 0]
17	14.3.10.2	ABF, ACG, BCH, ADI, BDJ, CDK, $ABCL$, $ABDM$, $ACDN$, $BCDO$, Aep, $BCDeq$	[30, 96, 184, 348, 598, 782, 768, 600, 394, 208, 72, 12, 2, 1, 0]
17	15.2.11.1	ABF, ACG, BCH, ADI, BDJ, CDK, $ABCL$, $ABDM$, $ACDN$, $BCDO$, $ABCDP$, Aeq	[36, 112, 196, 364, 624, 750, 680, 568, 420, 224, 84, 28, 8, 1, 0]

Table 14 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
18	1.17.0.13	Abf, Acg, bch, Adi, Aej, $bcdk$, $bcel$, $bdem$, $cden$, $Abcdo$, $Abcep$, $Abdeq$, $Acder$	[16, 148, 224, 560, 1008, 1374, 1600, 1248, 1008, 644, 224, 112, 16, 9, 0, 0]
18	1.17.0.13	Abf, Acg, bch, bdi, bej, $Acdk$, $Acel$, $Adem$, $cden$, $Abcdo$, $Abcep$, $Abdeq$, $bcder$	[16, 148, 224, 560, 1008, 1374, 1600, 1248, 1008, 644, 224, 112, 16, 9, 0, 0]
18	2.16.0.13	Acf, Bcg, Adh, Bdi, Aej, Bek, $ABcl$, $ABdm$, $ABen$, $cdeo$, $Acdep$, $Bcdeq$, $ABcder$	[16, 148, 224, 560, 1008, 1374, 1600, 1248, 1008, 644, 224, 112, 16, 9, 0, 0]
18	2.16.0.13	Acf, Bcg, Adh, Bdi, cej, dek, $Acdl$, $Bcdm$, $ABen$, $cdeo$, $ABcep$, $ABdeq$, $ABcder$	[16, 148, 224, 560, 1008, 1374, 1600, 1248, 1008, 644, 224, 112, 16, 9, 0, 0]
18	2.16.0.13	Acf, Bcg, cdh, cei, $ABcj$, $ABdk$, $ABel$, $Adem$, $Bden$, $ABcdo$, $ABcep$, $Acdeq$, $Bcder$	[16, 148, 224, 560, 1008, 1374, 1600, 1248, 1008, 644, 224, 112, 16, 9, 0, 0]
18	3.15.0.13	Adf, Bdg, Cdh, Aei, Bej, Cek, $ABdl$, $ACdm$, $BCdn$, $ABeo$, $ACep$, $BCeq$, $ABCdr$	[21, 126, 259, 532, 1029, 1380, 1515, 1368, 967, 602, 273, 84, 31, 3, 1, 0]
18	3.15.0.13	Adf, Bdg, Cdh, Aei, Bej, Cek, $ABdl$, $Adem$, $Bden$, $Cdeo$, $ABCdp$, $ABCeq$, $ABCder$	[21, 126, 259, 532, 1029, 1380, 1515, 1368, 967, 602, 273, 84, 31, 3, 1, 0]
18	3.15.0.13	Adf, Bdg, Cdh, Aei, Bej, Cek, $Adel$, $Bdem$, $Cden$, $ABCdo$, $ABCep$, $ABdeq$, $ABCder$	[21, 126, 259, 532, 1029, 1380, 1515, 1368, 967, 602, 273, 84, 31, 3, 1, 0]
18	3.15.0.13	Adf, Bdg, Cdh, dei, $ABdj$, $ACdk$, $ABel$, $ACem$, $BCen$, $ABCdo$, $ABdep$, $ACdeq$, $BCder$	[21, 126, 259, 532, 1029, 1380, 1515, 1368, 967, 602, 273, 84, 31, 3, 1, 0]
18	3.15.1.12	ABF, Acg, Bch, Adi, Aej, $Bcdk$, $Bcel$, $Bdem$, $cden$, $ABcdo$, $ABcep$, $ABdeq$, $Acder$	[16, 148, 224, 560, 1008, 1374, 1600, 1248, 1008, 644, 224, 112, 16, 9, 0, 0]

Table 14 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
18	4.14.0.13	Aef, Beg, Ceh, Dei, $ABej$, $ACek$, $BDel$, $CDem$, $ABCen$, $ABDeo$, $ACDep$, $BCDeq$, $ABCDer$	[24, 113, 272, 547, 1000, 1387, 1504, 1387, 1000, 547, 272, 113, 24, 0, 0, 1]
18	4.14.1.12	$ABCF$, Adg, Bdh, Cdi, Aej, Bek, Cel, $Adem$, $Bden$, $Cdeo$, $ABCdp$, $ABCeq$, $ABCder$	[16, 148, 224, 560, 1008, 1374, 1600, 1248, 1008, 644, 224, 112, 16, 9, 0, 0]
18	4.14.1.12	$ABCF$, Adg, Bdh, Cdi, dej, $ABdk$, $ABel$, $ACem$, $BCen$, $ABCdo$, $ABdep$, $ACdeq$, $BCder$	[16, 148, 224, 560, 1008, 1374, 1600, 1248, 1008, 644, 224, 112, 16, 9, 0, 0]
18	5.13.1.12	$ABCF$, Aeg, Beh, Cei, Dej, $ABek$, $ACel$, $BCem$, $ADen$, $ABDeo$, $ACDep$, $BCDeq$, $ABCDer$	[25, 108, 279, 556, 965, 1402, 1555, 1336, 971, 596, 277, 92, 23, 5, 1, 0]
18	5.13.1.12	$ABCDF$, Aeg, Beh, Cei, Dej, $ABek$, $ACel$, $BCem$, $ADen$, $BDeo$, $CDep$, $ABCeq$, $ABCDer$	[25, 108, 279, 556, 965, 1402, 1555, 1336, 971, 596, 277, 92, 23, 5, 1, 0]
18	5.13.2.11	ABF, ACG, Adh, Bdi, Aej, $BCdk$, $BCel$, $Bdem$, $Cden$, $ABCdo$, $ABCep$, $ABdeq$, $ACder$	[16, 148, 224, 560, 1008, 1374, 1600, 1248, 1008, 644, 224, 112, 16, 9, 0, 0]
18	6.12.2.11	$ABCF$, $ABDG$, Aeh, Bei, Cej, $ABek$, $ADel$, $BDem$, $CDen$, $ABCeo$, $ACDep$, $BCDeq$, $ABCDer$	[24, 108, 288, 552, 936, 1422, 1600, 1296, 936, 636, 288, 72, 24, 9, 0, 0]
18	6.12.3.10	ABF, ACG, BCH, Adi, Aej, $BCdk$, $BCel$, $Bdem$, $Cden$, $ABCdo$, $ABCep$, $ABdeq$, $ACder$	[16, 148, 224, 560, 1008, 1374, 1600, 1248, 1008, 644, 224, 112, 16, 9, 0, 0]
18	7.11.3.10	$ABCF$, $ABDG$, $ACDH$, Aei, Bej, Cek, Del, $ABem$, $ACen$, $ABCeo$, $ABDep$, $ACDeq$, $BCDer$	[21, 126, 259, 532, 1029, 1380, 1515, 1368, 967, 602, 273, 84, 31, 3, 1, 0]
18	7.11.4.9	ABF, ACG, BCH, $ABCI$, Adj, Bdk, Ael, Bem, $ABdn$, $ABeo$, $Cdep$, $ACdeq$, $BCder$	[22, 126, 252, 532, 1050, 1380, 1480, 1368, 1002, 602, 252, 84, 38, 3, 0, 0]

Table 14 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
18	8.10.4.9	$ABCF$, $ABDG$, $ACDH$, $BCDI$, Aej, Bek, Cel, Dem, $ABen$, $ABCeo$, $ABDep$, $ACDeq$, $BCDer$	[16, 148, 224, 560, 1008, 1374, 1600, 1248, 1008, 644, 224, 112, 16, 9, 0, 0]
18	9.9.5.8	ABF, ACG, ADH, $BCDI$, $ABCDJ$, Aek, Bel, $BCem$, $BDen$, $CDeo$, $ABCep$, $ABDeq$, $ACDer$	[16, 148, 224, 560, 1008, 1374, 1600, 1248, 1008, 644, 224, 112, 16, 9, 0, 0]
18	10.8.6.7	ABF, ACG, BCH, ADI, $BCDJ$, $ABCDK$, Ael, $BCem$, $BDen$, $CDeo$, $ABCep$, $ABDeq$, $ACDer$	[16, 148, 224, 560, 1008, 1374, 1600, 1248, 1008, 644, 224, 112, 16, 9, 0, 0]
18	11.7.7.6	ABF, ACG, BCH, ADI, BDJ, $ACDK$, $BCDL$, Cem, Den, $ABeo$, $CDep$, $ABCeq$, $ABDer$	[21, 126, 259, 532, 1029, 1380, 1515, 1368, 967, 602, 273, 84, 31, 3, 1, 0]
18	12.6.8.5	ABF, ACG, BCH, ADI, BDJ, $ACDK$, $BCDL$, $ABCDM$, Aen, $BCeo$, $BDep$, $CDeq$, $ACDer$	[24, 116, 272, 528, 1000, 1438, 1504, 1312, 1000, 612, 272, 80, 24, 9, 0, 0]
18	13.5.9.4	ABF, ACG, BCH, ADI, BDJ, CDK, $ABCL$, $ABDM$, $ACDN$, Aeo, Bep, $BCDeq$, $ABCDer$	[28, 112, 264, 536, 996, 1442, 1520, 1296, 996, 616, 264, 88, 28, 5, 0, 0]
18	14.4.10.3	ABF, ACG, BCH, ADI, BDJ, CDK, $ABCL$, $ABDM$, $ACDN$, $BCDO$, Aep, $BCDeq$, $ABCDer$	[32, 116, 256, 528, 992, 1438, 1536, 1312, 992, 612, 256, 80, 32, 9, 0, 0]
18	15.3.11.2	ABF, ACG, BCH, ADI, BDJ, CDK, $ABCL$, $ABDM$, $ACDN$, $BCDO$, $ABCDP$, Aeq, Ber	[38, 126, 252, 532, 1002, 1380, 1480, 1368, 1050, 602, 252, 84, 22, 3, 0, 0]
19	1.18.0.14	Abf, Acg, bch, Adi, bdj, Aek, $bcdl$, $bcem$, $bden$, $cdeo$, $Abcdp$, $Abceq$, $Abder$, $Acdes$	[24, 164, 344, 784, 1624, 2382, 2904, 2848, 2312, 1652, 840, 336, 136, 25, 8, 0, 0]
19	1.18.0.14	Abf, Acg, bch, Adi, bdj, bek, $Acdl$, $Acem$, $Aden$, $cdeo$, $Abcdp$, $Abceq$, $Abder$, $bcdes$	[24, 164, 344, 784, 1624, 2382, 2904, 2848, 2312, 1652, 840, 336, 136, 25, 8, 0, 0]

Table 14 Continued

n_1+n_2	$n_1.n_2.k_1.k_2$	defining words	word length pattern
19	2.17.0.14	Acf, Bcg, Adh, Bdi, cdj, Aek, Bel, $ABcm$, $ABdn$, $ABeo$, $cdep$, $Acdeq$, $Bcder$, $ABcdes$	[24, 164, 344, 784, 1624, 2382, 2904, 2848, 2312, 1652, 840, 336, 136, 25, 8, 0, 0]
19	2.17.0.14	Acf, Bcg, Adh, Bdi, cdj, cek, $ABdl$, $ABem$, $Aden$, $Bdeo$, $ABcdp$, $ABceq$, $Acder$, $Bcdes$	[24, 164, 344, 784, 1624, 2382, 2904, 2848, 2312, 1652, 840, 336, 136, 25, 8, 0, 0]
19	2.17.0.14	Acf, Bcg, Adh, Bdi, Aej, Bek, $Acdl$, $Bcdm$, $Acen$, $Bceo$, $Adep$, $Bdeq$, $Acder$, $Bcdes$	[24, 164, 344, 784, 1624, 2382, 2904, 2848, 2312, 1652, 840, 336, 136, 25, 8, 0, 0]
19	2.17.0.14	Acf, Bcg, Adh, Bdi, cej, dek, $Acdl$, $Bcdm$, $ABen$, $cdeo$, $ABcdp$, $ABceq$, $ABder$, $ABcdes$	[24, 164, 344, 784, 1624, 2382, 2904, 2848, 2312, 1652, 840, 336, 136, 25, 8, 0, 0]
19	3.16.0.14	Adf, Bdg, Cdh, Aei, Bej, Cek, $ABdl$, $ACdm$, $BCdn$, $ABeo$, $ACep$, $BCeq$, $ABCdr$, $ABCes$	[24, 164, 344, 784, 1624, 2382, 2904, 2848, 2312, 1652, 840, 336, 136, 25, 8, 0, 0]
19	3.16.1.13	ABF, Acg, Bch, Adi, Bdj, Aek, $Bcdl$, $Bcem$, $Bden$, $cdeo$, $ABcdp$, $ABceq$, $ABder$, $Acdes$	[24, 164, 344, 784, 1624, 2382, 2904, 2848, 2312, 1652, 840, 336, 136, 25, 8, 0, 0]
19	4.15.0.14	Aef, Beg, Ceh, Dei, $ABej$, $ACek$, $BCel$, $ADem$, $BDen$, $CDeo$, $ABCep$, $ABDeq$, $ACDer$, $BCDes$	[28, 147, 364, 791, 1596, 2409, 2860, 2883, 2356, 1569, 868, 357, 116, 34, 4, 1, 0]
19	4.15.1.13	$ABCF$, Adg, Bdh, Cdi, Aej, Bek, Cel, $ABdm$, $Aden$, $Bdeo$, $Cdep$, $ABCdq$, $ABCer$, $ABCdes$	[24, 164, 344, 784, 1624, 2382, 2904, 2848, 2312, 1652, 840, 336, 136, 25, 8, 0, 0]
19	4.15.1.13	$ABCF$, Adg, Bdh, Cdi, Aej, Bek, Cel, $Adem$, $Bden$, $Cdeo$, $ABCdp$, $ABCeq$, $ABder$, $ABCdes$	[24, 164, 344, 784, 1624, 2382, 2904, 2848, 2312, 1652, 840, 336, 136, 25, 8, 0, 0]
19	4.15.1.13	$ABCF$, Adg, Bdh, Cdi, dej, $ABdk$, $ACdl$, $ABem$, $ACen$, $BCeo$, $ABCdp$, $ABdeq$, $ACder$, $BCdes$	[24, 164, 344, 784, 1624, 2382, 2904, 2848, 2312, 1652, 840, 336, 136, 25, 8, 0, 0]

