

# Computing Factorial Designs for Large Numbers of Factors with $2^m$ levels

T. C. Peachey<sup>1</sup>, N. T. Diamond<sup>2</sup>, D. Abramson<sup>1</sup>, and C. M. Enticott<sup>1</sup>

<sup>1</sup> Caulfield School of Information Technology, Monash University, Melbourne, Australia

<sup>2</sup> Dept. of Econometrics and Business Statistics

**Abstract.** A new algorithm is presented for fractional factorial experimental design. It is faster than existing algorithms and hence can provide designs with more factors than previous designs. Such large experiments are required in the new field of parametric computing.

## 1 Introduction

Fractional Factorial Designs are the most commonly used statistical experimental designs, allowing the estimation of main effects and various interaction effects with less experimentation than using full factorial designs. Typically the number of factors and experimental runs is relatively small, ranging from  $n = 3$  to 11 factors, and  $N = 4$  to 128 runs in the catalogue given in Box, Hunter and Hunter, [1, p. 272], one of the most influential textbooks on the subject

Increasingly, physical experimentation is being augmented by the use of computer models [2]. A new field of computer science, parametric computing, has developed to facilitate the repeated execution of computer models where the input parameters (factors) are varied. The growing availability of clusters of processors allow parallel execution of these models. Software tools such as Nimrod [3] facilitate such experiments. For example, in a demonstration at SuperComputing 2003 [4], a quantum chemistry model [5] was performed 64000 times, each run taking about 30 min of execution time. Thus for experiments performed on computer models, it is possible to perform large numbers of runs and hence deal with large numbers of factors.

In this paper we are concerned with the design of regular fractional factorial experiments where the number of factors is large. We focus on the case where factors are restricted to two levels each, denoted by  $-$  and  $+$ , although the results can be applied to cases where the factors have  $2^m$ ,  $m = 2, 3, \dots$  levels, since it is well-known [6] that a factor with  $2^m$  levels may be replaced by  $m$  quasi-factors, each with 2 levels. Our motivation for this work is to develop software that provides experimental designs for large parameter sweep experiments.

The most important fractional factorial designs are those of resolution III, IV and V, where the resolution of a design is the minimum “word length”, defined

as the number of letters, of the elements of the defining relation, consisting of all the effects and interactions that are equal to a column of + signs. Resolution III designs allow the estimation of all main effects, but they may be biased by two-factor interactions; Resolution IV designs allow the estimation of all main effects unbiased by two-factor interactions, but the two-factor interactions may not be estimable; and Resolution V designs allow the estimation of all main effects and all two-factor interactions. Resolution V designs require more runs than Resolution IV designs which in turn require more runs than Resolution III designs. In order to rank designs of the same resolution, Fries and Hunter [7] introduced the criterion of minimum aberration, the number of words of minimum length in the defining relation.

Tables of minimum aberration designs have been presented in the literature. For example Franklin [8] presented the word length pattern of two-level designs for 7 to 18 factors, Chen, Sun, and Wu [9] enumerated all possible  $2^{n-p}$  fractional factorials of size 16 and 32 and all resolution IV (or higher) fractions of size 64 and hence provided the minimum aberration designs, while Block and Mee [10] enumerate all 128 run resolution IV designs for between 12 and 40 factors.

In addition to the tables, there are various methods that have been presented in the literature for constructing minimum aberration designs. For example, minimum aberration resolution III designs can be constructed based on the saturated resolution IV designs involving  $N/2$  factors, and a minimum aberration design for  $N/2$  runs and  $n - N/2$  factors [11]. Chen and Cheng [12] show that minimum aberration resolution IV designs with  $5N/16 < n \leq N/2$  must be projections of the saturated resolution IV designs involving  $N/2$  factors, and that for  $9N/32 < n \leq 5N/16$  factors, all minimum aberration designs are projections of the design with  $5N/16$  factors which is constructed by repeatedly “doubling” the  $2^{5-1}$  design defined by  $I = ABCDE$ , where the double of a design  $X$  is given by

$$\begin{pmatrix} X & X \\ X & -X \end{pmatrix}.$$

Finally, Draper and Mitchell [13, 14, 15] show and use the fact that resolution V designs of size  $2^{n-p}$  can always be constructed from even (i.e. only words of even length are allowed in the defining relation) resolution VI designs of size  $2^{n-p+1}$  and then “erasing” one of the letters from all of the words, and this is usually much quicker than constructing the resolution V design directly.

These tables and theories are useful, but only partially address the problem of designs for the number of factors that may occur in parametric computing. There is a need for an algorithm that provides designs outside of the published results, and the development of such an algorithm is the main goal of this paper.

One of the first algorithms for the design of fractional factorial experiments is that of Franklin and Bailey [16]. However this algorithm performs an exhaustive search and becomes quickly computationally impractical as the number of factors grows. Liao and Iyer [17] have devised an alternative, called the “sequential

elimination of factors (SEF)” algorithm, that can handle more factors. Their algorithm is stochastic in nature and cannot guarantee the smallest possible number of runs, nor the best aberration. In our implementation we run the algorithm several times and select the best design. Liao and Iyer did not give details of the execution time of the SEF algorithm nor the maximum number of factors possible. In Section 2 we investigate various implementations of that algorithm in order to optimize its speed. We find that the practical limit for that algorithm is about 80 factors with 2 levels each and 14 factors with 8 levels each. For parametric modelling experiments this may be insufficient.

In order to obtain designs for more factors we propose here a new algorithm, based on the SEF method, that extends the reach to over 100 factors. Performance of these algorithms depend critically on details of the implementation. So in the next section we describe the SEF algorithm, our implementation and its performance. The following section presents the new algorithm, its implementation and comparison with the SEF algorithm.

## 2 The SEF Algorithm of Liao and Iyer

The algorithm is most easily understood from an example.

### 2.1 An Example

Suppose that there are 5 factors,  $a, b, c, d, e, f$ . The corresponding single effects will be written  $A, B, C, D, E$ . We adopt the convention that these take the values  $+1$  or  $-1$ , rather than the alternative  $+1$  and  $0$ . Here we assume that the reader is familiar with manipulation of these functions, so that for example the product  $A \times A$  is  $I$ , the constant function, always  $+1$ . And so  $AB \times ACD = BCD$ . We maintain a list of current effects; at this stage it is  $\mathbf{E} = [A, B, C, D, E]$ .

