M. Tech (Computer Science) Dissertation Series

STUDY OF THREE COMBINATORIAL PROBLEMS

A dissertation submitted in partial fulfillment of the requirements for the M. Tech (Computer Science) degree of the Indian Statistical Institute

By

PALASH SARKAR

Under the supervision of

Dr. Bimal Kr. Roy

Indian Statistical Instutute

CERTIFICATE

This is to certify that the work described in the dissertation entitled "Study of 3 Combinatorial Problems" has been undertaken by Palash Sarkar under my guidance and supervision. The dissertation is found worthy of acceptance for the award of the Degree of Master of Technology in Computer Science.

Bimal Roy

(Dr. Bimal Kumar Roy)

Dated July , 1993.

Indian Statistical Institute.

Calcutta.

ACKNOWLEDGEMENT

This work would not have been possible without the guidance and constant help of my guide Dr. Bimal Kumar Roy.He provided encouragement and optimism when things got stuck and showed me how to tackle problems with enthusiasm and vigour.

I would also like to take this oppurtunity to thank Dr. Bikas Sinha, of Stat/Math Division of I.S.I, Calcutta, who took the trouble to personally introduce me to the first problem of this work.

I must also thank Mr. Indranil Ojha, for the help and support he provided during the course of my work. Mr. Ojha had been working under Dr. Roy for his M.Tech thesis over the same period.

Finally, I thank the staff of CSSC for providing such a wonderful computing environment and also for the warm help and cooperation they extended during the course of this work.

PALASH SARKAR

DATED : JULY '93

INDIAN STATISTICAL INSTITUTE

CALCUTTA

CONTENTS

CHAPTER 1

AN EXTREMAL PROBLEM IN THE CONSTRUCTION OF BINARY MATRICES

1. Introduction

The problem is to construct a set of n tests, to test a set of v items. The items can be combined, and a test applied on the combination. A test reports positive if any of the items in the combination is defective. If all the items which were combined together are good then a test conducted on the combination gives a negative result. The set up is assumed to be non-adaptive, in the sense that all the tests are conducted simultaneously, and the results of all the tests are used to discover the defective items. The purpose is to minimise the number of tests required, as it is always possible to individually test the items requiring v tests.

A set of tests to achieve the above purpose is called a design. Such designs have been considered in [1],[2]. In [1] a property of a design called detecting power is defined. A design is said to have detecting power t, or DP(t) if it can correctly detect the presence of upto t defective items. In the following we consider the problem for designs having DP(2). In [2], it is shown that minimizing the number of tests is equivalent to maximizing the number of rows of a binary matrix with n columns, having a certain property.

Here we take an algorithmic approach to the problem. The naive algorithm is outlined, and is shown to be inefficient. The algorithm has been implemented,and becomes impractical to run even for 7 columns. A bound on the maximum number of rows is established. A sufficient condition for a binary matrix to have DP(2) is used to get a heuristic for the problem. A counting argument is used to get a result which puts a bound on the number of rows that can be included in a matrix, when two other rows are present.

2. Definitions

Define the union of two binary row vectors r and q as a row vector p such that a component of p is 1, iff, the same component of at least one of the vectors r and q is 1.

A binary matrix (on n columns), i.e, a matrix with entries from the set {0,1} is said to have DP(2), if it possesses the following properties.

1. The null vector is not a row of the matrix.
2. The rows of the matrix are distinct.
3. Let q , r , s , t be any four row vectors of the matrix, such that $\{q, r\} \neq \{s, t\}$. Then the union of q and r is distinct from the union of s and t.

Let f(n) denote the maximum possible number of rows of a binary matrix on n columns having DP(2).

Two row vectors are said to share an 1 if there exists an column such that both have an 1 in that column.

If a row vector r shares all its 1's with a row vector s then we say that r is a subset of s.

Note 1: The identity matrix (n X n) has DP(2). This corrosponds to a design where each item is tested seperately.Hence n items can be tested by n tests.

Note 2: If a matrix has DP(2), then a matrix obtained by interchanging two rows, or interchanging two columns, also have DP(2).

Note 3: $f(n+m) \geqslant f(n) + f(m)$. Since given designs for n and m it is possible to combine them as follows to get a design for $f(n+m)$.

$$\begin{array}{c|c} [n \times f(n)]^T & 0 \\ \hline 0 & [m \times f(m)]^T \end{array}$$

Where 0 denotes the null matrix. Also note that $f(1) = 1$.