Table 14 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
19	5.14.1.13	$ABCF$, Aeg, Beh, Cei, Dej, $ABek$, $ACel$, $BCem$, $ADen$, $BDeo$, $ABDep$, $ACDeq$, $BCDer$, $ABCDes$	[30, 138, 372, 812, 1554, 2408, 2912, 2856, 2338, 1582, 868, 364, 110, 31, 8, 0, 0]
19	5.14.1.13	$ABCDF$, Aeg, Beh, Cei, Dej, $ABek$, $ACel$, $BCem$, $ADen$, $BDeo$, $CDep$, $ABCeq$, $ABDer$, $ABCDes$	[30, 138, 372, 812, 1554, 2408, 2912, 2856, 2338, 1582, 868, 364, 110, 31, 8, 0, 0]
19	5.14.2.12	ABF, ACG, Adh, Bdi, Cdj, Aek, $BCdl$, $BCem$, $Bden$, $Cdeo$, $ABCdp$, $ABCeq$, $ABder$, $ACdes$	[24, 164, 344, 784, 1624, 2382, 2904, 2848, 2312, 1652, 840, 336, 136, 25, 8, 0, 0]
19	5.14.2.12	ABF, ACG, Adh, Bdi, Aej, Bek, $BCdl$, $BCem$, $Bden$, $Cdeo$, $ABCdp$, $ABCeq$, $ABder$, $ACdes$	[24, 164, 344, 784, 1624, 2382, 2904, 2848, 2312, 1652, 840, 336, 136, 25, 8, 0, 0]
19	6.13.2.12	$ABCF$, $ABDG$, Aeh, Bei, Cej, Dek, $ABel$, $ACem$, $BCen$, $CDeo$, $ABDep$, $ACDeq$, $BCDer$, $ABCDes$	[30, 136, 378, 816, 1526, 2418, 2962, 2816, 2298, 1632, 878, 336, 114, 37, 6, 0, 0]
19	6.13.3.11	ABF, ACG, BCH, Adi, Bdj, Aek, $BCdl$, $BCem$, $Bden$, $Cdeo$, $ABCdp$, $ABCeq$, $ABder$, $ACdes$	[24, 164, 344, 784, 1624, 2382, 2904, 2848, 2312, 1652, 840, 336, 136, 25, 8, 0, 0]
19	7.12.3.11	$ABCF$, $ABDG$, $ACDH$, Aei, Bej, Cek, Del, $ABem$, $ACen$, $BCeo$, $ADep$, $BDeq$, $CDer$, $ABCDes$	[28, 147, 364, 791, 1596, 2409, 2860, 2883, 2356, 1569, 868, 357, 116, 34, 4, 1, 0]
19	7.12.4.10	ABF, ACG, BCH, $ABCI$, Adj, Bdk, Ael, Bem, $ABdn$, $ABeo$, $Cdep$, $ACdeq$, $BCder$, $ABCdes$	[25, 164, 336, 784, 1652, 2382, 2848, 2848, 2382, 1652, 784, 336, 164, 25, 0, 0, 1]
19	8.11.4.10	$ABCF$, $ABDG$, $ACDH$, $BCDI$, Aej, Bek, Cel, Dem, $ABen$, $ACeo$, $ABCep$, $ABDeq$, $ACDer$, $BCDes$	[24, 164, 344, 784, 1624, 2382, 2904, 2848, 2312, 1652, 840, 336, 136, 25, 8, 0, 0]
19	9.10.5.9	ABF, ACG, ADH, $BCDI$, $ABCDJ$, Aek, Bel, Cem, $BCen$, $BDeo$, $CDep$, $ABCeq$, $ABDer$, $ACDes$	[24, 164, 344, 784, 1624, 2382, 2904, 2848, 2312, 1652, 840, 336, 136, 25, 8, 0, 0]

Table 14 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
19	10.9.6.8	ABF, ACG, BCH, ADI, $BCDJ$, $ABCDK$, Ael, Bem, $BCen$, $BDeo$, $CDep$, $ABCeq$, $ABDer$, $ACDes$	[24, 164, 344, 784, 1624, 2382, 2904, 2848, 2312, 1652, 840, 336, 136, 25, 8, 0, 0]
19	11.8.7.7	ABF, ACG, BCH, ADI, BDJ, $ACDK$, $BCDL$, Cem, Den, $ABeo$, $CDep$, $ABCeq$, $ABDer$, $ABCDes$	[24, 164, 344, 784, 1624, 2382, 2904, 2848, 2312, 1652, 840, 336, 136, 25, 8, 0, 0]
19	12.7.8.6	ABF, ACG, BCH, ADI, BDJ, $ACDK$, $BCDL$, $ABCDM$, Aen, $BCeo$, $BDep$, $CDeq$, $ABCer$, $ABDes$	[28, 148, 364, 784, 1596, 2430, 2860, 2848, 2356, 1604, 868, 336, 116, 41, 4, 0, 0]
19	13.6.9.5	ABF, ACG, BCH, ADI, BDJ, CDK, $ABCL$, $ABDM$, $ACDN$, Aeo, Bep, $CDeq$, $BCDer$, $ABCDes$	[32, 140, 360, 800, 1584, 2438, 2872, 2816, 2368, 1612, 856, 352, 112, 33, 8, 0, 0]
19	14.5.10.4	ABF, ACG, BCH, ADI, BDJ, CDK, $ABCL$, $ABDM$, $ACDN$, $BCDO$, Aep, Beq, $ACDer$, $BCDes$	[36, 140, 348, 800, 1588, 2438, 2892, 2816, 2348, 1612, 852, 352, 124, 33, 4, 0, 0]
19	15.4.11.3	ABF, ACG, BCH, ADI, BDJ, CDK, $ABCL$, $ABDM$, $ACDN$, $BCDO$, $ABCDP$, Aeq, Ber, Ces	[41, 147, 337, 791, 1597, 2409, 2869, 2883, 2395, 1569, 819, 357, 127, 34, 7, 1, 0]
20	1.19.0.15	Abf, Acg, bch, Adi, bdj, Aek, bel, $Acdm$, $bcdn$, $Aceo$, $bcep$, $Adeq$, $bder$, $Acdes$, $bcdet$	[32, 188, 480, 1128, 2464, 4006, 5216, 5752, 5216, 3964, 2464, 1176, 480, 161, 32, 8, 0, 0]
20	1.19.0.15	Abf, Acg, bch, Adi, bdj, Aek, bel, $bcdm$, $bcen$, $bdeo$, $cdep$, $Abcdq$, $Abcer$, $Abdes$, $Acdet$	[32, 188, 480, 1128, 2464, 4006, 5216, 5752, 5216, 3964, 2464, 1176, 480, 161, 32, 8, 0, 0]
20	2.18.0.15	Acf, Bcg, Adh, Bdi, cdj, Aek, Bel, cem, $ABcn$, $ABdo$, $ABep$, $cdeq$, $Acder$, $Bcdes$, $ABcdet$	[32, 188, 480, 1128, 2464, 4006, 5216, 5752, 5216, 3964, 2464, 1176, 480, 161, 32, 8, 0, 0]
20	2.18.0.15	Acf, Bcg, Adh, Bdi, cdj, Aek, Bel, cem, $ABcn$, $Adeo$, $Bdep$, $cdeq$, $ABcdr$, $ABces$, $ABcdet$	[32, 188, 480, 1128, 2464, 4006, 5216, 5752, 5216, 3964, 2464, 1176, 480, 161, 32, 8, 0, 0]

Table 14 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
20	2.18.0.15	Acf, Bcg, Adh, Bdi, Aej, Bek, $Acdl$, $Bcdm$, $Acen$, $Bceo$, $Adep$, $Bdeq$, $ABcdr$, $Acdes$, $Bcdet$	[32, 188, 480, 1128, 2464, 4006, 5216, 5752, 5216, 3964, 2464, 1176, 480, 161, 32, 8, 0, 0]
20	2.18.0.15	Acf, Bcg, Adh, Bdi, Aej, Bek, $Acdl$, $Bcdm$, $Acen$, $Bceo$, $Adep$, $Bdeq$, $Acder$, $Bcdes$, $ABcdet$	[32, 188, 480, 1128, 2464, 4006, 5216, 5752, 5216, 3964, 2464, 1176, 480, 161, 32, 8, 0, 0]
20	3.17.0.15	Adf, Bdg, Cdh, Aei, Bej, Cek, del, $ABdm$, $ACdn$, $BCdo$, $ABep$, $ACeq$, $BCer$, $ABCds$, $ABCet$	[32, 188, 480, 1128, 2464, 4006, 5216, 5752, 5216, 3964, 2464, 1176, 480, 161, 32, 8, 0, 0]
20	3.17.1.14	ABF, Acg, Bch, Adi, Bdj, Aek, Bel, $Acdm$, $Bcdn$, $Aceo$, $Bcep$, $Adeq$, $Bder$, $Acdes$, $Bcdet$	[32, 188, 480, 1128, 2464, 4006, 5216, 5752, 5216, 3964, 2464, 1176, 480, 161, 32, 8, 0, 0]
20	4.16.0.15	Aef, Beg, Ceh, Dei, $ABej$, $ACek$, $BCel$, $ADem$, $BDen$, $CDeo$, $ABCep$, $ABDeq$, $ACDer$, $BCDes$, $ABCDet$	[32, 188, 480, 1128, 2464, 4006, 5216, 5752, 5216, 3964, 2464, 1176, 480, 161, 32, 8, 0, 0]
20	4.16.1.14	$ABCF$, Adg, Bdh, Cdi, Aej, Bek, Cel, dem, $ABdn$, $ACdo$, $BCdp$, $ABCeq$, $ABder$, $ACdes$, $BCdet$	[32, 188, 480, 1128, 2464, 4006, 5216, 5752, 5216, 3964, 2464, 1176, 480, 161, 32, 8, 0, 0]
20	4.16.1.14	$ABCF$, Adg, Bdh, Cdi, Aej, Bek, Cel, $ABdm$, $ACen$, $Adeo$, $Bdep$, $Cdeq$, $ABCdr$, $ABCes$, $ABCdet$	[32, 188, 480, 1128, 2464, 4006, 5216, 5752, 5216, 3964, 2464, 1176, 480, 161, 32, 8, 0, 0]
20	4.16.1.14	$ABCF$, Adg, Bdh, Cdi, Aej, Bek, Cel, $ABdm$, $Aden$, $Bdeo$, $Cdep$, $ABCdq$, $ABCer$, $ACdes$, $ABCdet$	[32, 188, 480, 1128, 2464, 4006, 5216, 5752, 5216, 3964, 2464, 1176, 480, 161, 32, 8, 0, 0]
20	5.15.1.14	$ABCDF$, Aeg, Beh, Cei, Dej, $ABek$, $ACel$, $BCem$, $ADen$, $BDeo$, $CDep$, $ABCeq$, $ABDer$, $ACDes$, $BCDet$	[35, 175, 491, 1155, 2415, 4005, 5255, 5743, 5225, 3925, 2465, 1225, 453, 150, 45, 5, 0, 0]
20	5.15.2.13	ABF, ACG, Adh, Bdi, Cdj, Aek, Bel, $BCdm$, $BCen$, $Bdeo$, $Cdep$, $ABCdq$, $ABCer$, $ABdes$, $ACdet$	[32, 188, 480, 1128, 2464, 4006, 5216, 5752, 5216, 3964, 2464, 1176, 480, 161, 32, 8, 0, 0]

Table 14 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
20	5.15.2.13	ABF, ACG, Adh, Bdi, Aej, Bek, $BCdl$, $BCem$, $Bden$, $Cdeo$, $ABCdp$, $ABCeq$, $ABder$, $ACdes$, $BCdet$	[32, 188, 480, 1128, 2464, 4006, 5216, 5752, 5216, 3964, 2464, 1176, 480, 161, 32, 8, 0, 0]
20	6.14.2.13	$ABCF$, $ABDG$, Aeh, Bei, Cej, Dek, $ABel$, $ACem$, $BCen$, $ADeo$, $CDep$, $ABDeq$, $ACDer$, $BCDes$, $ABCDet$	[36, 170, 496, 1170, 2380, 4004, 5320, 5698, 5180, 3990, 2464, 1190, 468, 155, 40, 6, 0, 0]
20	6.14.3.12	ABF, ACG, BCH, Adi, Bdj, Aek, Bel, $ACdm$, $BCdn$, $ACeo$, $BCep$, $Adeq$, $Bder$, $ACdes$, $BCdet$	[32, 188, 480, 1128, 2464, 4006, 5216, 5752, 5216, 3964, 2464, 1176, 480, 161, 32, 8, 0, 0]
20	7.13.3.12	$ABCF$, $ABDG$, $ACDH$, Aei, Bej, Cek, Del, $ABem$, $ACen$, $BCeo$, $ADep$, $BDeq$, $CDer$, $ABCes$, $ABCDet$	[35, 176, 490, 1148, 2422, 4026, 5234, 5708, 5260, 3960, 2430, 1204, 474, 157, 38, 4, 1, 0]
20	7.13.4.11	ABF, ACG, BCH, $ABCI$, Adj, Bdk, Cdl, Aem, Ben, $ABdo$, $ABep$, $Cdeq$, $ACder$, $BCdes$, $ABCdet$	[33, 188, 472, 1128, 2492, 4006, 5160, 5752, 5286, 3964, 2408, 1176, 508, 161, 24, 8, 1, 0]
20	8.12.4.11	$ABCF$, $ABDG$, $ACDH$, $BCDI$, Aej, Bek, Cel, Dem, $ABen$, $ACeo$, $BCep$, $ADeq$, $BDer$, $CDes$, $ABCDet$	[32, 188, 480, 1128, 2464, 4006, 5216, 5752, 5216, 3964, 2464, 1176, 480, 161, 32, 8, 0, 0]
20	9.11.5.10	ABF, ACG, ADH, $BCDI$, $ABCDJ$, Aek, Bel, Cem, Den, $BCeo$, $BDep$, $CDeq$, $ABCer$, $ABDes$, $ACDet$	[32, 188, 480, 1128, 2464, 4006, 5216, 5752, 5216, 3964, 2464, 1176, 480, 161, 32, 8, 0, 0]
20	10.10.6.9	ABF, ACG, BCH, ADI, $BCDJ$, $ABCDK$, Ael, Bem, Den, $BCeo$, $BDep$, $CDeq$, $ABCer$, $ABDes$, $ACDet$	[32, 188, 480, 1128, 2464, 4006, 5216, 5752, 5216, 3964, 2464, 1176, 480, 161, 32, 8, 0, 0]
20	11.9.7.8	ABF, ACG, BCH, ADI, BDJ, $ACDK$, $BCDL$, Aem, Ben, $ACeo$, $BCep$, $ADeq$, $BDer$, $ACDes$, $BCDet$	[32, 188, 480, 1128, 2464, 4006, 5216, 5752, 5216, 3964, 2464, 1176, 480, 161, 32, 8, 0, 0]
20	12.8.8.7	ABF, ACG, BCH, ADI, BDJ, $ACDK$, $BCDL$, $ABCDM$, Aen, $BCeo$, $BDep$, $CDeq$, $ABCer$, $ABDes$, $ACDet$	[32, 189, 480, 1120, 2464, 4034, 5216, 5696, 5216, 4034, 2464, 1120, 480, 189, 32, 0, 0, 1]