To create a design the user needs to specify the effects that are expected to be significant. Of these, the primary effects  $\mathbf{P}$  are those for which estimates are required and the secondary effects  $\mathbf{S}$  are those for which estimates are not required. All other effects are expected to be negligible. For example, to obtain a resolution III design,  $\mathbf{P}$  consists of the identity  $I$  together with all first order effects and  $\mathbf{S}$  is the empty set. That is,

$$\mathbf{P} = \{I, A, B, C, D, E\} \quad \text{and} \quad \mathbf{S} = \emptyset.$$

The first task for the algorithm is to construct a new set of effects, the ineligible set,  $\mathbf{X}$ , is formed by taking the cross product of  $\mathbf{P}$  with itself, augmented by the cross product of  $\mathbf{P}$  with  $\mathbf{S}$ . In this case

$$\mathbf{X} = \{I, A, B, C, D, E, AB, AC, AD, AE, BC, BD, BE, CD, CE, DE\}$$

Now we enter a loop. Select at random an effect, that is not in  $\mathbf{X}$ , say we select  $ADE$ . Choose a factor from this effect, say  $E$ . Then we use the effect to eliminate the factor  $E$  from  $\mathbf{X}$ . To do this, select each effect in  $\mathbf{X}$  that contains  $E$ , and replace it by its product with  $ADE$ . Deleting duplications, this gives an updated set

$$\mathbf{X} = \{I, A, B, C, D, AD, AB, AC, BC, BD, ABD, CD, ACD\}.$$

We also replace all occurrences of  $E$  in  $\mathbf{E}$  in the same way, giving  $\mathbf{E} = [A, B, C, D, AD]$ .

For the next iteration we again select an effect not in  $\mathbf{X}$ , but this time only factors  $A, B, C, D$  are permitted. Suppose we select  $BCD$  and decide to use it to eliminate  $D$ . Then we obtain

$$\mathbf{X} = \{I, A, B, C, BC, ABC, AB, AC, AC\}.$$

The new effect is added also multiplied by each element of  $\mathbf{E}$  containing  $D$ .

At this stage the loop terminates as  $\mathbf{X}$  is now a complete set on the surviving factors so finding an effect not in  $\mathbf{X}$  is impossible. The  $2^{5-2}$  fractional factorial design is then provided by the surviving factors and the defining effects  $\mathbf{D}$  set to  $I$ . We consider the design to be complete at this stage with two outputs:  
(i) a Boolean list showing which factors survived, in this case  $[1, 1, 1, 0, 0]$ ,  
(ii) a list of effects showing the defining contrasts, here  $[A, B, C, BC, ABC]$ .  
In practice this design would be followed by the trivial task of generating the design table, a full factorial design on the surviving factors with corresponding values for the other factors.

## 2.2 The Formal Algorithm

For a precise definition of the algorithm, and for efficient implementation, it is more convenient to use set notation. We define the factors to be symbols  $A, B, \dots$ , and the full set of  $n$  factors will be denoted by  $\mathcal{F}$ . An effect is a subset of  $\mathcal{F}$  (an element of  $2^{\mathcal{F}}$ ) and will be denoted by just the concatenation of the symbols, such as  $ABD$ . The effect which is the empty set will be denoted by  $I$ , so this symbol will not be allowed as one of the factors. Sets of effects will be denoted by boldface letters as in  $\mathbf{S} = \{I, A, AB, AC\}$ . When discussing effects in general we will represent them by lower case letters,  $p, q, \dots$ .

Products of effects will then be the symmetric difference,

$$p \ominus q = (p \setminus q) \cup (q \setminus p).$$

For example  $ABC \ominus AB = C$  and  $A \ominus I = A$ . The same notation will be used for the pair-wise operation on sets of effects,

$$\mathbf{S} \ominus \mathbf{T} = \{s \ominus t : s \in \mathbf{S}, t \in \mathbf{T}\}.$$

In this notation the algorithm of Liao and Iyer is as shown in Figure 1.

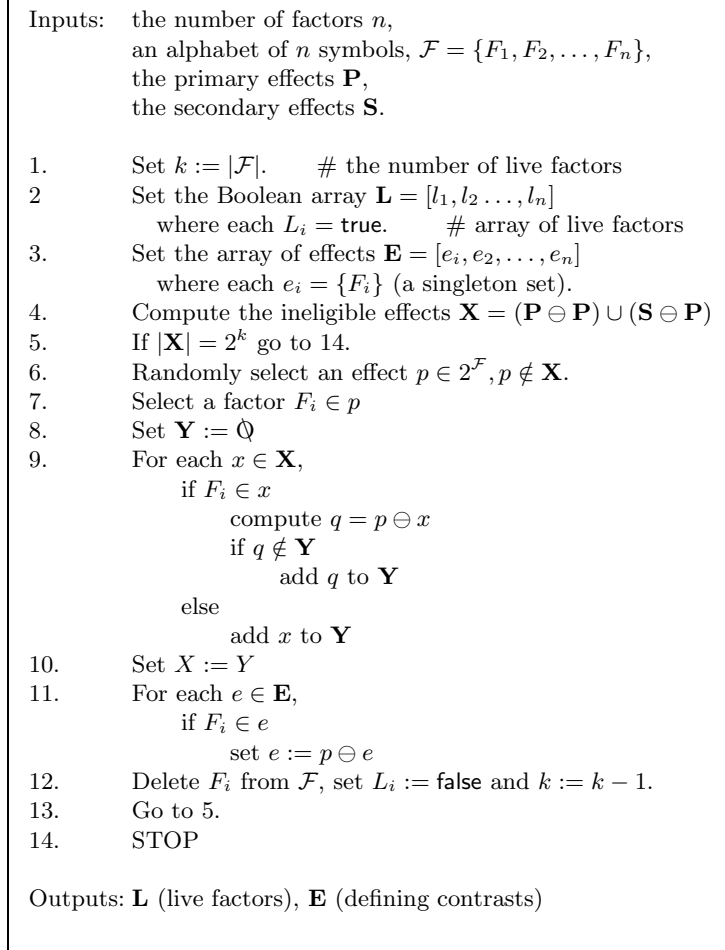


**Fig. 1.** The SEF Algorithm

## 2.3 Implementation

The SEF algorithm was coded in C to run on Linux and on Windows. The efficiency of this algorithm is sensitive to details of the implementation, so this section gives the salient aspects of our implementation. Complete information may be obtained by inspecting the source code which is available for free download at the Nimrod download site [18].