3. Results and Algorithms

The naive algorithm uses a recursive procedure to implement the following idea. A matrix with 1 row has DP(2). Having a matrix with DP(2), augment the matrix with a row not considered previously. Test the resulting matrix for violation of DP(2) property. If the resulting matrix has DP(2), then try to augment furthur, else try to augment the old matrix with a new row. If all possible choices of row has been considered, without increasing the size of the matrix, then we have a maximal solution. If furthur there does not exist any choice for any of the rows such that the matrix can be augmented, then we have a maximum solution. In implementing the algorithm the rows have been considered to be binary numbers.

It is clear that the search space for the above algorithm consists of all possible n bit binary numbers. And to ascertain that a solution is maximum, the algorithm has to consider all possible i selections from this search space, where i varies from 1 to f(n). This clearly requires exponential time and makes the algorithm a grossly inefficient one.As a result it becomes impractical to run the algorithm even for $n = 7$. However, for $n = 6$, a design having 8 rows can be constructed, and the value of $f(6)$ is established as 8. This corroborates with a catalogue given in [2].

The naive algorithm has also been implemented with bounds on the number of 1's in a row. This does not improve the efficiency of the algorithm significantly enough to yield better results.

Lemma 1: for $n > 1$, $f(n-1) + 1 \leqslant f(n) \leqslant 2f(n-1)$

Proof: The left hand inequality follows from note 3 of section 2 and the fact that $n = (n-1) + 1$.

For the right hand it is sufficient to show that any n column matrix having DP(2) cannot be such that a column has more than $f(n-1)$ 1's or more than $f(n-1)$ 0's, so that the maximum possible size of any column is $2f(n-1)$. To see this suppose that a column has more than $f(n-1)$ 0's. By note 2 of section 2, without loss of generality we can assume that all the 0's in the column occur before any of the 1's in the column. Also we can assume that this column is the first column.Let the number of 0's in the column be p. Then if we leave out the first column,and all rows after the p-th row,we have a matrix which satisfies DP(2), and has $n-1$ columns. Thus $f(n-1) \geq p$. But by assumtion $p>f(n-1)$. This is a contradiction. Hence no column can have more than $f(n-1)$ 0's. Similarly we can show that no column can have more than $f(n-1)$ 1's.

Thus the proof of the lemma follows.

Lemma 2: If an n column binary matrix having distinct rows, none of which is the null row vector, has the following property, then it also possesses DP(2).

Suppose there are two rows having i and j 1's.Then if i be less than or equal to j, the rows cannot share more than $\left\lfloor\frac{i-1}{2}\right\rfloor$ 1's.

Proof: First note that 1 and 2 of property DP(2) is trivially satisfied. So we have to show that under the above assumption the matrix also satisfies 3 of property DP(2).

Let r be any row of the matrix, and let p , q be any two other distinct rows of the matrix whose union we denote by x. Then r cannot share all of its 1's with x, i.e, r cannot be a subset of x. Let i,j,k be the number of 1's in r,p and q respectively. Now by the above property, r shares at most $\left\lfloor\frac{i-1}{2}\right\rfloor$ 1's with p and also at most $\left\lfloor\frac{i-1}{2}\right\rfloor$ 1's with q. So r shares at most $(i-1)$ 1's with x. Hence r cannot be a subset of x. Now if s be any other row of the matrix, the union of r and s cannot be a subset of x, and therefore cannot be equal to x. Hence the matrix satisfies 3 of the DP(2) property.

Thus the matrix satisfies property DP(2).

Note 1: If we consider a graph whose nodes represent binary row vectors of size n, and two nodes are connected if they satisfy the property of the above lemma, then the row vectors represented by a clique in the graph satisfies DP(2).

An algorithm using the above idea has been implemented. The number of 1's in any row is bounded and is supplied as an input to the algorithm. To find a clique, a heuristic has been used, since any exact algorithm is bound to require exponential time, as the decision version of the problem is known to be NP-complete. However using the heuristic it has not been possible to get solutions better than the identity matrix. It seems that a better implementation of the above idea could yield better results.

Lemma 3: Suppose two rows p and q share k 1's and ℓ 0's. Then there are $2^{k+\ell}$ row vectors such that including any of them alongwith both p and q in a matrix will violate property DP(2).