Table 14 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
20	13.7.9.6	ABF, ACG, BCH, ADI, BDJ, CDK, $ABCL$, $ABDM$, $ACDN$, Aeo, Bep, $ABeq$, $CDer$, $ACDes$, $BCDet$	[37, 176, 476, 1148, 2464, 4026, 5164, 5708, 5330, 3960, 2388, 1204, 488, 157, 36, 4, 1, 0]
20	14.6.10.5	ABF, ACG, BCH, ADI, BDJ, CDK, $ABCL$, $ABDM$, $ACDN$, $BCDO$, Aep, Beq, $ACDer$, $BCDes$, $ABCDet$	[40, 173, 472, 1152, 2440, 4050, 5240, 5632, 5240, 4050, 2440, 1152, 472, 173, 40, 0, 0, 1]
20	15.5.11.4	ABF, ACG, BCH, ADI, BDJ, CDK, $ABCL$, $ABDM$, $ACDN$, $BCDO$, $ABCDP$, Aeq, Ber, Ces, Det	[45, 175, 453, 1155, 2465, 4005, 5225, 5743, 5255, 3925, 2415, 1225, 491, 150, 35, 5, 0, 0]

Table 15. 64-run minimum aberration 2-level regular fractional factorial split-plot designs with resolution ≥ 3

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
7	1.6.0.1	$Abcdefg$	[0, 0, 0, 0, 1]
7	2.5.0.1	$ABcdefg$	[0, 0, 0, 0, 1]
7	3.4.0.1	$ABCdefg$	[0, 0, 0, 0, 1]
7	4.3.0.1	$ABCDefg$	[0, 0, 0, 0, 1]
7	5.2.0.1	$ABCDEfg$	[0, 0, 0, 0, 1]
8	1.7.0.2	$Abcdg$, $Abefh$	[0, 0, 2, 1, 0, 0]
8	1.7.0.2	$Abcdg$, $bcefh$	[0, 0, 2, 1, 0, 0]
8	2.6.0.2	$ABcdg$, $ABefh$	[0, 0, 2, 1, 0, 0]
8	2.6.0.2	$ABcdg$, $Acefh$	[0, 0, 2, 1, 0, 0]
8	2.6.0.2	$ABcdg$, $cdefh$	[0, 0, 2, 1, 0, 0]
8	2.6.0.2	$Acdeg$, $Bcdfh$	[0, 0, 2, 1, 0, 0]
8	3.5.0.2	$ABCdg$, $ABefh$	[0, 0, 2, 1, 0, 0]
8	3.5.0.2	$ABCdg$, $Adefh$	[0, 0, 2, 1, 0, 0]
8	3.5.0.2	$ABCdg$, $ABCefh$	[0, 0, 2, 1, 0, 0]
8	3.5.0.2	$ABdeg$, $ACdfh$	[0, 0, 2, 1, 0, 0]
8	3.5.0.2	$ABdeg$, $Cdefh$	[0, 0, 2, 1, 0, 0]
8	3.5.1.1	ABG, $Acdefh$	[1, 0, 0, 1, 1, 0]
8	4.4.0.2	$ABCeg$, $ABDfh$	[0, 0, 2, 1, 0, 0]
8	4.4.0.2	$ABCeg$, $ADefh$	[0, 0, 2, 1, 0, 0]
8	4.4.0.2	$ABCeg$, $ABCDfh$	[0, 0, 2, 1, 0, 0]
8	4.4.0.2	$ABefg$, $CDefh$	[0, 0, 2, 1, 0, 0]
8	4.4.1.1	$ABCG$, $ABdefh$	[0, 1, 0, 2, 0, 0]
8	5.3.0.2	$ABCfg$, $ADEfh$	[0, 0, 2, 1, 0, 0]
8	5.3.1.1	$ABCDG$, $ABefh$	[0, 0, 2, 1, 0, 0]
8	6.2.1.1	$ABCDG$, $ABEfh$	[0, 0, 2, 1, 0, 0]
8	6.2.1.1	$ABCDEG$, $ABCfh$	[0, 0, 2, 1, 0, 0]
9	1.8.0.3	$Abcg$, $Abdeh$, $Acdfi$	[0, 1, 4, 2, 0, 0, 0]
9	1.8.0.3	$bcdg$, $Abceh$, $Abdfi$	[0, 1, 4, 2, 0, 0, 0]
9	1.8.0.3	$bcdg$, $Abceh$, $bdefi$	[0, 1, 4, 2, 0, 0, 0]
9	2.7.0.3	$ABcg$, $ABdeh$, $Acdfi$	[0, 1, 4, 2, 0, 0, 0]
9	2.7.0.3	$ABcg$, $Acdeh$, $Bcdfi$	[0, 1, 4, 2, 0, 0, 0]
9	2.7.0.3	$Acdg$, $ABceh$, $ABdfi$	[0, 1, 4, 2, 0, 0, 0]
9	2.7.0.3	$Acdg$, $ABceh$, $Adefi$	[0, 1, 4, 2, 0, 0, 0]
9	2.7.0.3	$cdeg$, $ABcdh$, $Acefi$	[0, 1, 4, 2, 0, 0, 0]
9	2.7.0.3	$cdeg$, $ABcdh$, $ABcefi$	[0, 1, 4, 2, 0, 0, 0]
9	2.7.0.3	$cdeg$, $Acdfh$, $Bcefi$	[0, 1, 4, 2, 0, 0, 0]
9	3.6.0.3	$ABdg$, $ABCeh$, $ACdfi$	[0, 1, 4, 2, 0, 0, 0]
9	3.6.0.3	$ABdg$, $ABCeh$, $Adefi$	[0, 1, 4, 2, 0, 0, 0]

Table 15 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
9	3.6.0.3	$ABdg$, $ACdeh$, $BCdfi$	[0, 1, 4, 2, 0, 0, 0]
9	3.6.0.3	$ABdg$, $ACdeh$, $ABefi$	[0, 1, 4, 2, 0, 0, 0]
9	3.6.0.3	$ABdg$, $ACdeh$, $Bdefi$	[0, 1, 4, 2, 0, 0, 0]
9	3.6.0.3	$Adeg$, $ABCdh$, $ABefi$	[0, 1, 4, 2, 0, 0, 0]
9	3.6.0.3	$Adeg$, $ABCdh$, $ABCefi$	[0, 1, 4, 2, 0, 0, 0]
9	3.6.0.3	$Adeg$, $ABdfh$, $ACefi$	[0, 1, 4, 2, 0, 0, 0]
9	3.6.0.3	$defg$, $ABdeh$, $ACdfi$	[0, 1, 4, 2, 0, 0, 0]
9	3.6.0.3	$defg$, $ABdeh$, $ABCdfi$	[0, 1, 4, 2, 0, 0, 0]
9	3.6.0.3	$ABCdg$, $ABCeh$, $Adefi$	[0, 1, 4, 2, 0, 0, 0]
9	3.6.1.2	ABG, $Acdeh$, $Bcdfi$	[1, 0, 3, 3, 0, 0, 0]
9	4.5.0.3	$ABeg$, $ACDeh$, $ABCfi$	[0, 1, 4, 2, 0, 0, 0]
9	4.5.0.3	$ABeg$, $ACDeh$, $BCefi$	[0, 1, 4, 2, 0, 0, 0]
9	4.5.0.3	$ABeg$, $ACDeh$, $ABCDfi$	[0, 1, 4, 2, 0, 0, 0]
9	4.5.0.3	$ABeg$, $ACDeh$, $BCDefi$	[0, 1, 4, 2, 0, 0, 0]
9	4.5.0.3	$ABeg$, $ABCfh$, $ADefi$	[0, 1, 4, 2, 0, 0, 0]
9	4.5.0.3	$ABeg$, $ACefh$, $BDefi$	[0, 1, 4, 2, 0, 0, 0]
9	4.5.0.3	$Aefg$, $ABCeh$, $ABDfi$	[0, 1, 4, 2, 0, 0, 0]
9	4.5.0.3	$Aefg$, $ABCeh$, $ABCDfi$	[0, 1, 4, 2, 0, 0, 0]
9	4.5.0.3	$ABCeg$, $ABCfh$, $ADefi$	[0, 1, 4, 2, 0, 0, 0]
9	4.5.0.3	$ABefg$, $CDefh$, $ABCDei$	[0, 1, 4, 2, 0, 0, 0]
9	4.5.1.2	$ABCG$, $ABdeh$, $ACdfi$	[0, 1, 4, 2, 0, 0, 0]
9	5.4.0.3	$ABfg$, $ACDfh$, $BCEfi$	[0, 1, 4, 2, 0, 0, 0]
9	5.4.0.3	$ABfg$, $ACDfh$, $BCDEfi$	[0, 1, 4, 2, 0, 0, 0]
9	5.4.0.3	$ABCfg$, $ADEfh$, $BCDEfi$	[0, 1, 4, 2, 0, 0, 0]
9	5.4.1.2	$ABCG$, $ABDeh$, $ACDfi$	[0, 1, 4, 2, 0, 0, 0]
9	5.4.1.2	$ABCG$, $ABDeh$, $ACefi$	[0, 1, 4, 2, 0, 0, 0]
9	5.4.1.2	$ABCDG$, $ABeh$, $ACefi$	[0, 1, 4, 2, 0, 0, 0]
9	5.4.2.1	ABG, ACH, $BCdefi$	[2, 1, 0, 2, 2, 0, 0]
9	6.3.1.2	$ABCG$, $ABDfh$, $ACEfi$	[0, 1, 4, 2, 0, 0, 0]
9	6.3.1.2	$ABCDG$, $ABfh$, $ACEfi$	[0, 1, 4, 2, 0, 0, 0]
9	6.3.1.2	$ABCDEG$, $ABfh$, $ACDfi$	[0, 1, 4, 2, 0, 0, 0]
9	6.3.2.1	$ABCG$, $ABDH$, $ACDefi$	[0, 3, 0, 4, 0, 0, 0]
9	7.2.2.1	$ABCG$, $ABDEH$, $ACDfi$	[0, 1, 4, 2, 0, 0, 0]
10	1.9.0.4	$Abcg$, $defh$, $Abdei$, $Acdfj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	1.9.0.4	$bcdg$, $Abceh$, $Abcfi$, $bdefj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	2.8.0.4	$ABcg$, $defh$, $ABdei$, $Acdfj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	2.8.0.4	$ABcg$, $defh$, $Acdei$, $Bcdfj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	2.8.0.4	$Acdg$, $Befh$, $ABcei$, $ABdfj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	2.8.0.4	$Acdg$, $ABceh$, $ABcfi$, $Adefj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	2.8.0.4	$cdeg$, $Acdfh$, $Bcefi$, $ABdefj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	3.7.0.4	$ABdg$, $Cefh$, $ABCei$, $ACdfj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	3.7.0.4	$ABdg$, $Cefh$, $ACdei$, $BCdfj$	[0, 2, 8, 4, 0, 1, 0, 0]

Table 15 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
10	3.7.0.4	$ABdg$, $ABCeh$, $ABCfi$, $Adefj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	3.7.0.4	$ABdg$, $ACdeh$, $ACdfi$, $ABefj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	3.7.0.4	$ABdg$, $ACdeh$, $ACdfi$, $Bdefj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	3.7.0.4	$Adeg$, $ABCdh$, $ABdfi$, $ACefj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	3.7.0.4	$Adeg$, $ABdfh$, $ACefi$, $BCdefj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	3.7.1.3	ABG, $cdeh$, $Acdfi$, $Bcefj$	[1, 1, 6, 6, 1, 0, 0, 0]
10	4.6.0.4	$ABeg$, $CDfh$, $ACDei$, $ABCfj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	4.6.0.4	$ABeg$, $CDfh$, $ACDei$, $BCefj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	4.6.0.4	$ABeg$, $CDfh$, $ACefi$, $BDefj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	4.6.0.4	$ABeg$, $ACDeh$, $ABCfi$, $ADefj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	4.6.0.4	$ABeg$, $ACDeh$, $ACefi$, $BDefj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	4.6.0.4	$ABeg$, $ABCfh$, $ADefi$, $BCDefj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	4.6.0.4	$ABeg$, $ACefh$, $BDefi$, $ABCDfj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	4.6.0.4	$Aefg$, $ABCeh$, $ABDfi$, $BCDefj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	4.6.1.3	$ABCG$, $defh$, $ABdei$, $ACdfj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	5.5.0.4	$ABfg$, $ACDfh$, $ACEfi$, $BDEfj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	5.5.1.3	$ABCG$, $Defh$, $ABDei$, $ACDfj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	5.5.1.3	$ABCG$, $ABDeh$, $ABDfi$, $ACefj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	5.5.1.3	$ABCDG$, $ABeh$, $CDfi$, $ACefj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	5.5.2.2	ABG, ACH, $BCdei$, $Adefj$	[2, 1, 5, 6, 0, 0, 1, 0]
10	6.4.1.3	$ABCG$, $DEfh$, $ABDfi$, $ACEfj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	6.4.1.3	$ABCG$, $ABDfh$, $ACEfi$, $BCDEfj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	6.4.1.3	$ABCDG$, $ABfh$, $ACEfi$, $ADEfj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	6.4.1.3	$ABCDEG$, $ABfh$, $ACDfi$, $ACEfj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	6.4.2.2	$ABCG$, $ABDH$, $ACDei$, $ABefj$	[0, 3, 7, 4, 0, 0, 1, 0]
10	6.4.3.1	ABG, ACH, BCI, $ABCdefj$	[4, 3, 0, 3, 4, 0, 0, 1]
10	7.3.2.2	$ABCG$, $ABDEH$, $DEfi$, $ACDfj$	[0, 2, 8, 4, 0, 1, 0, 0]
10	7.3.3.1	$ABCG$, $ABDH$, $ACDI$, $BCDefj$	[0, 7, 0, 7, 0, 0, 0, 1]
10	8.2.3.1	$ABCG$, $ABDH$, $ACDEI$, $ABEfj$	[0, 3, 7, 4, 0, 0, 1, 0]
11	1.10.0.5	$Abcg$, $Abdh$, $Acdei$, $Acdfj$, $Abefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	1.10.0.5	$Abcg$, $defh$, $Abdei$, $Abdfj$, $Acefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	1.10.0.5	$bcdg$, $bceh$, $Abcfi$, $bdefj$, $Acdefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	2.9.0.5	$ABcg$, $ABdh$, $Acdei$, $Acdfj$, $ABefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	2.9.0.5	$ABcg$, $Acdh$, $ABdei$, $ABdfj$, $Acefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	2.9.0.5	$ABcg$, $defh$, $ABdei$, $ABdfj$, $Acefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]