Effects were implemented as sets using a Bitset structure, basically an array of unsigned long integers. Bits in these longs correspond to factors; a factor is present or absent in an effect according to whether the corresponding bit is 1 or 0. Then the symmetric difference operation may be efficiently computed as the exclusive or of the longs. If the number of factors is no greater than the number of bits per long (32 for our platforms) then one long is sufficient.

Sets of effects (**P**, **S**, **X**, **Y** and **E** in the algorithm) are stored as arrays of the Bitset structures. Steps 6 and 9 in the algorithm involve determining whether an effect is contained in a set of effects, an operation we will denote as *EffectInSet()*. Step 4 also involves such determinations because as each element is added to **X** the implementation must avoid duplication of elements already in the set. For large designs these sets of effects may contain many thousands of elements. Consequently, the operation *EffectInSet()* provides the dominant part of the execution time.

Our first implementation of *EffectInSet()*, referred to as *I1*, used a simple loop; the given effect is compared with each effect in the set of effects. This version became unacceptably slow for 30 or more factors. An alternative, *I2* used a hash value computed from the long integers in the Bitset structure representing the effect. As each effect is added to the set of effects, its corresponding hash value is added to an open hash table. Then, the *EffectInSet()* operation is just a matter of checking whether the hash position is occupied.

Because a new list of effects is created for each iteration of the algorithm, it is necessary to free the hash table memory at each iteration, otherwise memory usage becomes prohibitive. We found that a table of size  $2^{16}$  is a reasonable compromise between memory allocation (and deallocation) and collision frequency.

## 2.4 Aberration

For resolution  $r$  designs, effects of length  $r - 1$  or less cannot be in the defining relation. An indication of the robustness of a design is provided by the aberration, the number of length  $r$  effects that will be set to  $I$ . This is the number of length  $r$  effects generated by taking all combinations of the defining effects.

We have included in our implementation, a computation of the aberration of the design produced. There are a number of ways to approach the computation of aberration. One way is to generate all combinations of the defining contrasts

and to count the number that have length equal to the resolution. However, for a  $2^{n-p}$  design there are  $2^p$  such combinations. This is impractical for the designs that we wish to create. For example a design for 90 factors may produce  $p = 74$  defining contrasts and hence  $2^{74}$  combinations to check.

An alternative approach for resolution  $r$  designs that we have adopted is to generate all combinations of  $r$  factors and to count those that reduce to  $I$  with the application of the defining contrasts. This requires checking  ${}^{n-p}C_r$  combinations, a feasible proposition.

In Figure 2, a recursive algorithm is shown that will perform this search. The inputs are the number of factors  $n$  and the array of contrast  $\mathbf{E}$  produced by the SEF algorithm.

```

GLOBAL VARIABLES
     $n$ ,  $r$ ,  $\mathbf{E}$ , aberration,

MAIN
    INPUTS:
         $n$       #number of factors
         $r$       #resolution
         $\mathbf{E} = [e_1, e_2, \dots, e_n]$     #defining contrasts

    aberration = 0
    current_effect =  $\emptyset$ 
    CALL RecursivelyConsiderEffects(1, 1, current_effect)

    OUTPUT: aberration

SUBROUTINE RecursivelyConsiderEffects(int depth, int value, Bitset current_effect)
    for( $i := \text{value}$ ;  $i \leq n$ ;  $i := i + 1$ )
        next_effect := current_effect  $\ominus e_i$ 
        if(depth <  $r$  )
            RecursivelyConsiderEffects(depth+1,  $i+1$ , next_effect)
        else
            if(next_effect =  $I$ )
                aberration := aberration+1
            endif
        endif
    endif
    return

```

**Fig. 2.** Recursive Algorithm for Computing Aberration

Note that the aberration could also be computed using the results of Butler [11], who showed that the problem of finding a minimum aberration design

may be reformulated in terms of the elements of the  $XX^T$  matrix, where  $X$  is the  $n \times m$  design matrix. He defines the  $k$ th moment of a design as

$$M_k = n^{-2} \sum_{p=1}^q \sum_{q=1}^n T_{pq}^k$$

where  $T = XX^T$ . Then he obtains

**Theorem 1.** *For a regular fractional factorial design of resolution III or more*

$$\begin{aligned} M_1 &= 0, M_2 = m, M_3 = 6A_3, M_4 = 24A_4 + m(3m - 2), \\ M_5 &= 120A_5 + (60m - 120)A_3, M_6 = 720A_6 + (360m - 960)A_4 + 15m(m - 1)^2 + m \end{aligned}$$

where  $A_k$  is the number of  $k$ -letter words in the defining relation, and hence the aberration of a design can be obtained from the calculated moments.

## 2.5 Selection of Defining Factors

The formal algorithm specifies that the eligible effects are selected “randomly”; the precise meaning of that term was not specified. Liao and Iyer recommend that the search should exhaust all effects of a certain length before progressing to longer ones. We interpret this strategy in the following way. If the required design has resolution  $r$  then we begin by randomly selecting an effect involving  $r$  surviving factors; all such effects have equal probability of selection. If this effect is not eligible then a sweep is performed through all effects of that length until an eligible one is found. If all effects of length  $r$  are exhausted then the process is repeated with effects of length  $r + 1$ , and so on until an eligible effect is found. We will label this strategy S1.

It might seem that biasing the selection of eligible effects to shorter strings in this way would yield a greater aberration. Hence we also implemented an alternative strategy, S2, where all effects involving the surviving factors are equally likely to be the first selected, and then a sweep is performed through all effects. The results for S2 are compared with those for S1 in the following section.