Proof: Consider a row vector r such that it has a 1 in a column if exactly one of p and q have 1 in that column. Else it can have either 0 or 1. Thus there are $2^{k+\ell}$ possible choices for r. Let x denote r union p and y denote r union q. Then x is equal to y, because if both p and q have an 1 in a column then so does x and y. If both have 0 in a column then x and y both have the digit that r has in that column. If exactly one of p and q have an 1 in some column then both x and y have an 1 in that column. Hence equality of x and y follows. Therefore we cannot include p,q and r in a matrix without violating the DP(2) property.

Hence the proof of the lemma follows.

4. References

1. Saha, G.M. and Sinha, B.k. (1980): Some combinatorial aspects of designs useful in group testing experiments.Tech. report, Indian Statistical Institute, Calcutta.

2. --------------- (1981): Some Combinatorial aspects of designs useful in group testing experiments - an addendum. Tech Report,I.S.I,Calcutta.

CHAPTER 2

CONSTRUCTION OF SYMMETRIC BALANCED SQUARES

1. Introduction

Construction of symmetric balanced squares are considered. The square is an array of size n X n with entries from the set $\{1,...,v\}$ having v elements. The square is to be symmetric satisfying the following balance conditions.

1. Each of the v elements is present in each row either $\lfloor \frac{n}{v} \rfloor$ or $\lceil \frac{n}{v} \rceil$ times.
2. Each of the v elements is present $\lfloor \frac{n^2}{v} \rfloor$ or $\lceil \frac{n^2}{v} \rceil$ times in the entire array.

A symmetric balanced square described above is abbreviated SBS(n,v). An NBS(n,v) is a nearly balanced symmetric square n X n filled with v elements such that the first balance condition is satisfied and the second is modified as follows. An unit may have frequency f-1,f, or f+1 where f is $\lfloor \frac{n^2}{v} \rfloor$ or $\lceil \frac{n^2}{v} \rceil$ according as which is odd. Such squares are considered only in the case that SBS(n,v) does not exist.

The feasibility conditions for the existence of such squares have been derived in [1]. There they have also given construction methods for squares whenever they are feasible. The heart of the algorithm is the case when $n < v$. When $n = v$ a symmetric latin square suffices. When $n > v$, then n is written as $n = qv + r$, $0<r<v$, and using SBS(v,v) and SBS(r,v) if such a square is feasible or a near balanced square NBS(r,v) if SBS(r,v) is not feasible, they have shown how to construct SBS(n,v).

When $n < v$, and n is odd they use a 1-factorization of K_{n+1} to get an $O(n^2)$ algorithm to construct a SBS(n,v), if feasible. However when $n < v$, and n is even they use Hall's matching theorem alongwith Fulkerson's Network Feasibility Flow Theorem to show the existence of feasible SBS(n,v)'s. The construction method requires time $O(n^3)$ since finding a max flow in a network with n vertices require time $O(n^3)$.

Here their method for odd values of n is adapted to get an $O(n^2)$ algorithm for even values of n. The technique is to properly use a 1-factorization of K_{n+2} when n is even.

2. Definitions

Let us define the frequency of a element to be the number of times it occurs in the square. By the second balance condition given above, this is $\lfloor \frac{n^2}{v} \rfloor$ or $\lceil \frac{n^2}{v} \rceil$. Hence it can be either odd or even.

The feasibilty conditions for the existnce of such a square as derived in [1] are.

1. The number of elements with odd frequency must not exceed n. This is because an element with odd frequency must occur at least once in the diagonal, for the square to be symmetric.
2. The total number of elements v, to be placed in the square must be less than or equal to n(n+1)/2, (where n is the size of the square) since this is the maximum number of elements that can be accomodated in a symmetric square.

Let n be even, then a spanning subgraph of K_n consisting of n/2 vertex disjoint edges, is called an 1-factor of K_n. A decomposition of K_n into (n-1) disjoint 1-factors is defined as a 1-factorization of K_n.

One method of obtaining a 1-factorization which will be used in subsequent sections is as follows.