Table 15 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
11	2.9.0.5	$ABcg$, $defh$, $ABdei$, $Acdfj$, $Acefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	2.9.0.5	$ABcg$, $defh$, $Acdei$, $Acdfj$, $Bcefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	2.9.0.5	$Acdg$, $Aceh$, $ABdei$, $ABcfj$, $Adefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	2.9.0.5	$Acdg$, $Aceh$, $ABcfi$, $Adefj$, $Bcdefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	2.9.0.5	$Acdg$, $ABceh$, $ABcfi$, $Adefj$, $Bcdefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	3.8.0.5	$ABdg$, $ACdh$, $ABCei$, $ABCfj$, $Adefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	3.8.0.5	$ABdg$, $ACdh$, $BCdei$, $BCdfj$, $Adefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	3.8.0.5	$ABdg$, $ABeh$, $ACdei$, $ABCfj$, $Adefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	3.8.0.5	$ABdg$, $ABeh$, $ABCfi$, $Adefj$, $BCdefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	3.8.0.5	$ABdg$, $Adeh$, $ABCei$, $ACdfj$, $ABefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	3.8.0.5	$ABdg$, $Adeh$, $ACdfi$, $ABefj$, $BCdefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	3.8.0.5	$ABdg$, $Cefh$, $ABCei$, $ABCfj$, $Adefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	3.8.0.5	$ABdg$, $Cefh$, $ABCei$, $ACdfj$, $Adefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	3.8.0.5	$ABdg$, $Cefh$, $ACdei$, $ACdfj$, $Bdefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	3.8.0.5	$ABdg$, $ABCeh$, $ABCfi$, $Adefj$, $BCdefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	3.8.0.5	$ABdg$, $ACdeh$, $ACdfi$, $ABefj$, $BCdefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	3.8.0.5	$ABdg$, $ACdeh$, $ACdfi$, $Bdefj$, $ABCefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	3.8.0.5	$Adeg$, $Adfh$, $ABCdi$, $ABefj$, $BCdefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	3.8.1.4	ABG, $cdeh$, $Acdfi$, $Bcefj$, $ABdefk$	[1, 2, 12, 12, 2, 1, 0, 0, 1]
11	4.7.0.5	$ABeg$, $ACeh$, $BCDei$, $BCefj$, $ADefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	4.7.0.5	$ABeg$, $ACeh$, $ABCfi$, $ADefj$, $BCDefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]

Table 15 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
11	4.7.0.5	$ABeg$, $ACeh$, $BCefi$, $ADefj$, $ABCDfk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	4.7.0.5	$ABeg$, $ABfh$, $ACefi$, $ADefj$, $BCDefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	4.7.0.5	$ABeg$, $CDfh$, $ACDei$, $BCDej$, $ABCfk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	4.7.0.5	$ABeg$, $CDfh$, $ACDei$, $ABCfj$, $BCefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	4.7.0.5	$ABeg$, $CDfh$, $ACDei$, $ACefj$, $BDefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	4.7.0.5	$ABeg$, $Aefh$, $ACDei$, $ABCfj$, $BCDefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	4.7.0.5	$ABeg$, $ACDeh$, $ABCfi$, $ADefj$, $BCDefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	4.7.0.5	$ABeg$, $ACDeh$, $ACefi$, $BDefj$, $ABCDfk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	4.7.1.4	$ABCG$, $ABdh$, $ACdei$, $ACdfj$, $ABefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	4.7.1.4	$ABCG$, $defh$, $ABdei$, $ABdfj$, $ACefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	5.6.0.5	$ABfg$, $ACfh$, $BCDfi$, $BCEfj$, $ADEfk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	5.6.1.4	$ABCG$, $ABeh$, $ACDei$, $ABDfj$, $ACefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	5.6.1.4	$ABCG$, $ABeh$, $ABDfi$, $ACefj$, $BCDefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	5.6.1.4	$ABCG$, $Defh$, $ABDei$, $ABDfj$, $ACefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	5.6.1.4	$ABCG$, $ABDeh$, $ABDfi$, $ACefj$, $BCDefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	5.6.1.4	$ABCDG$, $ABeh$, $ACei$, $ABCfj$, $ADefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	5.6.1.4	$ABCDG$, $ABeh$, $ABfi$, $ACefj$, $ADefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	5.6.2.3	ABG, ACH, $defi$, $BCdej$, $ABCdfk$	[2, 2, 10, 12, 2, 1, 2, 0, 0]
11	6.5.1.4	$ABCG$, $ABfh$, $ACDfi$, $ACEfj$, $BCDEfk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	6.5.1.4	$ABCDG$, $ABfh$, $ACfi$, $ADEfj$, $BCDEfk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	6.5.1.4	$ABCDEG$, $ABfh$, $ACfi$, $BCDfj$, $BCEfk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]

Table 15 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
11	6.5.2.3	$ABCG$, $ABDH$, $ACDei$, $ACDfj$, $ABefk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	6.5.3.2	ABG, ACH, BCI, $Adej$, $ABCdfk$	[4, 4, 6, 8, 4, 3, 2, 0, 0]
11	6.5.3.2	ABG, ACH, BCI, $defj$, $ABCdek$	[4, 4, 6, 8, 4, 3, 2, 0, 0]
11	7.4.2.3	$ABCG$, $ABDH$, $ACDfi$, $ABEfj$, $BCDEfk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	7.4.2.3	$ABCG$, $ABDEH$, $ABfi$, $ACDfj$, $ACEfk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	7.4.2.3	$ABCG$, $ABDEH$, $DEfi$, $ABDfj$, $ACEfk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	7.4.2.3	$ABCG$, $ABDEH$, $DEfi$, $ACDfj$, $ACEfk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	7.4.3.2	$ABCG$, $ABDH$, $ACDI$, $Aefj$, $BCDek$	[0, 8, 10, 4, 4, 3, 2, 0, 0]
11	7.4.4.1	ABG, ACH, BCI, $ABCJ$, $Adefk$	[7, 7, 1, 3, 5, 4, 3, 1, 0]
11	8.3.3.2	$ABCG$, $ABDH$, $ACDEI$, $ACDfj$, $ABEfk$	[0, 4, 14, 8, 0, 3, 2, 0, 0]
11	8.3.4.1	$ABCG$, $ABDH$, $ACDI$, $BCDJ$, $ABefk$	[0, 14, 4, 0, 8, 1, 4, 0, 0]
11	9.2.4.1	$ABCG$, $ABDH$, $ABEI$, $ACDEJ$, $BCDEfk$	[0, 6, 12, 8, 0, 1, 4, 0, 0]
12	1.11.0.6	$Abcg$, $Abdh$, $Acdei$, $Acdfj$, $Abefk$, $bcdefl$	[0, 6, 24, 16, 0, 9, 8, 0, 0, 0]
12	2.10.0.6	$ABcg$, $ABdh$, $Acdei$, $Acdfj$, $ABefk$, $Bcdefl$	[0, 6, 24, 16, 0, 9, 8, 0, 0, 0]
12	2.10.0.6	$ABcg$, $Acdh$, $ABdei$, $ABdfj$, $Acefk$, $Bcdefl$	[0, 6, 24, 16, 0, 9, 8, 0, 0, 0]
12	2.10.0.6	$Acdg$, $Aceh$, $ABdei$, $ABcfj$, $Adefk$, $Bcdefl$	[0, 6, 24, 16, 0, 9, 8, 0, 0, 0]
12	3.9.0.6	$ABdg$, $ACdh$, $ABCei$, $ABCfj$, $Adefk$, $BCdefl$	[0, 6, 24, 16, 0, 9, 8, 0, 0, 0]
12	3.9.0.6	$ABdg$, $ACdh$, $BCdei$, $BCdfj$, $Adefk$, $ABCefl$	[0, 6, 24, 16, 0, 9, 8, 0, 0, 0]
12	3.9.0.6	$ABdg$, $ABeh$, $ACdei$, $ABCfj$, $Adefk$, $BCdefl$	[0, 6, 24, 16, 0, 9, 8, 0, 0, 0]
12	3.9.0.6	$ABdg$, $Adeh$, $ABCei$, $ACdfj$, $ABefk$, $BCdefl$	[0, 6, 24, 16, 0, 9, 8, 0, 0, 0]
12	3.9.1.5	ABG, $Acdh$, $Acei$, $Bcfj$, $Adefk$, $ABcdefl$	[1, 6, 20, 16, 6, 9, 4, 0, 1, 0]
12	4.8.0.6	$ABeg$, $ACeh$, $BCDei$, $BCefj$, $ADefk$, $ABCDfl$	[0, 6, 24, 16, 0, 9, 8, 0, 0, 0]

Table 15 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
12	4.8.0.6	$ABeg$, $CDfh$, $ACDei$, $BCDej$, $ABCfk$, $ABDfl$	[0, 6, 24, 16, 0, 9, 8, 0, 0, 0]
12	4.8.0.6	$ABeg$, $CDfh$, $ACDei$, $ABCfj$, $BCefk$, $ADefl$	[0, 6, 24, 16, 0, 9, 8, 0, 0, 0]
12	4.8.1.5	$ABCG$, $ABdh$, $ACdei$, $ACdfj$, $ABefk$, $BCdefl$	[0, 6, 24, 16, 0, 9, 8, 0, 0, 0]
12	5.7.0.6	$ABfg$, $ACfh$, $DEfi$, $BCDfj$, $BCEfk$, $ABCDEfl$	[0, 8, 20, 14, 8, 7, 4, 2, 0, 0]
12	5.7.1.5	$ABCG$, $ABeh$, $ACDei$, $ABDfj$, $ACefk$, $BCDefl$	[0, 6, 24, 16, 0, 9, 8, 0, 0, 0]
12	5.7.1.5	$ABCDG$, $ABeh$, $ACei$, $ABCfj$, $ADefk$, $BCDefl$	[0, 6, 24, 16, 0, 9, 8, 0, 0, 0]
12	5.7.2.4	ABG, ACH, $defi$, $BCdej$, $BCdfk$, $ABCefl$	[2, 4, 18, 22, 6, 3, 6, 2, 0, 0]
12	6.6.1.5	$ABCDEG$, $ABfh$, $ACfi$, $BCDfj$, $BCEfk$, $ADEfl$	[0, 6, 24, 16, 0, 9, 8, 0, 0, 0]
12	6.6.2.4	$ABCG$, $ABDH$, $ACDei$, $ACDfj$, $ABefk$, $BCDefl$	[0, 6, 24, 16, 0, 9, 8, 0, 0, 0]
12	6.6.3.3	ABG, ACH, BCI, $Adej$, $Adfk$, $ABCefl$	[4, 6, 12, 16, 12, 9, 4, 0, 0, 0]
12	6.6.3.3	ABG, ACH, BCI, $Adej$, $Bdfk$, $Cefl$	[4, 6, 12, 16, 12, 9, 4, 0, 0, 0]
12	6.6.3.3	ABG, ACH, BCI, $defj$, $ABCdek$, $ABCdfl$	[4, 6, 12, 16, 12, 9, 4, 0, 0, 0]
12	7.5.2.4	$ABCG$, $ABDEH$, $ABfi$, $ACDfj$, $ACEfk$, $BCDEfl$	[0, 6, 24, 16, 0, 9, 8, 0, 0, 0]
12	7.5.3.3	$ABCG$, $ABDH$, $ACDI$, $Aefj$, $BCDek$, $BCDfl$	[0, 10, 20, 8, 8, 13, 4, 0, 0, 0]
12	7.5.4.2	ABG, ACH, BCI, $ABCJ$, $Adek$, $Bdfl$	[7, 9, 7, 11, 13, 10, 5, 1, 0, 0]
12	8.4.3.3	$ABCG$, $ABDH$, $ACDEI$, $ACDfj$, $ABEfk$, $BCDEfl$	[0, 6, 24, 16, 0, 9, 8, 0, 0, 0]
12	8.4.4.2	$ABCG$, $ABDH$, $ACDI$, $BCDJ$, $ABek$, $ACefl$	[0, 18, 8, 8, 16, 5, 8, 0, 0, 0]
12	9.3.4.2	$ABCG$, $ABDH$, $ABEI$, $ACDEJ$, $ACfk$, $ABDEfl$	[0, 10, 16, 16, 8, 5, 8, 0, 0, 0]
12	9.3.5.1	ABG, ACH, ADI, $BCDJ$, $ABCDK$, $BCefl$	[4, 14, 12, 4, 12, 9, 4, 4, 0, 0]
12	10.2.5.1	$ABCG$, $ABDH$, $ABEI$, $ACDEJ$, $BCDEK$, $ACfl$	[0, 12, 20, 6, 8, 11, 4, 2, 0, 0]
13	1.12.0.7	$Abcg$, $Abdh$, $Abei$, $Acfj$, $Acdek$, $Adefl$, $Abcdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]