## 2.6 Performance

The algorithm is both computationally and memory intensive. All experiments were run under Linux on a dedicated 3GHz processor with 2GByte of memory.

The first experiment compared selection strategies S1 and S2. For each strategy nine designs were created with the number of factors varying from 5 to 25. All factors had 2 levels, the resolution was V and the function *EffectInSet()* used implementation *I1*. Details of the designs created are shown in Table 1.

The results show that strategy S1 produces on average, superior designs to S2. By this we mean fewer runs or the same number of runs with smaller aberration. This supports the advice on the matter given by Liao and Iyer.

The designs with the minimum aberration in Table 1 for 5, 10, and 15 factors are the minimum aberration designs, previously presented in the literature. For 20 factors, the minimum aberration is 32, based on checking all the designs given by Draper and Mitchell [15]. For 25 factors, 1024 runs are needed for a resolution V design, but to our knowledge the minimum aberration design has not been provided in the literature.

Table 2 compares the execution time for these runs. Again strategy S1 is superior. All the following experiments will employ strategy S1 for selection of eligible effects. (Note however that the difference between the strategies appears to evaporate for the largest designs described later.)

| Strategy S1 |      |       |      |       |      |       |      |       |      |       |
|-------------|------|-------|------|-------|------|-------|------|-------|------|-------|
| Factors $n$ | 5    |       | 10   |       | 15   |       | 20   |       | 25   |       |
| Design      | Runs | Aber. | Runs | Aber. | Runs | Aber. | Runs | Aber. | Runs | Aber. |
| 1           | 16   | 1     | 128  | 3     | 512  | 8     | 512  | 39    | 2048 | 37    |
| 2           | 16   | 1     | 128  | 4     | 512  | 9     | 1024 | 23    | 2048 | 30    |
| 3           | 16   | 1     | 128  | 3     | 256  | 16    | 1024 | 24    | 1024 | 68    |
| 4           | 16   | 1     | 128  | 4     | 256  | 17    | 1024 | 23    | 1024 | 66    |
| 5           | 16   | 1     | 128  | 4     | 256  | 15    | 512  | 39    | 2048 | 32    |
| 6           | 16   | 1     | 128  | 3     | 512  | 9     | 1024 | 21    | 2048 | 32    |
| 7           | 16   | 1     | 256  | 3     | 512  | 10    | 1024 | 21    | 1024 | 65    |
| 8           | 16   | 1     | 128  | 4     | 512  | 8     | 1024 | 21    | 2048 | 33    |
| 9           | 16   | 1     | 128  | 3     | 512  | 9     | 1024 | 20    | 2048 | 33    |
| Strategy S2 |      |       |      |       |      |       |      |       |      |       |
| Factors $n$ | 5    |       | 10   |       | 15   |       | 20   |       | 25   |       |
| Design      | Runs | Aber. | Runs | Aber. | Runs | Aber. | Runs | Aber. | Runs | Aber. |
| 1           | 16   | 1     | 128  | 4     | 512  | 8     | 1024 | 18    | 2048 | 26    |
| 2           | 16   | 1     | 128  | 3     | 512  | 9     | 1024 | 23    | 2048 | 31    |
| 3           | 16   | 1     | 128  | 3     | 512  | 9     | 1024 | 19    | 2048 | 30    |
| 4           | 16   | 1     | 128  | 3     | 512  | 7     | 1024 | 18    | 2048 | 31    |
| 5           | 16   | 1     | 128  | 3     | 512  | 6     | 1024 | 14    | 2048 | 31    |
| 6           | 16   | 1     | 128  | 4     | 512  | 6     | 1024 | 20    | 2048 | 23    |
| 7           | 16   | 1     | 256  | 1     | 512  | 7     | 1024 | 18    | 2048 | 25    |
| 8           | 16   | 1     | 128  | 3     | 512  | 9     | 1024 | 17    | 2048 | 30    |
| 9           | 16   | 1     | 128  | 4     | 512  | 7     | 1024 | 16    | 2048 | 27    |

**Table 1.** Details of nine  $2^{n-p}$  designs with resolution V for a range values of  $n$

A second experiment compared implementations *I1* and *I2* for storing sets of effects. The experiments again used various numbers of factors with 2 levels per factor and resolution V. Both implementations of the SEF algorithm were executed 9 times in each case.

| Factors | Strategy S1 |       |       | Strategy S2 |       |        |
|---------|-------------|-------|-------|-------------|-------|--------|
|         | Min.        | Max.  | Mean  | Min.        | Max.  | Mean   |
| 15      | 0.45        | 0.48  | 0.461 | 0.46        | 0.51  | 0.491  |
| 20      | 6.24        | 6.55  | 6.438 | 6.98        | 7.44  | 7.252  |
| 25      | 48.8        | 52.9  | 51.83 | 57.86       | 60.57 | 59.654 |
| 30      | 277.7       | 298.6 | 289.9 | 334.3       | 340.4 | 337.64 |

**Table 2.** Execution time (sec) for nine  $2^{n-p}$  designs for a range values of  $n$

As expected, with the same random number seeds, the two implementations produced identical designs. Figure 4a shows a log-log plot of the mean execution time,  $E$ , for the 9 designs against the number of factors  $n$ , for both  $I1$  and  $I2$ . Error bars show the range for  $E$ .

Both plots show an approximate fit to a straight line, implying power laws for the execution time. Where the graphs overlap,  $n = 30$ , the use of a hash table gives an improvement in speed of three orders of magnitude. This extends the practical limit of experiment sizes with current processors from 30 factors to 80 factors.

Our third experiment used 8 levels for each factor. Since 8 values may be indexed by 3 bits, each factor corresponds to 3 quasi-factors. Hence an experiment with say, 10 factors at 8 levels, reduces to 30 quasi-factors at 2 levels. However for a given resolution, the number of primary effects (and perhaps secondary effects) will be considerably greater than for 30 real factors at 2 levels. So execution time is considerably longer in the former case. This is demonstrated by Figure 4b which shows execution time against the number of factors for 8 levels each with resolution V. These results in particular show the need for a faster algorithm.

Again the hash table strategy (2) is clearly superior. All following implementation will use this.

### 3 A New Algorithm

The algorithm described below is a type of “divide and conquer” algorithm. We will refer to the algorithm as the Recursive Splitting Sequential Elimination of Factors (RSSEF) algorithm. It is best comprehended from a simple example.

#### 3.1 An Example

Suppose we have 8 factors, say  $A, B, C, D, E, F, G$ , and  $H$ , and seek a design with resolution III. Then the set of primary effects is  $\mathbf{P} = \{I, A, B, C, D, E, F, G, H\}$  and the set of secondary effects  $\mathbf{S}$  is the empty set. The initial set of ineligible

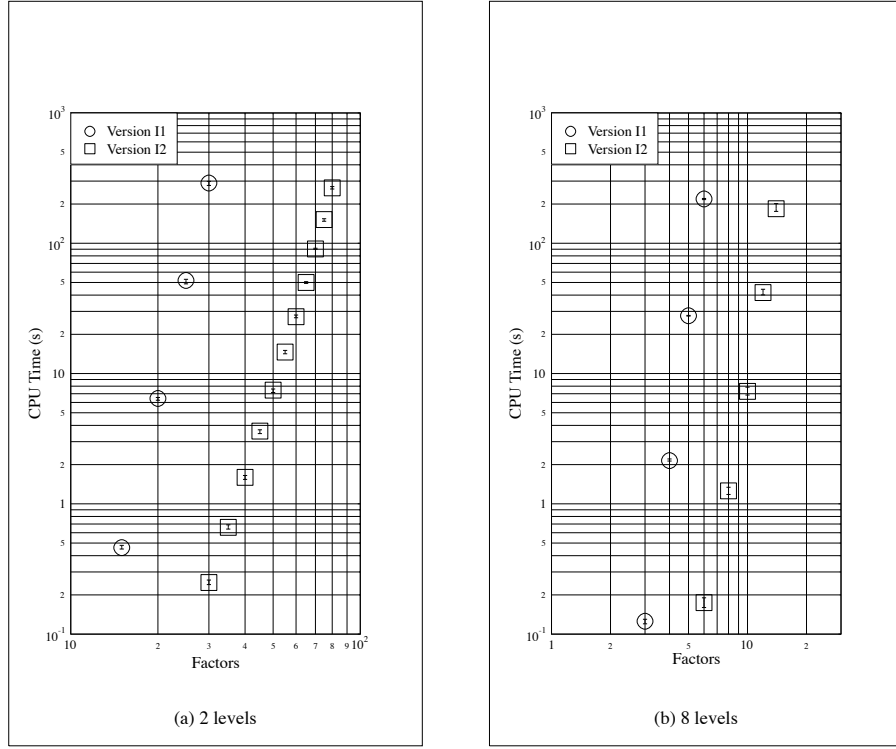


**Fig. 3.** Execution times v. no. factors for two SEF implementations  
 (a) 2 levels per factor (b) 8 levels per factor

effects would be

$$\mathbf{X} = \{I, A, B, C, D, E, F, G, H, AB, AC, \dots GH\}$$

containing 37 elements. But we do not use this set. Instead we partition the factors into two groups:  $A, B, C, D$  and  $E, F, G, H$ , applying the SEF algorithm to each group separately. For each group we reduce the primary and secondary effects, keeping only those involving the current factors.

So for the first group,  $\mathbf{P} = \{I, A, B, C, D\}$ ,  $\mathbf{S} = \emptyset$  and thus the ineligible effects are  $\mathbf{X} = \{I, A, B, C, D, AB, AC, AD, BC, BD, CD\}$ . Select an eligible effect (an element not in  $\mathbf{X}$ ), say  $ABD$ , set it to  $I$  and use it to eliminate one of the factors. Here we may eliminate  $D = AB$ . Then replacing  $D$  in  $\mathbf{X}$  and eliminating duplicates gives  $\mathbf{X} = \{I, A, B, C, AB, AC, BC, ABC\}$ . Since this is a complete set on the factors  $A, B$  and  $C$  the iterations are complete.

Similarly for the factors  $E, F, G, H$ , a defining possible contrast may be  $HEF = I$  giving  $H = EF$  and a complete design on the factors  $E, F, G$ .

Now recombine the two groups. The surviving factors are  $A, B, C, E, F, G$  and the primary effects are now  $\mathbf{P} = \{I, A, B, C, AB, E, F, G, EF\}$ . This generates the ineligible effects

$$\mathbf{X} = \{I, A, B, C, E, F, G, AB, AC, AE, AF, AG, BC, BE, BF, BG, CE, CF, CG, EF, EG, FG, ABC, ABE, ABF, ABG, AEF, BEF, CEF, EFG, ABEF\}$$

This set has 31 elements rather than the 37. The algorithm now continues with the search for an eligible effect, as in the standard algorithm. Two iteration would be needed. Suppose the defining contrasts selected are  $G = ABC$  and  $E = AC$ . Then the surviving factors are  $A, B, C, F$  and the full collection of defining contrasts is  $D = AB, H = EF = ACF, G = ABC$  and  $E = AC$ .

Note that this algorithm gives valid defining contrast because the contrast chosen in the split state are valid choices for the original SEF algorithm.

The point of the splitting is to restrict the size of the set of ineligible factors. In this case the reduction is trivial, but when the number of factors is large, significant savings can be made. The speedup achieved is shown in a later section.

For a large number of factors several splits may be advantageous. Splitting is controlled by a value *SPLITTING\_SIZE*. The set of factors is split recursively until the number of surviving factors is less than *SPLITTING\_SIZE*.