Table 15 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
13	1.12.0.7	$Abcg$, $Abdh$, $Abei$, $cdfj$, $Acdek$, $Acefl$, $bdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	1.12.0.7	$Abcg$, $Abdh$, $Acei$, $Adej$, $Acdfk$, $Abefl$, $bcdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	1.12.0.7	$Abcg$, $Abdh$, $Acei$, $Adej$, $Acdfk$, $bcefl$, $Abdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	2.11.0.7	$ABcg$, $ABdh$, $ABei$, $Acfj$, $Acdek$, $Adefl$, $ABcdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	2.11.0.7	$ABcg$, $ABdh$, $ABei$, $cdfj$, $Acdek$, $Acefl$, $Bdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	2.11.0.7	$ABcg$, $ABdh$, $Acei$, $Adej$, $Bcfk$, $Bdefl$, $cdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	2.11.0.7	$ABcg$, $ABdh$, $Acei$, $Adej$, $cefk$, $Bcdfl$, $Bdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	2.11.0.7	$ABcg$, $ABdh$, $Acei$, $Adej$, $Acdfk$, $ABefl$, $Bcdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	2.11.0.7	$ABcg$, $ABdh$, $Acei$, $Adej$, $ABefk$, $cdefl$, $ABcdfm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	2.11.0.7	$ABcg$, $Acdh$, $Acei$, $Adfj$, $ABdek$, $ABefl$, $ABcdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	2.11.0.7	$ABcg$, $Acdh$, $Acei$, $defj$, $ABdek$, $ABdfl$, $Bcefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	2.11.0.7	$ABcg$, $Acdh$, $Bcei$, $Adej$, $Bdfk$, $Acefl$, $ABcdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	2.11.0.7	$ABcg$, $Acdh$, $Bcei$, $Adej$, $defk$, $ABdfl$, $ABefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	2.11.0.7	$ABcg$, $Acdh$, $Bcei$, $Adej$, $ABdfk$, $Acefl$, $Bcdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	2.11.0.7	$Acdg$, $Aceh$, $Bdei$, $Acfj$, $ABdfk$, $Bcefl$, $Adefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	2.11.0.7	$Acdg$, $Aceh$, $Adfi$, $Aefj$, $ABdek$, $ABcfl$, $Bcdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $ACdh$, $ABei$, $ACej$, $Bdfk$, $BCefl$, $Cdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $ACdh$, $ABei$, $ACej$, $ABCfk$, $Adefl$, $BCdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $ACdh$, $ABei$, $Cdej$, $ACfk$, $Bdefl$, $ABCdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $ACdh$, $ABei$, $Cdej$, $Bdfk$, $ACefl$, $ABCdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $ACdh$, $ABei$, $Cdej$, $Befk$, $ABCfl$, $ACefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]

Table 15 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
13	3.10.0.7	$ABdg$, $ACdh$, $ABei$, $Cdej$, $Cefk$, $BCdfl$, $Bdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $ACdh$, $ABei$, $Cdej$, $ABCfk$, $Adefl$, $BCdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $ACdh$, $ABei$, $Cdej$, $BCdfk$, $ACefl$, $ABdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $ACdh$, $ABei$, $Adfj$, $ABCfk$, $ACefl$, $ABCdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $ACdh$, $Adei$, $Aefj$, $ABCek$, $ABCfl$, $ABCdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $ABeh$, $Cdei$, $ABfj$, $ACdfk$, $BCefl$, $Adefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $ABeh$, $Cdei$, $Adfj$, $Befk$, $ABCfl$, $ABCdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $ABeh$, $ACfi$, $Adfj$, $defk$, $BCdel$, $ABCdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $ABeh$, $ACfi$, $Adfj$, $ACdek$, $BCdel$, $ABCdfm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $ABeh$, $Adfi$, $Aefj$, $ACdek$, $ABCfl$, $BCdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $ABeh$, $Cdfi$, $Cefj$, $ACdek$, $BCdel$, $ABdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $ACeh$, $Adei$, $Adfj$, $ABCfk$, $ABefl$, $ABCdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $ACeh$, $Adei$, $Cdfj$, $BCefk$, $Bdefl$, $ABCdem$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $ACeh$, $Adfi$, $Aefj$, $ABCfk$, $BCdfl$, $ACdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $Adeh$, $Adfi$, $Cefj$, $ABCek$, $BCdfl$, $ABefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $Adeh$, $Adfi$, $Cefj$, $ABCek$, $BCdfl$, $Bdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $Adeh$, $Bdfi$, $Aefj$, $ABCek$, $ACdfl$, $BCdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.0.7	$ABdg$, $Adeh$, $Cefi$, $ABCej$, $BCdek$, $ABCfl$, $Bdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	3.10.1.6	ABG, $Acdh$, $Acei$, $Bdej$, $Bcfk$, $Adefl$, $ABcdefm$	[1, 10, 32, 28, 14, 21, 16, 4, 1, 0, 0]
13	4.9.0.7	$ABeg$, $ACeh$, $ADei$, $ABfj$, $BCDek$, $ACDfl$, $ABCDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	4.9.0.7	$ABeg$, $ACeh$, $BDei$, $CDej$, $ABfk$, $ACDfl$, $BCDfm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]

Table 15 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
13	4.9.0.7	$ABeg$, $ACeh$, $BDei$, $CDej$, $ABCfk$, $ADefl$, $BCDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	4.9.0.7	$ABeg$, $ACeh$, $BDei$, $CDej$, $BCefk$, $ADefl$, $ABCDfm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	4.9.0.7	$ABeg$, $ACeh$, $BDei$, $ABfj$, $ACfk$, $BCDfl$, $CDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	4.9.0.7	$ABeg$, $ACeh$, $BDei$, $ABfj$, $Cefk$, $ACDfl$, $ABCDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	4.9.0.7	$ABeg$, $ACeh$, $ABfi$, $CDfj$, $Cefk$, $BCDel$, $BDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	4.9.0.7	$ABeg$, $ACeh$, $ABfi$, $Cefj$, $BCDek$, $ACDfl$, $ABDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	4.9.0.7	$ABeg$, $ACeh$, $ADfi$, $Aefj$, $BCDek$, $ABCfl$, $BCDfm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	4.9.0.7	$ABeg$, $ACeh$, $ADfi$, $Aefj$, $BCDek$, $BCDfl$, $BCefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	4.9.0.7	$ABeg$, $CDeh$, $ABfi$, $CDfj$, $ACefk$, $BCefl$, $ABDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	4.9.0.7	$ABeg$, $CDeh$, $ABfi$, $Cefj$, $ACDfk$, $BCDfl$, $ABDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	4.9.0.7	$ABeg$, $CDeh$, $ABfi$, $ACDfj$, $ACefk$, $BDefl$, $ABCDem$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	4.9.0.7	$ABeg$, $CDeh$, $ACfi$, $Aefj$, $Defk$, $BCDfl$, $ABCDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	4.9.0.7	$ABeg$, $ABfh$, $Cefi$, $Defj$, $ACDek$, $BCDfl$, $ABCDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	4.9.0.7	$ABeg$, $CDfh$, $Aefi$, $BCDej$, $ABCfk$, $ABDfl$, $BCefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	4.9.1.6	$ABCG$, $ABdh$, $ABei$, $ACfj$, $ACdek$, $Adefl$, $ABCdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	4.9.1.6	$ABCG$, $ABdh$, $ABei$, $Adfj$, $ACdek$, $ACefl$, $ABCdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	4.9.1.6	$ABCG$, $ABdh$, $ABei$, $defj$, $ACdek$, $ACdfl$, $BCefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	4.9.1.6	$ABCG$, $ABdh$, $ACei$, $Adej$, $Bdfk$, $BCefl$, $Cdefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	4.9.1.6	$ABCG$, $ABdh$, $ACei$, $Adej$, $defk$, $BCdfl$, $BCefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	5.8.0.7	$ABfg$, $ACfh$, $ADfi$, $BEfj$, $BCDfk$, $CDEfl$, $ABCDEfm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	5.8.1.6	$ABCG$, $ABeh$, $ADei$, $CDej$, $BCfk$, $ABDfl$, $ACDfm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]

Table 15 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
13	5.8.1.6	$ABCG$, $ABeh$, $ADei$, $CDej$, $ADfk$, $BCefl$, $ABCDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	5.8.1.6	$ABCG$, $ABeh$, $ADei$, $CDej$, $Befk$, $ACDfl$, $ABCDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	5.8.1.6	$ABCG$, $ABeh$, $ADei$, $ABfj$, $ACDfk$, $ACefl$, $ABCDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	5.8.1.6	$ABCG$, $ABeh$, $ADei$, $ACfj$, $Befk$, $BCDfl$, $ABCDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	5.8.1.6	$ABCG$, $ABeh$, $ABfi$, $Defj$, $ACDek$, $BCDfl$, $ACefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	5.8.1.6	$ABCG$, $ABeh$, $ACfi$, $Aefj$, $Defk$, $BCDel$, $BCDfm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	5.8.1.6	$ABCG$, $ABeh$, $ADfi$, $Aefj$, $ACDek$, $BCDel$, $ABDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	5.8.1.6	$ABCG$, $ABeh$, $Defi$, $ACDej$, $BCDek$, $ACDfl$, $BCefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	5.8.1.6	$ABCDG$, $ABeh$, $ACei$, $ADej$, $ABfk$, $ACDfl$, $ABCDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	5.8.1.6	$ABCDG$, $ABeh$, $ACei$, $ABfj$, $ACfk$, $ADefl$, $BCDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	5.8.1.6	$ABCDG$, $ABeh$, $ACei$, $ABfj$, $Cefk$, $ADefl$, $BCDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	5.8.2.5	ABG, ACH, $BCdi$, $Adej$, $Adfk$, $BCefl$, $ABCdefm$	[2, 9, 30, 30, 14, 21, 18, 2, 0, 1, 0]
13	6.7.1.6	$ABCDG$, $ABfh$, $ACfi$, $ADfj$, $BEfk$, $CDEfl$, $ABCDEfm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	6.7.1.6	$ABCDEG$, $ABfh$, $ACfi$, $BDfj$, $CDfk$, $BCEfl$, $ADEfm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	6.7.2.5	$ABCG$, $ABDH$, $ABei$, $ACfj$, $ACDek$, $ADefl$, $ABCDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	6.7.2.5	$ABCG$, $ABDH$, $ABei$, $Aefj$, $ACDek$, $ACDfl$, $ABCDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	6.7.2.5	$ABCG$, $ABDH$, $ACei$, $ADej$, $BCfk$, $BDefl$, $CDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	6.7.2.5	$ABCG$, $ABDH$, $ACei$, $ADej$, $Cefk$, $BCDfl$, $BDefm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	6.7.3.4	ABG, ACH, BCI, $Adej$, $Bdfk$, $Cefl$, $ABCdefm$	[4, 9, 21, 31, 29, 18, 7, 5, 3, 0, 0]
13	7.6.2.5	$ABCG$, $ADEH$, $ABfi$, $ADfj$, $CDfk$, $BCEfl$, $ABCDEfm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	7.6.2.5	$ABCG$, $ABDEH$, $ABfi$, $ADfj$, $CDfk$, $ACEfl$, $BCDEfm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]

Table 15 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
13	7.6.2.5	$ABCG$, $ABDEH$, $ACfi$, $DEfj$, $ABDfk$, $BCDfl$, $BCEfm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	7.6.3.4	$ABCG$, $ABDH$, $ACDI$, $Aefj$, $BCDek$, $BCDfl$, $ABCDefm$	[0, 14, 33, 16, 16, 33, 14, 0, 0, 0, 1]
13	7.6.4.3	ABG, ACH, BCI, $ABCJ$, $Adek$, $Bdfl$, $Cefm$	[7, 11, 15, 25, 29, 24, 13, 3, 0, 0, 0]
13	8.5.3.4	$ABCG$, $ABDH$, $ACEI$, $ABfj$, $ACDfk$, $ADEfl$, $ABCDEfm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	8.5.3.4	$ABCG$, $ABDH$, $ACDEI$, $ABfj$, $AEfk$, $ACDfl$, $ABCDEfm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	8.5.3.4	$ABCG$, $ABDH$, $ACDEI$, $ACfj$, $ADfk$, $ABEfl$, $BCDEfm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	8.5.4.3	$ABCG$, $ABDH$, $ACDI$, $BCDJ$, $ABek$, $ACfl$, $ADefm$	[0, 22, 16, 20, 32, 17, 16, 4, 0, 0, 0]
13	9.4.4.3	$ABCG$, $ABDH$, $ABEI$, $ACDEJ$, $ACfk$, $ADEfl$, $ABCDEfm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	9.4.4.3	$ABCG$, $ABDH$, $ACEI$, $ADEJ$, $BCfk$, $BDEfl$, $CDEfm$	[0, 14, 28, 24, 24, 17, 12, 8, 0, 0, 0]
13	9.4.5.2	ABG, ACH, ADI, $BCDJ$, $ABCDK$, $BCel$, $BDefm$	[4, 18, 20, 16, 28, 21, 12, 8, 0, 0, 0]
13	10.3.5.2	$ABCG$, $ABDH$, $ABEI$, $ACDEJ$, $BCDEK$, $ACfl$, $ADfm$	[0, 16, 28, 18, 24, 23, 12, 6, 0, 0, 0]
13	10.3.6.1	ABG, ACH, BCI, ADJ, $BCDK$, $ABCDL$, $BDefm$	[8, 18, 20, 16, 20, 21, 12, 8, 4, 0, 0]
13	11.2.6.1	$ABCG$, $ABDH$, $ACDI$, $ABEJ$, $ACEK$, $ADEL$, $BCDfm$	[0, 25, 13, 27, 25, 10, 23, 1, 3, 0, 0]

Table 16. 128-run minimum aberration 2-level regular fractional factorial split-plot designs with resolution ≥ 4

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
8	1.7.0.1	$Abcdefgh$	[0, 0, 0, 0, 1]
8	2.6.0.1	$ABcdefgh$	[0, 0, 0, 0, 1]
8	3.5.0.1	$ABCdefgh$	[0, 0, 0, 0, 1]
8	4.4.0.1	$ABCDefgh$	[0, 0, 0, 0, 1]
8	5.3.0.1	$ABCDEfgh$	[0, 0, 0, 0, 1]
8	6.2.0.1	$ABCDEFgh$	[0, 0, 0, 0, 1]
9	1.8.0.2	$Abcdeh$, $Abcfgi$	[0, 0, 3, 0, 0, 0]
9	2.7.0.2	$ABcdeh$, $ABcfgi$	[0, 0, 3, 0, 0, 0]
9	2.7.0.2	$ABcdeh$, $Acdfgi$	[0, 0, 3, 0, 0, 0]
9	3.6.0.2	$ABCdeh$, $ABCfgi$	[0, 0, 3, 0, 0, 0]
9	3.6.0.2	$ABCdeh$, $ABdfgi$	[0, 0, 3, 0, 0, 0]
9	3.6.0.2	$ABdefh$, $ACdegi$	[0, 0, 3, 0, 0, 0]
9	4.5.0.2	$ABCDeh$, $ABCfgi$	[0, 0, 3, 0, 0, 0]
9	4.5.0.2	$ABCDeh$, $ABefgi$	[0, 0, 3, 0, 0, 0]
9	4.5.0.2	$ABCefh$, $ABDegi$	[0, 0, 3, 0, 0, 0]
9	4.5.1.1	$ABCH$, $ABdefgi$	[1, 0, 0, 2, 0, 0]
9	5.4.0.2	$ABCDfh$, $ABCEgi$	[0, 0, 3, 0, 0, 0]
9	5.4.0.2	$ABCDfh$, $ABEfgi$	[0, 0, 3, 0, 0, 0]
9	5.4.1.1	$ABCDH$, $ABefgi$	[0, 1, 1, 1, 0, 0]
9	6.3.0.2	$ABCDgh$, $ABEFgi$	[0, 0, 3, 0, 0, 0]
9	6.3.1.1	$ABCDEH$, $ABCfgi$	[0, 0, 3, 0, 0, 0]
9	7.2.1.1	$ABCDEH$, $ABCFgi$	[0, 0, 3, 0, 0, 0]
10	1.9.0.3	$Abcdh$, $Abefi$, $Acegj$	[0, 3, 3, 1, 0, 0, 0]
10	1.9.0.3	$Abcdh$, $Abefi$, $bcegj$	[0, 3, 3, 1, 0, 0, 0]
10	1.9.0.3	$Abcdh$, $bcefi$, $bdegj$	[0, 3, 3, 1, 0, 0, 0]
10	2.8.0.3	$ABcdh$, $ABefi$, $Acegj$	[0, 3, 3, 1, 0, 0, 0]
10	2.8.0.3	$ABcdh$, $Acefi$, $Bcegj$	[0, 3, 3, 1, 0, 0, 0]
10	2.8.0.3	$ABcdh$, $Acefi$, $Adegj$	[0, 3, 3, 1, 0, 0, 0]
10	2.8.0.3	$ABcdh$, $Acefi$, $cdegj$	[0, 3, 3, 1, 0, 0, 0]
10	2.8.0.3	$ABcdh$, $cdefi$, $ABcegj$	[0, 3, 3, 1, 0, 0, 0]
10	2.8.0.3	$Acdeh$, $Bcdfi$, $Bcegj$	[0, 3, 3, 1, 0, 0, 0]
10	2.8.0.3	$Acdeh$, $Bcdfi$, $cefgj$	[0, 3, 3, 1, 0, 0, 0]
10	3.7.0.3	$ABCdh$, $ABefi$, $ACegj$	[0, 3, 3, 1, 0, 0, 0]
10	3.7.0.3	$ABCdh$, $ABefi$, $Adegj$	[0, 3, 3, 1, 0, 0, 0]
10	3.7.0.3	$ABCdh$, $Adefi$, $Bdegj$	[0, 3, 3, 1, 0, 0, 0]
10	3.7.0.3	$ABCdh$, $Adefi$, $ABCegj$	[0, 3, 3, 1, 0, 0, 0]
10	3.7.0.3	$ABCdh$, $Adefi$, $BCdegj$	[0, 3, 3, 1, 0, 0, 0]
10	3.7.0.3	$ABdeh$, $ACdfi$, $BCdgj$	[0, 3, 3, 1, 0, 0, 0]

Table 16 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
10	3.7.0.3	$ABdeh$, $ACdfi$, $ACegj$	[0, 3, 3, 1, 0, 0, 0]
10	3.7.0.3	$ABdeh$, $ACdfi$, $Cdegj$	[0, 3, 3, 1, 0, 0, 0]
10	3.7.0.3	$ABdeh$, $ACdfi$, $Aefgj$	[0, 3, 3, 1, 0, 0, 0]
10	3.7.0.3	$ABdeh$, $ACdfi$, $defgj$	[0, 3, 3, 1, 0, 0, 0]
10	3.7.0.3	$ABdeh$, $Cdefi$, $Adfgj$	[0, 3, 3, 1, 0, 0, 0]
10	3.7.0.3	$ABdeh$, $Cdefi$, $ABCdgj$	[0, 3, 3, 1, 0, 0, 0]
10	3.7.0.3	$ABdeh$, $Cdefi$, $ABdfgj$	[0, 3, 3, 1, 0, 0, 0]
10	3.7.0.3	$Adefh$, $Bdegi$, $Cdfgj$	[0, 3, 3, 1, 0, 0, 0]
10	4.6.0.3	$ABCeh$, $ABDfi$, $ACDgj$	[0, 3, 3, 1, 0, 0, 0]
10	4.6.0.3	$ABCeh$, $ABDfi$, $ADegj$	[0, 3, 3, 1, 0, 0, 0]
10	4.6.0.3	$ABCeh$, $ABDfi$, $Aefgj$	[0, 3, 3, 1, 0, 0, 0]
10	4.6.0.3	$ABCeh$, $ADefi$, $BDegj$	[0, 3, 3, 1, 0, 0, 0]
10	4.6.0.3	$ABCeh$, $ADefi$, $ABfgj$	[0, 3, 3, 1, 0, 0, 0]
10	4.6.0.3	$ABCeh$, $ADefi$, $Befgj$	[0, 3, 3, 1, 0, 0, 0]
10	4.6.0.3	$ABCeh$, $ADefi$, $ABCDgj$	[0, 3, 3, 1, 0, 0, 0]
10	4.6.0.3	$ABCeh$, $ADefi$, $BCDegj$	[0, 3, 3, 1, 0, 0, 0]
10	4.6.0.3	$ABCeh$, $ADefi$, $ABCfgj$	[0, 3, 3, 1, 0, 0, 0]
10	4.6.0.3	$ABCeh$, $ADefi$, $BCefgj$	[0, 3, 3, 1, 0, 0, 0]
10	4.6.0.3	$ABCeh$, $Aefgi$, $ABCDfj$	[0, 3, 3, 1, 0, 0, 0]
10	4.6.0.3	$ABefh$, $CDefi$, $ACegj$	[0, 3, 3, 1, 0, 0, 0]
10	4.6.0.3	$ABefh$, $CDefi$, $ABCegj$	[0, 3, 3, 1, 0, 0, 0]
10	4.6.0.3	$ABefh$, $CDefi$, $ABCDegj$	[0, 3, 3, 1, 0, 0, 0]
10	4.6.0.3	$ABefh$, $ACegi$, $ADfgj$	[0, 3, 3, 1, 0, 0, 0]
10	4.6.0.3	$ABefh$, $ACegi$, $Defgj$	[0, 3, 3, 1, 0, 0, 0]
10	4.6.0.3	$ABefh$, $Cefgi$, $ABDegj$	[0, 3, 3, 1, 0, 0, 0]
10	4.6.1.2	$ABCH$, $ABdefi$, $ACdegj$	[1, 0, 6, 0, 0, 0, 0]
10	5.5.0.3	$ABCfh$, $ADEfi$, $ABDgj$	[0, 3, 3, 1, 0, 0, 0]
10	5.5.0.3	$ABCfh$, $ADEfi$, $BDfgj$	[0, 3, 3, 1, 0, 0, 0]
10	5.5.0.3	$ABCfh$, $ADEfi$, $ABCDgj$	[0, 3, 3, 1, 0, 0, 0]
10	5.5.0.3	$ABCfh$, $ADEfi$, $BCDfgj$	[0, 3, 3, 1, 0, 0, 0]
10	5.5.0.3	$ABCfh$, $ADEfi$, $ABCDEgj$	[0, 3, 3, 1, 0, 0, 0]
10	5.5.0.3	$ABCfh$, $ADEfi$, $BCDEfgj$	[0, 3, 3, 1, 0, 0, 0]
10	5.5.0.3	$ABCfh$, $ABDgi$, $AEfgj$	[0, 3, 3, 1, 0, 0, 0]
10	5.5.0.3	$ABCfh$, $ADfgi$, $BEfgj$	[0, 3, 3, 1, 0, 0, 0]
10	5.5.0.3	$ABCfh$, $ADfgi$, $ABCEgj$	[0, 3, 3, 1, 0, 0, 0]
10	5.5.0.3	$ABCfh$, $ADfgi$, $BCEfgj$	[0, 3, 3, 1, 0, 0, 0]
10	5.5.0.3	$ABfgh$, $CDfgi$, $ABCEfj$	[0, 3, 3, 1, 0, 0, 0]
10	5.5.0.3	$ABfgh$, $CDfgi$, $ABCDEfj$	[0, 3, 3, 1, 0, 0, 0]
10	5.5.1.2	$ABCDH$, $ABefi$, $ACegj$	[0, 3, 3, 1, 0, 0, 0]
10	6.4.0.3	$ABCgh$, $ADEgi$, $BDFgj$	[0, 3, 3, 1, 0, 0, 0]
10	6.4.0.3	$ABCgh$, $ADEgi$, $BCDFgj$	[0, 3, 3, 1, 0, 0, 0]
10	6.4.0.3	$ABCgh$, $ADEgi$, $BCDEFgj$	[0, 3, 3, 1, 0, 0, 0]

Table 16 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
10	6.4.1.2	$ABCDH$, $ABEfi$, $ACEgj$	[0, 3, 3, 1, 0, 0, 0]
10	6.4.1.2	$ABCDH$, $ABEfi$, $ACfgj$	[0, 3, 3, 1, 0, 0, 0]
10	6.4.1.2	$ABCDEH$, $ABCfi$, $ADfgj$	[0, 3, 3, 1, 0, 0, 0]
10	6.4.2.1	$ABCH$, $ABDI$, $ACDefgj$	[3, 0, 0, 4, 0, 0, 0]
10	7.3.1.2	$ABCDH$, $ABEgi$, $ACFgj$	[0, 3, 3, 1, 0, 0, 0]
10	7.3.1.2	$ABCDEH$, $ABCgi$, $ADFgj$	[0, 3, 3, 1, 0, 0, 0]
10	7.3.1.2	$ABCDEFH$, $ABCgi$, $ADEgj$	[0, 3, 3, 1, 0, 0, 0]
10	7.3.2.1	$ABCH$, $ABDEI$, $ACDfgj$	[1, 2, 2, 2, 0, 0, 0]
10	8.2.2.1	$ABCDH$, $ABEFI$, $ACEgj$	[0, 3, 3, 1, 0, 0, 0]
11	1.10.0.4	$Abcdh$, $Abefi$, $Acegj$, $bdfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	1.10.0.4	$Abcdh$, $bcefi$, $bdegj$, $Aefgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	2.9.0.4	$ABcdh$, $ABefi$, $Acegj$, $Bdfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	2.9.0.4	$ABcdh$, $Acefi$, $Bcegj$, $Adfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	2.9.0.4	$ABcdh$, $Acefi$, $Adegj$, $Befgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	2.9.0.4	$Acdeh$, $Bcdfi$, $Bcegj$, $Adfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	2.9.0.4	$Acdeh$, $Bcdfi$, $cefgj$, $ABcdgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	3.8.0.4	$ABCdh$, $ABefi$, $ACegj$, $Bdfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	3.8.0.4	$ABCdh$, $ABefi$, $Adegj$, $Cefgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	3.8.0.4	$ABCdh$, $Adefi$, $Bdegj$, $Cefgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	3.8.0.4	$ABdeh$, $ACdfi$, $BCdgj$, $Aefgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	3.8.0.4	$ABdeh$, $ACdfi$, $ACegj$, $BCfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	3.8.0.4	$ABdeh$, $ACdfi$, $ACegj$, $Bdfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	3.8.0.4	$ABdeh$, $ACdfi$, $BCegj$, $Bdfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	3.8.0.4	$ABdeh$, $ACdfi$, $Aefgj$, $ABCdgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	3.8.0.4	$ABdeh$, $Cdefi$, $ACfgj$, $Bdfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	3.8.0.4	$Adefh$, $Bdegi$, $Cdfgj$, $ABCefgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	4.7.0.4	$ABCeh$, $ABDfi$, $ACDgj$, $Befgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	4.7.0.4	$ABCeh$, $ABDfi$, $ADegj$, $BCfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	4.7.0.4	$ABCeh$, $ABDfi$, $ADegj$, $CDfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	4.7.0.4	$ABCeh$, $ABDfi$, $Aefgj$, $ABCDgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	4.7.0.4	$ABCeh$, $ADefi$, $BDegj$, $ACfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	4.7.0.4	$ABCeh$, $ADefi$, $BDegj$, $CDfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	4.7.0.4	$ABCeh$, $ADefi$, $ABfgj$, $CDfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	4.7.0.4	$ABCeh$, $ADefi$, $ABfgj$, $ACDegk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	4.7.0.4	$ABCeh$, $ADefi$, $BDfgj$, $Cefgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	4.7.0.4	$ABCeh$, $ADefi$, $Befgj$, $ACDegk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	4.7.0.4	$ABefh$, $CDefi$, $ACegj$, $BDfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	4.7.0.4	$ABefh$, $CDefi$, $ACegj$, $BDefgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	4.7.0.4	$ABefh$, $CDefi$, $ABCegj$, $ACDfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	4.7.0.4	$ABefh$, $ACegi$, $ADfgj$, $BCDefgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	4.7.0.4	$ABefh$, $ACegi$, $ADfgj$, $ABCDefgk$	[0, 6, 6, 2, 1, 0, 0, 0]

Table 16 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
11	4.7.1.3	$ABCH$, $Adefi$, $Bdegj$, $Cdfgk$	[1, 4, 6, 4, 0, 0, 0, 0]
11	5.6.0.4	$ABCfh$, $ADEfi$, $ABDgj$, $BCEgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	5.6.0.4	$ABCfh$, $ADEfi$, $ABDgj$, $CEfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	5.6.0.4	$ABCfh$, $ADEfi$, $ABDgj$, $ACEfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	5.6.0.4	$ABCfh$, $ADEfi$, $BCDgj$, $BEfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	5.6.0.4	$ABCfh$, $ADEfi$, $BDfgj$, $ACEfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	5.6.0.4	$ABCfh$, $ADEfi$, $ABCDgj$, $ABEfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	5.6.0.4	$ABCfh$, $ABDgi$, $AEfgj$, $BCDEfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	5.6.0.4	$ABCfh$, $ABDgi$, $AEfgj$, $ABCDEfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	5.6.0.4	$ABCfh$, $ADEgi$, $BDfgj$, $CEfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	5.6.0.4	$ABCfh$, $ADEgi$, $BDfgj$, $ACEfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	5.6.0.4	$ABCfh$, $ADfgi$, $BEfgj$, $ABCDEgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	5.6.0.4	$ABfgh$, $CDfgi$, $ABCEfj$, $ACDEgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	5.6.1.3	$ABCDH$, $ABefi$, $ACegj$, $BDfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	6.5.0.4	$ABCgh$, $ADEgi$, $BDFgj$, $ACEFgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	6.5.0.4	$ABCgh$, $ADEgi$, $BDFgj$, $ABCDEFgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	6.5.1.3	$ABCDH$, $ABEfi$, $ACEgj$, $BDfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	6.5.1.3	$ABCDH$, $ABEfi$, $CDEgj$, $ACfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	6.5.1.3	$ABCDEH$, $ABCfi$, $ABDgj$, $AEfgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	6.5.2.2	$ABCH$, $ABDI$, $ACDefj$, $BCDegk$	[3, 0, 11, 0, 0, 0, 1, 0]
11	7.4.1.3	$ABCDH$, $ABEgi$, $ACFgj$, $DEFgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	7.4.1.3	$ABCDH$, $ABEgi$, $ACFgj$, $ADEFgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	7.4.1.3	$ABCDEH$, $ABCgi$, $ADFgj$, $ABEFgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	7.4.1.3	$ABCDEFH$, $ABCgi$, $ADEgj$, $BDFgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	7.4.2.2	$ABCH$, $ABDEI$, $ACDfj$, $DEfgk$	[1, 5, 6, 2, 0, 1, 0, 0]
11	7.4.3.1	$ABCH$, $ABDI$, $ACDJ$, $BCDefgk$	[7, 0, 0, 7, 0, 0, 0, 1]
11	8.3.2.2	$ABCDH$, $ABEFI$, $ACEgj$, $BDFgk$	[0, 6, 6, 2, 1, 0, 0, 0]
11	8.3.3.1	$ABCH$, $ABDI$, $ACDEJ$, $ABEfgk$	[3, 4, 3, 4, 0, 0, 1, 0]
11	9.2.3.1	$ABCH$, $ABDEI$, $ACDFJ$, $DEFgk$	[1, 5, 6, 2, 0, 1, 0, 0]