### 3.2 Splitting Strategy

In the previous example the set  $\mathbf{P}$  was split so that contiguous factors fell into the same sets. This can be a problem if there are more than two levels per factor. Consider for example the case of 3 factors, 8 levels each, with resolution 3 and splitting size 5. The quasi-factors are  $A1, A2, A3, B1, B2, B3, C1, C2, C3$  and the primary effects

$$\begin{aligned} \mathbf{P} = \{ & I, A1, A2, A1A2, A3, A1A3, A2A3, A1A2A3, \\ & B1, B2, B1B2, B3, B1B3, B2B3, B1B2B3, \\ & C1, C2, C1C2, C3, C1C3, C2C3, C1C2C3 \}. \end{aligned}$$

Splitting into contiguous groups, the first group of quasi-factors is  $A1, A2, A3, B1$  giving  $\mathbf{P} = \{I, A1, A2, A1A2, A3, A1A3, A2A3, A1A2A3, B1, \}$  Then the ineligible factors  $\mathbf{X} = \mathbf{P} \ominus \mathbf{P}$  is a complete set on the 4 quasi-factors. Then there are no reductions in the surviving quasi-factors. Similarly for the other group, so splitting will give no speedup in this case.

Suppose instead that we split the quasi-factors so that alternate factors go to different groups. The first group becomes  $A1, A3, B2, C1, C3$  and the restricted primary effects  $\mathbf{P} = \{I, A1, A3, A1A3, B2, C1, C3, C1C3, \}$  In this case  $\mathbf{X}$  is not a complete set on the quasi-factors so some defining contrasts will be found for the split problems.

In the following experiments we denote the first splitting strategy as Strategy I and the second as Strategy II.

### 3.3 The Formal Algorithm

The algorithm will be specified as a recursive procedure. First we define the routine **Restrict** that takes a set of effects,  $\mathbf{S}$ , and an alphabet of symbols,  $\mathcal{F}$ , and creates a new set of effects,  $\mathbf{T}$ , consisting of those elements of  $\mathbf{S}$  that contain only the symbols of  $\mathcal{F}$ . In terms of this the RSSEF algorithm is as shown in Figure 4.

### 3.4 Comparison of Design Quality

This section compare the designs produced by (i) the SEF algorithm, (ii) the RSSEF algorithm with splitting Strategy I, and (iii) RSSEF with splitting Strategy II. In cases (ii) and (iii) the splitting size was 10. The designs are assessed by the number of runs and the aberration. Results are shown in Table 3 for  $2^{n-p}$  designs for a range of values of  $n$  within the capacity of all algorithms namely  $n = 20, 40, 60$ .

The results show that for 20 factors there is no obvious difference in the quality of designs produced by the three algorithms. However for 60 factors RSSEF with Strategy II gives better aberration designs than with Strategy I, but inferior designs to the SEF algorithm. The following experiments with the RSSEF algorithm all use Strategy II.

### 3.5 Performance for $2^{n-p}$ Designs

An initial experiment was performed to determine the effect of the splitting size  $S$  on execution time  $E$ . Again 9 designs were produced and the mean execution time recorded. Figure 5 shows the relation between  $E$  and  $S$  for a range of values of  $n$ . Note that when the splitting size is set to the number of factors, no splitting occurs and the algorithm reverts to the SEF algorithm. This gives the final point for each value of  $n$ . Apart from these final values, the execution time is little affected by the splitting size. In each case use of splitting size 10 will be near optimal.

Figure 6 compares the execution times for the RSSEF algorithm (with splitting size 10) with those for the SEF algorithm. The new algorithm shows a speedup of about an order of magnitude for  $n \geq 70$ .

Note that for such large designs as are considered here the computation time for the aberration starts to become significant, comparable to that for the RSSEF algorithm. These times are also shown in Figure 6. To avoid confusion with data points for aberration colliding with those for the RSSEF times, we have staggered the former.

```

INPUTS:
    the number of factors  $n$ ,
    an alphabet of  $n$  symbols,  $\mathcal{F} = \{F_1, F_2, \dots, F_n\}$ ,
    the primary effects  $\mathbf{P}$ ,
    the secondary effects  $\mathbf{S}$ ,
    the splitting size  $s$ .

1. Set  $k := |\mathcal{F}|$ .      # the number of live factors
2. Set the Boolean array  $\mathbf{L} = [L_1, L_2, \dots, L_n]$ 
   where each  $L_i = \text{true}$ .      # array of live factors
3. Set the defining contrasts  $\mathbf{D} := \mathbb{Q}$ .
4. CALL Design(0,  $\mathbf{P}$ ,  $\mathbf{S}$ ,  $\mathbf{L}$ ,  $k$ ,  $\mathcal{F}$ )
OUTPUTS:  $\mathbf{L}$ ,  $\mathbf{D}$ .

SUBROUTINE Design( $\mathbf{U}$ ,  $\mathbf{V}$ ,  $\mathbf{L}$ ,  $\mathcal{A}$ )
11. Set  $k := |\mathcal{A}|$ 
12. If  $k > s$  then
     $k' = \text{floor}(|\mathcal{A}|/2)$ 
     $k'' := k - k'$ 
    split  $\mathcal{A}$  into two alphabets  $\mathcal{A}'$  and  $\mathcal{A}''$ 
    set  $k' := |\mathcal{A}'|$ ,  $k'' := |\mathcal{A}''|$ 
    CALL Restrict( $\mathbf{U}$ ,  $\mathcal{A}'$ ,  $\mathbf{U}'$ )
    CALL Restrict( $\mathbf{V}$ ,  $\mathcal{A}'$ ,  $\mathbf{V}'$ )
    CALL Design( $\mathbf{U}'$ ,  $\mathbf{V}'$ ,  $\mathbf{L}$ ,  $\mathcal{A}'$ )
    CALL Restrict( $\mathbf{U}$ ,  $\mathcal{A}''$ ,  $\mathbf{U}''$ )
    CALL Restrict( $\mathbf{V}$ ,  $\mathcal{A}''$ ,  $\mathbf{V}''$ )
    CALL Design( $\mathbf{U}''$ ,  $\mathbf{V}''$ ,  $\mathbf{L}$ ,  $\mathcal{A}''$ )
    for each  $d \in \mathbf{D}$  set  $\mathbf{U} := d \oplus \mathbf{U}$ , set  $\mathbf{V} := d \oplus \mathbf{V}$ 
endif
13. Compute the ineligible effects  $\mathbf{X} := (\mathbf{U} \oplus \mathbf{U}) \cup (\mathbf{V} \oplus \mathbf{V})$ 
14. If  $|\mathbf{X}| = 2^k$  then return
15. Select an effect  $p \in 2^{\mathcal{A}}, p \notin \mathbf{X}$ .
16. Select a factor  $F_i \in p$ 
17. Set  $\mathbf{Y} := \mathbb{Q}$ 
18. For each  $x \in \mathbf{X}$ ,
    if  $F_i \in x$ 
        compute  $q = p \oplus x$ 
        if  $q \notin \mathbf{Y}$ 
            add  $q$  to  $\mathbf{Y}$ 
    else
        add  $x$  to  $\mathbf{Y}$ 
19. Set  $\mathbf{X} \leftarrow \mathbf{Y}$ 
20. For each  $e \in \mathbf{E}$ ,
    if  $F_i \in e$ 
        set  $e := p \oplus e$ 
21. Delete  $F_i$  from  $\mathcal{F}$ , set  $L_i := \text{false}$  and  $k := k - 1$ .
22. Go to 14.