Table 16 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
12	1.11.0.5	$Abch$, $Adefi$, $bdegj$, $cdfgk$, $Abcefgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	1.11.0.5	$bcdh$, $Abefi$, $Acegj$, $Adfgk$, $bcdefgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	2.10.0.5	$ABch$, $Adefi$, $Bdegj$, $cdfgk$, $ABcefgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	2.10.0.5	$Acdh$, $ABefi$, $Bcegj$, $Bdfgk$, $Acdefgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	2.10.0.5	$cdeh$, $ABcfi$, $ABdgj$, $Aefgk$, $Bcdefgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	2.10.0.5	$cdeh$, $Acfgi$, $Bdfgj$, $ABcefk$, $ABdegl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	3.9.0.5	$ABdh$, $ACefi$, $BCegj$, $Cdfgk$, $ABdefgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	3.9.0.5	$Adeh$, $ABCfi$, $BCdgj$, $Befgk$, $ACdefgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	3.9.0.5	$Adeh$, $BCdfi$, $BCegj$, $ABfgk$, $ACdefgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	3.9.0.5	$Adeh$, $ABfgi$, $Cdfgj$, $ABCefk$, $BCdegl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	3.9.0.5	$defh$, $ABCdi$, $ABegj$, $ACfgk$, $BCdefgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	3.9.0.5	$defh$, $ABdgi$, $ACegj$, $ABCdfk$, $BCefgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	4.8.0.5	$ABeh$, $ACDfi$, $BCDgj$, $Cefgk$, $ABDefgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	4.8.0.5	$ABeh$, $ACDfi$, $CDegj$, $BCfgk$, $ABDefgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	4.8.0.5	$ABeh$, $CDefi$, $ACfgj$, $BDfgk$, $ABCDegl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	4.8.0.5	$ABeh$, $ACfgi$, $BDfgj$, $ACDefk$, $BCDegl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	4.8.0.5	$Aefh$, $BCDei$, $ABCgj$, $BDfgk$, $ACDefgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	4.8.0.5	$Aefh$, $ABCgi$, $BDegj$, $ABCDfk$, $CDefgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	4.8.0.5	$efgh$, $ABCei$, $ABDfj$, $ACDgk$, $BCDefgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	4.8.0.5	$efgh$, $ABCei$, $ABDfj$, $ACDegk$, $BCDfgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	4.8.0.5	$ABefh$, $CDefi$, $ACegj$, $BDegk$, $ABCDefgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]

Table 16 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
12	4.8.1.4	$ABCH$, $Adefi$, $Bdegj$, $Cdfgk$, $ABCefgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	5.7.0.5	$ABfh$, $CDEfi$, $ACDgj$, $BCEgk$, $ABDEfgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	5.7.0.5	$ABfh$, $ACDgi$, $BCEgj$, $ACDEfk$, $BDEfgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	5.7.0.5	$Afgh$, $BCDfi$, $BCEgj$, $ABDEfk$, $ACDEgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	5.7.0.5	$ABCfh$, $ADEfi$, $ABDgj$, $ACEgk$, $ABCDEfgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	5.7.1.4	$ABCH$, $ADefi$, $BDegj$, $CDfgk$, $ABCefgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	5.7.1.4	$ABCDH$, $Aefi$, $BCegj$, $BDfgk$, $ACDefgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	6.6.0.5	$ABgh$, $CDEgi$, $ACDFgj$, $BCEFgk$, $ABDEFgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	6.6.0.5	$ABCgh$, $ADEgi$, $BDFgj$, $CEFgk$, $ABCDEFgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	6.6.1.4	$ABCH$, $ADEfi$, $BDEgj$, $CDfgk$, $ABCEfgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	6.6.1.4	$ABCH$, $ADfgi$, $BEfgj$, $ACDEfk$, $BCDEgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	6.6.1.4	$ABCDH$, $AEfi$, $BCEgj$, $BDfgk$, $ACDEfgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	6.6.1.4	$ABCDH$, $Afgi$, $BCEfj$, $BDEgk$, $ACDEfgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	6.6.1.4	$ABCDEH$, $ABfi$, $ACDgj$, $CEfgk$, $BDEfgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	6.6.2.3	$ABCH$, $ABDI$, $ACefj$, $BCegk$, $Defgl$	[3, 6, 11, 8, 0, 2, 1, 0, 0]
12	7.5.1.4	$ABCH$, $ABDgi$, $ACEgj$, $BCFgk$, $ADEFgl$	[1, 10, 10, 5, 4, 0, 0, 1, 0]
12	7.5.1.4	$ABCDH$, $ABgi$, $ACEgj$, $BCFgk$, $DEFgl$	[1, 10, 10, 5, 4, 0, 0, 1, 0]
12	7.5.1.4	$ABCDEH$, $ABgi$, $ACDgj$, $CEFgk$, $BCDFgl$	[1, 10, 10, 5, 4, 0, 0, 1, 0]
12	7.5.1.4	$ABCDEFH$, $ABCgi$, $ADEgj$, $BDFgk$, $CEFgl$	[1, 10, 10, 5, 4, 0, 0, 1, 0]
12	7.5.2.3	$ABCH$, $ABDEI$, $ACDfj$, $BCDgk$, $AEfgl$	[1, 10, 10, 5, 4, 0, 0, 1, 0]
12	7.5.3.2	$ABCH$, $ABDI$, $ACDJ$, $ABefk$, $BCDegl$	[7, 4, 7, 8, 0, 4, 1, 0, 0]

Table 16 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
12	8.4.2.3	$ABCH$, $ADEFI$, $BDEgj$, $CDFgk$, $ABCEFgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	8.4.2.3	$ABCH$, $ABDEFI$, $ADEgj$, $CDFgk$, $BCEFgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	8.4.2.3	$ABCDH$, $ABEFI$, $CEgj$, $ADFgk$, $BCDEFgl$	[1, 8, 12, 8, 1, 0, 0, 0, 1]
12	8.4.3.2	$ABCH$, $ABDI$, $ACDEJ$, $ABEfk$, $ACDfgl$	[3, 8, 11, 4, 0, 4, 1, 0, 0]
12	8.4.4.1	$ABCH$, $ABDI$, $ACDJ$, $BCDK$, $ABefgl$	[14, 0, 4, 0, 9, 0, 4, 0, 0]
12	9.3.3.2	$ABCH$, $ABDEI$, $ACDFJ$, $BCDgk$, $AEFgl$	[1, 10, 10, 5, 4, 0, 0, 1, 0]
12	9.3.4.1	$ABCH$, $ABDI$, $ABEJ$, $ACDEK$, $BCDEfgl$	[6, 8, 4, 8, 1, 0, 4, 0, 0]
12	10.2.4.1	$ABCH$, $DEFI$, $ABDEJ$, $ACDFK$, $BCDEgl$	[2, 10, 10, 4, 1, 2, 2, 0, 0]
13	1.12.0.6	$Abch$, $defi$, $Abdej$, $Acdgk$, $bcfgl$, $Abefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	1.12.0.6	$Abch$, $defi$, $Abdej$, $Adfgk$, $cefgl$, $bcdegm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	1.12.0.6	$bcdh$, $efgi$, $Abcej$, $Abdfk$, $cdegl$, $Acdfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	1.12.0.6	$bcdh$, $Abcei$, $Abcfj$, $Abdgk$, $befgl$, $Abcdefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	2.11.0.6	$ABch$, $defi$, $ABdej$, $Acdgk$, $Bcfgl$, $ABefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	2.11.0.6	$ABch$, $defi$, $ABdej$, $Adfgk$, $cefgl$, $Bcdegm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	2.11.0.6	$ABch$, $defi$, $Acdej$, $ABdgk$, $Bcfgl$, $Acefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	2.11.0.6	$ABch$, $defi$, $Acdej$, $Adfgk$, $Befgl$, $Bcdegm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	2.11.0.6	$Acdh$, $Befi$, $ABcej$, $ABdgk$, $cdfgl$, $Acefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	2.11.0.6	$Acdh$, $efgi$, $ABcej$, $ABdfk$, $cdegl$, $Bcdfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	2.11.0.6	$Acdh$, $efgi$, $ABefj$, $Acegk$, $Bdfgl$, $Bcdegm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	2.11.0.6	$Acdh$, $efgi$, $Acefj$, $Bcegk$, $Bdfgl$, $ABdefm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	2.11.0.6	$Acdh$, $ABcei$, $ABcfj$, $ABdgk$, $Aefgl$, $ABcdefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]

Table 16 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
13	2.11.0.6	$Acdh$, $ABcei$, $ABcfj$, $Adegk$, $cdfgl$, $Befgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	2.11.0.6	$cdeh$, $ABcdi$, $Acefj$, $Acegk$, $Bcfgl$, $ABcdefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	2.11.0.6	$cdeh$, $ABcdi$, $Acefj$, $Bcegk$, $cdfgl$, $ABdefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	2.11.0.6	$cdeh$, $ABcdi$, $Acfgj$, $Befgk$, $ABcefl$, $ABdegm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$ABdh$, $Cefi$, $ABCej$, $ACdgk$, $Bdfgl$, $ABefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$ABdh$, $Cefi$, $ABCej$, $ACfgk$, $defgl$, $BCdegm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$ABdh$, $Cefi$, $ACdej$, $ABCgk$, $Bdfgl$, $Adefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$ABdh$, $Cefi$, $ACdej$, $BCdgk$, $ABfgl$, $Adefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$ABdh$, $Cefi$, $ACdej$, $ACfgk$, $Befgl$, $BCdegm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$ABdh$, $efgi$, $ABCej$, $ACdfk$, $Bdegl$, $BCdfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$ABdh$, $efgi$, $ACdej$, $BCdfk$, $ABegl$, $ABCfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$ABdh$, $efgi$, $ABefj$, $ACegk$, $Cdfgl$, $BCdefm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$ABdh$, $efgi$, $ACefj$, $BCegk$, $Adfgl$, $BCdfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$ABdh$, $efgi$, $ACefj$, $Adegk$, $ABCegl$, $ABCdfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$ABdh$, $efgi$, $Adefj$, $Cdegk$, $BCdefl$, $ABCdfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$ABdh$, $ABCei$, $ABCfj$, $ACdgk$, $Aefgl$, $ABCdefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$ABdh$, $ABCei$, $ABCfj$, $Adegk$, $Bdfgl$, $Cefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$ABdh$, $ABCei$, $ACdfj$, $ACdgk$, $Aefgl$, $ABCdefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$ABdh$, $ACdei$, $ACdfj$, $ABegk$, $Bdfgl$, $Cefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$Adeh$, $Bfgi$, $ABCdj$, $ABefk$, $Cdegl$, $ACefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$Adeh$, $Bfgi$, $ABdfj$, $ACefk$, $Cdegl$, $ABCdgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]

Table 16 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
13	3.10.0.6	$Adeh$, $ABCdi$, $ABdfj$, $ABegk$, $ACfgl$, $ABCdefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$Adeh$, $ABCdi$, $ABefj$, $ABegk$, $ACfgl$, $ABCdefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$Adeh$, $ABCdi$, $ABefj$, $ABegk$, $Cefgl$, $BCdfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$Adeh$, $ABCdi$, $ABefj$, $ACegk$, $Adfgl$, $BCdefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$Adeh$, $ABCdi$, $ABfgj$, $Cefgk$, $ABCefl$, $BCdegm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$Adeh$, $ABCfi$, $ABCgj$, $Adfgk$, $ABefgl$, $Cdefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$Adeh$, $ABdfi$, $ACefj$, $ABdgk$, $Cdegl$, $BCfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$Adeh$, $ABdfi$, $ACefj$, $ACdgk$, $ABegl$, $BCdefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$Adeh$, $ABdfi$, $ACefj$, $BCdgk$, $defgl$, $BCefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$Adeh$, $ABdfi$, $ACefj$, $defgk$, $ABCfgl$, $ABCdegm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$defh$, $ABdei$, $ACdfj$, $BCdgk$, $Aefgl$, $ABCdefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$defh$, $ABdei$, $ACdfj$, $Cdegk$, $Bdfgl$, $ABCefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	3.10.0.6	$defh$, $ABCgi$, $Adegj$, $Bdfgk$, $BCdegl$, $ACefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $CDfi$, $ACDej$, $ABCgk$, $Befgl$, $ADefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $CDfi$, $ACDej$, $BCegk$, $ABfgl$, $ADefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $CDfi$, $ACDej$, $ACfgk$, $BDfgl$, $BCDegm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $CDfi$, $ACefj$, $ABCgk$, $BDegl$, $ADefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $Cfgi$, $ACDej$, $ABCfk$, $BDegl$, $ABDfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $Cfgi$, $ACDej$, $ABDfk$, $BCegl$, $BDefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $Cfgi$, $ABCfj$, $ADefk$, $BDegl$, $ABCDgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $Cfgi$, $ABCfj$, $ACDgk$, $Defgl$, $BCDefm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]