```

**Fig. 4.** The RSSEF Algorithm

|             | SEF   |       | RSSEF with<br>splitting strategy I |       | RSSEF with<br>splitting strategy II |       |
|-------------|-------|-------|------------------------------------|-------|-------------------------------------|-------|
| Factors $n$ | Runs  | Aber. | Runs                               | Aber. | Runs                                | Aber. |
| 20          | 512   | 39    | 1024                               | 22    | 512                                 | 37    |
|             | 1024  | 23    | 1024                               | 22    | 1024                                | 19    |
|             | 1024  | 24    | 1024                               | 25    | 1024                                | 21    |
|             | 1024  | 23    | 512                                | 41    | 1024                                | 18    |
|             | 512   | 39    | 1024                               | 23    | 1024                                | 22    |
|             | 1024  | 21    | 1024                               | 21    | 1024                                | 25    |
|             | 1024  | 21    | 1024                               | 19    | 1024                                | 23    |
|             | 1024  | 21    | 1024                               | 21    | 1024                                | 21    |
|             | 1024  | 20    | 1024                               | 22    | 1024                                | 22    |
|             |       |       |                                    |       |                                     |       |
| 40          | 8192  | 90    | 8192                               | 115   | 8192                                | 126   |
|             | 8192  | 110   | 8192                               | 118   | 8192                                | 104   |
|             | 8192  | 102   | 8192                               | 117   | 8192                                | 111   |
|             | 8192  | 101   | 8192                               | 108   | 8192                                | 127   |
|             | 8192  | 89    | 8192                               | 135   | 8192                                | 118   |
|             | 8192  | 104   | 8192                               | 112   | 8192                                | 131   |
|             | 8192  | 100   | 8192                               | 117   | 4096                                | 175   |
|             | 8192  | 98    | 8192                               | 114   | 8192                                | 133   |
|             | 8192  | 111   | 8192                               | 118   | 8192                                | 102   |
| 60          | 32768 | 194   | 32768                              | 305   | 32768                               | 271   |
|             | 32768 | 219   | 32768                              | 300   | 32768                               | 250   |
|             | 32768 | 221   | 32768                              | 275   | 32768                               | 262   |
|             | 32768 | 184   | 32768                              | 299   | 32768                               | 290   |
|             | 32768 | 208   | 32768                              | 325   | 32768                               | 249   |
|             | 32768 | 212   | 32768                              | 280   | 32768                               | 270   |
|             | 32768 | 237   | 32768                              | 333   | 32768                               | 229   |
|             | 32768 | 199   | 32768                              | 276   | 32768                               | 261   |
|             | 32768 | 208   | 32768                              | 325   | 32768                               | 269   |

**Table 3.** Details of nine  $2^{n-m}$  designs for a range values of  $n$

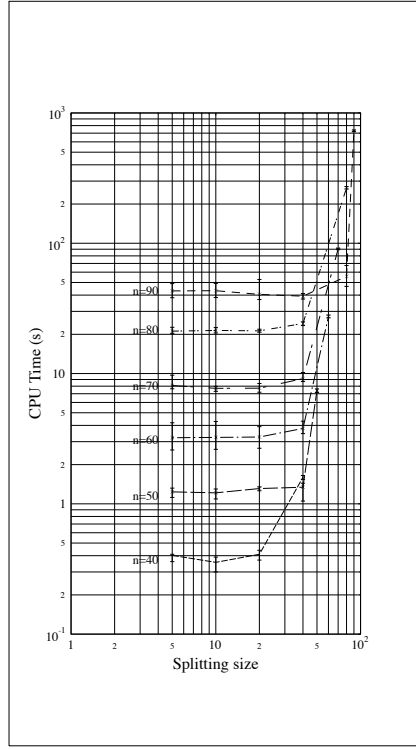


**Fig. 5.** Execution times of  $2^{n-p}$  RSSEF designs v. splitting size for various  $n$

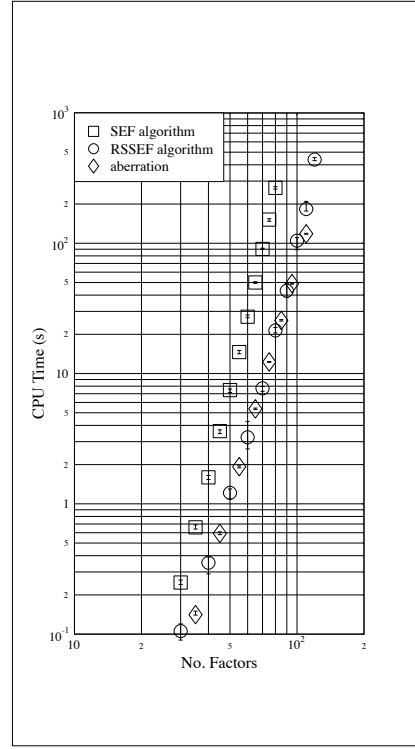


**Fig. 6.** Execution times of  $2^{n-p}$  designs v.  $n$

### 3.6 Performance for $8^{n-p}$ Designs

The experiments of Section 3.5 were repeated with eight levels per factor. Figure 7 compares the speed of the RSSEF algorithm with that of SEF using a splitting size of 10. As for the  $2^{n-m}$  designs, the speedup is approximately an order of magnitude. Computation time for the aberration is also shown.

### 3.7 Concurrency

When the factors are split into two groups in the RSSEF algorithm, the resulting two design are independent and may hence be performed concurrently. However a look at the time spent in various parts of the algorithm shows that the great majority of the execution time is spent in the search for eligible effects after the final recombination of the split groups. Hence there appears to be very little speedup possible by this strategy and we have not pursued the matter further.

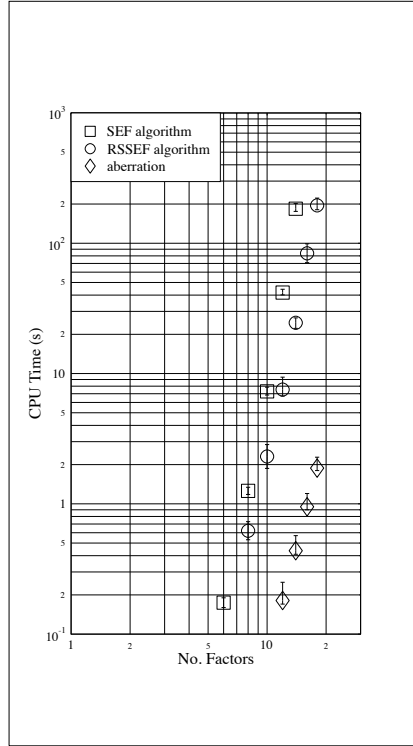


**Fig. 7.** Execution times of  $8^{n-p}$  designs for v.  $n$

## 4 Discussion

Experiments using computer models enable experiments with many more runs than are customary in physical experiments. Current algorithms for fractional factorial experimental design are too slow for some computer experiments that are now feasible. The fastest general purpose method, the SEF algorithm of Liao and Iyer, when optimized for speed, is limited to about 80 two level factors with current processors. The variant that we propose here, the RSSEF algorithm, gives designs with a somewhat inferior aberration but shows a speedup of approximately one order of magnitude for large designs. This extends the scope of designs to 120 two-level factors.

A compromise between the quality of SEF and the speed of RSSEF would use the former for smaller designs and the latter where the SEF method becomes impractical.

In addition to the regular two-level designs examined in this paper, there may be advantages to using non-regular designs [see, for example, 19] or Latin

Hypercube designs [see, for example, 20], which require less runs, but are more difficult to analyse.

## Bibliography

- [1] Box G. E. P., J. S. Hunter, and W. G. Hunter. *Statistics for Experimenters*. Wiley, second edition, 2005.
- [2] Abramson D., R. Sasic, J. Giddy, and B. Hall. Nimrod: A tool for performing parametrised simulations using distributed workstations. In *The 4th IEEE Symposium on High Performance Distributed Computing*, Virginia, August 1995.
- [3] D. Abramson, J. Giddy, and L. Kotler. High performance parametric modeling with Nimrod/G: Killer application for the global grid? In *International Parallel and Distributed Processing Symposium (IPDPS)*, May 2000.
- [4] [http://www.cudi.edu.mx/noticias/enero\\_2004/Supercomputing\\_2003.html](http://www.cudi.edu.mx/noticias/enero_2004/Supercomputing_2003.html), accessed 3 August 2005.
- [5] Sudholt W. et al. Applying grid computing to the parameter sweep of a group difference pseudopotential. In M Bubak et al., editor, *Proceedings of the 4th International Conference on Computational Science [ICCS 2004]*, volume 3036 of *Lecture Notes in Computer Science*, Springer-Verlag, 2004.
- [6] Addelman S. Orthogonal main-effect plans for asymmetrical factorial experiments. *Technometrics*, 4:21–46, 1962.
- [7] Fries A. and W. G. Hunter. Minimum aberration  $2^{k-p}$  designs. *Technometrics*, 22:601–608, 1980.
- [8] Franklin M. F. Constructing tables of minimum aberration  $p^{n-m}$  designs. *Technometrics*, 26:225–232, 1984.
- [9] Chen J., Sun D. X., and Wu C. F. J. A catalogue of two-level and three-level fractional factorial designs with small runs. *International Statistical Review*, 61:131–145, 1993.
- [10] Block R. M. and Mee R. W. Resolution IV designs with 128 runs. *Journal of Quality Technology*, 37:282–293, 2005.
- [11] N. A. Butler. Some theory for construction minimum aberration fractional factorial designs. *Biometrika*, 90:233–238, 2003.
- [12] Chen H. H. and Cheng C. S. Doubling and projection: A method of construction two-level designs of resolution IV. *The Annals of Statistics*, 34:546–558, 2006.
- [13] N.R. Draper and T.B. Mitchell. The construction of saturated  $2_r^{k-p}$  designs. *The Annals of Mathematical Statistics*, 38:1110–1126, 1967.
- [14] N.R. Draper and T.B. Mitchell. Construction of the set of 256-run designs of resolution  $\geq 5$  and the set of even 512-run designs of resolution  $\geq 6$  with special reference to the unique saturated designs. *The Annals of Mathematical Statistics*, 39:246–255, 1967.
- [15] N.R. Draper and T.B. Mitchell. Construction of the set of 512-run designs of resolution  $\geq 5$  and the set of even 1024-run designs of resolution  $\geq 6$ . *The Annals of Mathematical Statistics*, 41:876–887, 1970.

- [16] Franklin M. F. and R. A. Bailey. Selection of defining contrasts and confounding effects in two-level experiments. *Applied Statistics*, 26:321–326, 1977.
- [17] Liao C. T. and H. K. Iyer. A stochastic algorithm for selecting [sic] of defining contrasts in two-level experiments. *Biometrical Journal*, 6:671–678, 1999.
- [18] <http://www.MessageLab.infotech.monash.edu.au/Nimrod>, accessed 3 August 2007.
- [19] Mee R. W. Efficient two-level designs for estimating all main effects and two-factor interactions. *Journal of Quality Technology*, 36:400–412, 2004.
- [20] Santner T. J., Williams B. J., and Notz W. *The Design and Analysis of Computer Experiments*. Springer, 2003.