Table 16 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
13	4.9.0.6	$ABeh$, $Cfgi$, $ABDfj$, $ACefk$, $BDegl$, $ACDegm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $Cfgi$, $ABDfj$, $ACegk$, $BDegl$, $ACDefm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $Cfgi$, $ACDfj$, $BCDgk$, $Aefgl$, $BDefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $Cfgi$, $ACDfj$, $ACegk$, $ABCDgl$, $ABDefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $Cfgi$, $ACefj$, $CDegk$, $BCDefl$, $ABDefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $ACDei$, $ABCfj$, $ABCgk$, $ADfgl$, $ABCDefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $ACDei$, $ABCfj$, $ABDgk$, $Aefgl$, $BCDefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $ACDei$, $ABCfj$, $ACegk$, $ADfgl$, $ABCDefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $ACDei$, $ABCfj$, $BCegk$, $ADfgl$, $ABCDefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $ACDei$, $ABCfj$, $ADegk$, $CDfgl$, $Befgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $ACDei$, $ACefj$, $BDegk$, $ABfgl$, $CDfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $ACDei$, $BCefj$, $BDegk$, $ABfgl$, $ACDefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $ACDei$, $ACfgj$, $BDfgk$, $ABCDfl$, $BCDegm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $ACDei$, $ACfgj$, $BDfgk$, $BCDefl$, $BCDegm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $ABCfi$, $ADefj$, $ABCgk$, $BDegl$, $CDfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $ABCfi$, $ADefj$, $ABDgk$, $ACegl$, $BCDefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $ABCfi$, $ADefj$, $ACDgk$, $Befgl$, $ABCDefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $ABCfi$, $ADefj$, $BCegk$, $ADegl$, $CDfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $ABCfi$, $ADefj$, $CDegk$, $Befgl$, $BCDfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $ACDfi$, $ACDgj$, $ABfgk$, $ACefgl$, $BDefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$ABeh$, $ACefi$, $BDefj$, $BCegk$, $ADegl$, $ABCDfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]

Table 16 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
13	4.9.0.6	$ABeh$, $ACefi$, $BDefj$, $CDegk$, $ABfgl$, $ABCDefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$Aefh$, $ABCei$, $ABDfj$, $ACDgk$, $ABegl$, $ABCDefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$Aefh$, $ABCei$, $ABDfj$, $ACDgk$, $Befgl$, $ABCDefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$Aefh$, $ABCei$, $ABDfj$, $ABegk$, $CDegl$, $BCDfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$Aefh$, $ABCei$, $ABDfj$, $ADegk$, $ACfgl$, $BCDefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$Aefh$, $ABCei$, $ABDfj$, $ADegk$, $Cefgl$, $ABCDfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$Aefh$, $ABCei$, $ABDfj$, $CDegk$, $Befgl$, $BCDfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$Aefh$, $ABCei$, $ABDfj$, $Befgk$, $ABCDgl$, $ACDefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$Aefh$, $ABCei$, $ABDgj$, $CDfgk$, $ABCDfl$, $BCefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$Aefh$, $ABCei$, $ABDgj$, $CDfgk$, $BCDefl$, $ABCfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$Aefh$, $ABCei$, $ADegj$, $ABfgk$, $ABCDfl$, $CDefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$Aefh$, $ABCei$, $BDegj$, $CDfgk$, $ABCDfl$, $BCefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$Aefh$, $ABCei$, $ABfgj$, $Defgk$, $ABCDfl$, $ACDegm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$Aefh$, $BCDgi$, $ABegj$, $ACfgk$, $ACDegl$, $BDefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$efgh$, $ABefi$, $CDefj$, $ACegk$, $BDegl$, $ABCDfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.0.6	$efgh$, $ABefi$, $ACegj$, $ACDefk$, $ABDegl$, $ABCDfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.1.5	$ABCH$, $defi$, $ABdej$, $ACdgk$, $BCfgl$, $ABefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	4.9.1.5	$ABCH$, $defi$, $ABdej$, $Adfgk$, $Cefgl$, $BCdegm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $CDgi$, $ACDfj$, $BCEfk$, $ABEgl$, $ADEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $CDgi$, $ACDfj$, $ACEgk$, $BDEgl$, $BCDEfm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $CDgi$, $ACDfj$, $CEfgk$, $BCDEfl$, $ABDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]

Table 16 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
13	5.8.0.6	$ABfh$, $CDgi$, $ACEfj$, $ABEgk$, $BDfgl$, $BCDEfm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $CDgi$, $CDEfj$, $ACfgk$, $BCEfgl$, $ABDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $ACDfi$, $ACEfj$, $ABCgk$, $ADEgl$, $ABCDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $ACDfi$, $BCEfj$, $ABCgk$, $ADEgl$, $BCDEgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $ACDfi$, $BCEfj$, $ABCgk$, $ADEgl$, $CDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $ACDfi$, $BCEfj$, $ABCgk$, $DEfgl$, $ABCDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $ACDfi$, $BCEfj$, $ABDgk$, $CDEgl$, $BEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $ACDfi$, $BCEfj$, $ABDgk$, $AEfgl$, $BCDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $ACDfi$, $BCEfj$, $BDfgk$, $AEfgl$, $ABCDEgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $ACDfi$, $ABCgj$, $CDEgk$, $BCDEfl$, $ACEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $ACDfi$, $ABCgj$, $CDEgk$, $BCDEfl$, $BCEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $ACDfi$, $ABEgj$, $BCfgk$, $BCDEfl$, $ADEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $ACDfi$, $ACEgj$, $BDEgk$, $BCDEfl$, $ABCDgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $ACDfi$, $ACEgj$, $BDEgk$, $BCDEfl$, $BCDfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $ACDfi$, $BCfgj$, $AEfgk$, $BCDEfl$, $ABDEgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $ACDfi$, $CEfgj$, $BCDEfk$, $ABCDgl$, $ABDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $CDEfi$, $ABCgj$, $ADEgk$, $ACDfgl$, $BCEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $CDEfi$, $ABCgj$, $ADfgk$, $BEfgl$, $ABCDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $ABCgi$, $CDEgj$, $ADfgk$, $ABDEgl$, $BCEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABfh$, $ABCgi$, $CDEgj$, $ADfgk$, $ACEfgl$, $BDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$Afgh$, $ABCfi$, $ADEfj$, $ABDgk$, $ACEgl$, $BCDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]

Table 16 Continued

n_1+n_2	$n_1.n_2.k_1.k_2$	defining words	word length pattern
13	5.8.0.6	$Afgh$, $ABCfi$, $ADEfj$, $ABDgk$, $ABCEgl$, $CDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$Afgh$, $ABCfi$, $BDEfj$, $ACDgk$, $CEfgl$, $BCDEgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$Afgh$, $ABCfi$, $ABDgj$, $ABDEfk$, $ABCEgl$, $BCDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.0.6	$ABCfh$, $ADEfi$, $BDfgj$, $BCDEfk$, $ACEfgl$, $ABCDEgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.1.5	$ABCH$, $Defi$, $ABDej$, $ACDgk$, $BCfgl$, $ABefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.1.5	$ABCH$, $Defi$, $ABDej$, $ADfgk$, $Cefgl$, $BCDegm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.1.5	$ABCH$, $efgi$, $ABDej$, $ACDfk$, $BCegl$, $BCDfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.1.5	$ABCH$, $efgi$, $ABefj$, $ADegk$, $CDfgl$, $BCDefm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.1.5	$ABCH$, $ABDei$, $ABDfj$, $ACDgk$, $Aefgl$, $ABCDefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.1.5	$ABCH$, $ABDei$, $ABDfj$, $ACegk$, $BCfgl$, $Defgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.1.5	$ABCDH$, $ABei$, $CDfj$, $ACegk$, $Befgl$, $ABDfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.1.5	$ABCDH$, $ABei$, $Cfgj$, $ACefk$, $BDegl$, $ADefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	5.8.1.5	$ABCDH$, $ABei$, $ACefj$, $ACegk$, $ADfgl$, $ABCDefgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	6.7.0.6	$ABgh$, $ACDgi$, $ACEgj$, $BCFgk$, $DEFgl$, $ABCDEFgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	6.7.0.6	$ABgh$, $ACDgi$, $BCEgj$, $BDFgk$, $ACEFgl$, $ADEFgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	6.7.1.5	$ABCH$, $DEfi$, $ABDfj$, $ACDgk$, $BCEgl$, $ABEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	6.7.1.5	$ABCH$, $DEfi$, $ABDfj$, $ADEgk$, $CEfgl$, $BCDfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	6.7.1.5	$ABCH$, $Dfgi$, $ABDfj$, $ACEfk$, $BCEgl$, $ABDEgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	6.7.1.5	$ABCH$, $Dfgi$, $ABDfj$, $ADEgk$, $CEfgl$, $BCDEfm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	6.7.1.5	$ABCH$, $ABDfi$, $ACEfj$, $ABDgk$, $BCEgl$, $DEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	6.7.1.5	$ABCH$, $ABDfi$, $ACEfj$, $ACDgk$, $ABEgl$, $BCDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]

Table 16 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
13	6.7.1.5	$ABCH$, $ABDfi$, $ACEfj$, $ADEgk$, $BCfgl$, $ABCDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	6.7.1.5	$ABCDH$, $ABfi$, $CDgj$, $ACEfk$, $BEfgl$, $ABDEgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	6.7.1.5	$ABCDH$, $ABfi$, $CEgj$, $ACEfk$, $BDfgl$, $ADEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	6.7.1.5	$ABCDH$, $ABfi$, $CEgj$, $ADEfk$, $BCfgl$, $BDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	6.7.1.5	$ABCDH$, $ABfi$, $ACEfj$, $ABEgk$, $ADfgl$, $BCDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	6.7.1.5	$ABCDH$, $ABfi$, $ACEfj$, $ADEgk$, $ACfgl$, $ABCDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	6.7.1.5	$ABCDH$, $ABfi$, $ACEfj$, $ADEgk$, $BCfgl$, $ABCDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	6.7.1.5	$ABCDH$, $ABfi$, $ACEfj$, $CDEgk$, $BDfgl$, $AEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	6.7.1.5	$ABCDEH$, $ABfi$, $CDgj$, $ACDfk$, $ACEgl$, $ABDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	6.7.1.5	$ABCDEH$, $ABfi$, $ACDfj$, $ABCgk$, $AEfgl$, $BDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	6.7.1.5	$ABCDEH$, $ABfi$, $ACDfj$, $ACEgk$, $BCDfgl$, $ABDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	6.7.2.4	$ABCH$, $ABDI$, $ABefj$, $ACegk$, $ADfgl$, $ABCDefgm$	[3, 15, 15, 13, 12, 1, 1, 3, 0, 0]
13	7.6.1.5	$ABCH$, $DEgi$, $ABDgj$, $ACFgk$, $ABEFgl$, $BCDEFgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	7.6.1.5	$ABCH$, $ABDgi$, $ACEgj$, $DEFgk$, $ACDFgl$, $BCEFgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	7.6.1.5	$ABCDH$, $ABgi$, $ACEgj$, $BCFgk$, $DEFgl$, $ABCDEFgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	7.6.1.5	$ABCDEH$, $ABgi$, $ACDgj$, $BCFgk$, $AEFgl$, $BDEFgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	7.6.1.5	$ABCDEFH$, $ABgi$, $ACDgj$, $BCEgk$, $BDFgl$, $AEFgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	7.6.2.4	$ABCH$, $ABDEI$, $DEfj$, $ACDgk$, $BCfgl$, $ABEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	7.6.2.4	$ABCH$, $ABDEI$, $DEfj$, $ADfgk$, $CEfgl$, $BCDEgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	7.6.2.4	$ABCH$, $ABDEI$, $Dfgj$, $ACDfk$, $BCEgl$, $ACEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	7.6.2.4	$ABCH$, $ABDEI$, $ACDfj$, $ACDgk$, $AEfgl$, $ABCDEfgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]

Table 16 Continued

$n_1 + n_2$	$n_1.n_2.k_1.k_2$	defining words	word length pattern
13	7.6.3.3	$ABCH$, $ABDI$, $ACDJ$, $ABefk$, $ACegl$, $ADfgm$	[7, 12, 9, 16, 12, 4, 3, 0, 0, 0]
13	8.5.2.4	$ABCH$, $ADEFI$, $DEgj$, $ABDgk$, $CEFgl$, $BCDFgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	8.5.2.4	$ABCDH$, $ABEFI$, $ABgj$, $ACEgk$, $ADFgl$, $BCDEFgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	8.5.2.4	$ABCDH$, $ABEFI$, $ACgj$, $ADEgk$, $ADFgl$, $ABCDEFgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	8.5.3.3	$ABCH$, $ABDI$, $ACDEJ$, $ABEfk$, $ABEgl$, $ACDfgm$	[4, 12, 22, 8, 3, 12, 2, 0, 0, 0]
13	8.5.4.2	$ABCH$, $ABDI$, $ACDJ$, $BCDK$, $ABefl$, $ACegm$	[14, 8, 4, 16, 9, 8, 4, 0, 0, 0]
13	9.4.3.3	$ABCH$, $DEFI$, $ABDEJ$, $ACDgk$, $BCFgl$, $ABEFgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	9.4.3.3	$ABCH$, $ABDEI$, $ACDFJ$, $DEgk$, $BCFgl$, $ABEFgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	9.4.3.3	$ABCH$, $ABDEI$, $ACDFJ$, $EFgk$, $BCEgl$, $BCDFgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	9.4.3.3	$ABCH$, $ABDEI$, $ACDFJ$, $ABDgk$, $AEFgl$, $ABCDEFgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	9.4.3.3	$ABCH$, $ABDEI$, $ACDFJ$, $BCDgk$, $AEFgl$, $ABCDEFgm$	[2, 16, 18, 10, 9, 4, 2, 2, 0, 0]
13	9.4.4.2	$ABCH$, $ABDI$, $ABEJ$, $ACDEK$, $ACfgl$, $BCDEfm$	[6, 16, 12, 8, 9, 8, 4, 0, 0, 0]
13	10.3.4.2	$ABCH$, $DEFI$, $ABDEJ$, $ACDFK$, $ACDEgl$, $BCDFgm$	[2, 16, 20, 8, 5, 8, 4, 0, 0, 0]
13	10.3.5.1	$ABCH$, $ABDI$, $ABEJ$, $ACDEK$, $BCDEL$, $ACfgm$	[10, 18, 4, 6, 13, 6, 4, 2, 0, 0]
13	11.2.5.1	$ABCH$, $ABDI$, $ACDEJ$, $ACDFK$, $ABEFL$, $BCDEFgm$	[4, 16, 18, 8, 3, 8, 6, 0, 0, 0]

VITA

Abhishek Kumar Shrivastava received his Bachelors of Technology (Honors) in industrial engineering from Indian Institute of Technology Kharagpur, India, in 2003. He joined Texas A&M University in Fall 2003 for graduate studies in industrial engineering, where he was associated with the Advanced Metrology Lab. Abhishek's research interests are in the integration of statistical and optimization methods for modeling and analyzing complex systems. His current research focuses on methods in design of experiments, data mining, spatio-temporal statistics and combinatorial optimization. He will be joining the faculty of Department of Manufacturing Engineering and Engineering Management at City University of Hong Kong in October 2009.

Abhishek may be reached at his work address:

Department of MEEM
Room Y6700, 6/F, Academic Building
City University of Hong Kong,
83 Tat Chee Avenue,
Kowloon Tong,
HONG KONG

This document was typeset in LaTeX by the author.