Let the vertices of K_{2n} be denoted by $0,\ldots,2n-2,\infty$. Define for

$i = 1,\ldots,2n-1$, the set of edges

$$S_i = (\infty\ i-1) \quad (i-1+j\ \ i-1-j\ ;\ j = 1,\ldots,n-1)$$

where each of the vertices $i-1+j$ and $i-1-j$ is expressed as one of the numbers $0,\ldots,2n-2$ modulo $2n-1$. Clearly the collection $\{S_i : i=1,\ldots,2n-1\}$ is a partition of edges of K_{2n}, and the sum of subgraphs G_i induced by S_i is a 1-factorization of K_{2n}.

The above partition of K_{2n} will be considered to be organized as a table as shown below.

Ex : 1-factorization of K_8.

$(\infty\ 0)$	(6 1)	(5 2)	(4 3)
$(\infty\ 1)$	(0 2)	(6 3)	(5 4)
$(\infty\ 2)$	(1 3)	(0 4)	(6 5)
$(\infty\ 3)$	(2 4)	(1 5)	(0 6)
$(\infty\ 4)$	(3 5)	(2 6)	(1 0)
$(\infty\ 5)$	(4 6)	(3 0)	(2 1)
$(\infty\ 6)$	(5 0)	(4 1)	(3 2)

A near 1-factorization of K_n when n is even, is constructed from a 1-factorization of K_{n+2} as follows. The vertex labelled ∞ is left out leaving a column of isolated vertices as the first column of the factorization. From the resulting table all arcs having vertex n as one of the vertices are removed. Also the isolated vertex n is removed. The resulting table will be called a near 1-factorization of K_n and is used as a reference table for the construction of the squares.

Ex: A near 1-factorization of K_8.

0	$\boxed{(8\ 1)}$	(7 2)	(6 3)	(5 4)
1	(0 2)	$\boxed{(8\ 3)}$	(7 4)	(6 5)
2	(1 3)	(0 4)	$\boxed{(8\ 5)}$	(7 6)
3	(2 4)	(1 5)	(0 6)	$\boxed{(8\ 7)}$
4	(3 5)	(2 6)	(1 7)	$\boxed{(0\ 8)}$
5	(4 6)	(3 7)	$\boxed{(2\ 8)}$	(1 0)
6	(5 7)	$\boxed{(4\ 8)}$	(3 0)	(2 1)
7	$\boxed{(6\ 8)}$	(5 0)	(4 1)	(3 2)
$\boxed{8}$	(7 0)	(6 1)	(5 2)	(4 3)

Fig 1: The boxed portions indicate left out entries.

Note 1: In the table for a near 1-factorization of K_n, the last row contains $n/2$ arcs. Each of the other rows contain $(n-2)/2$ arcs alongwith an isolated vertex which occur as the leftmost element of a row.

Note 2: Any arc $(i\ j)$ in the factorization represent two cells of a square array, namely $(i\ j)$ and $(j\ i)$. The vertex i represent the cell $(i\ i)$. We say an element is placed in the arc $(i\ j)$, iff, the corrosponding cells in the array contain the element.

Note 3: We say a set of m arcs to be consecutive if either they are consecutive arcs in the same row of the array of arcs, or are divided between rows as consecutive arcs at the left most end of a row and the right most end of a row.

Note 4: It is easy to note that any consecutive set of $(n-2)/2$ arcs in the near 1-factorization of K contains $(n-2)$ distinct vertices of the corrosponding complete graph.

Note 5: Henceforth the following things will be assumed.

1. The isolated vertex i will be referred to as arc $(i\ i)$.

2. The near 1-factorization of K_n described above will be called the reference table.

3. n will be taken to be even and v to be greater than n.

4. f will be taken to be $\left\lfloor \frac{n^2}{v} \right\rfloor$.

3. Algorithms

Case 1: $f = n-1$.

Let $v = n+x$, where $x > 0$.

Now $n^2 = (n-1)(n+x) + [x-(x-1)n]$

$f = n-1$, implies $0 < x - (x-1)n < v$, viz, $0 < x < 1 + 1/(n-1)$.

x being an integer, we have $x = 1$.

So, an SBS(n,n+1) has to be constructed. This is easily achieved by considering a symmetric latin square of order n+1 and then by dropping any of the n+1 rows and the same column from the square,viz if we drop i-th row then i-th column get dropped.

The time taken is $O(n^2)$.

Case 2: $f = n-2$.

$n^2 = (n-2)(n+x) + [2x - (x-2)n]$, where $v = n+x$, $x > 0$.

Now $f = n-2$, implies $2 < x < 2 + x/(n-2)$

Thus for $n > 6$, $x = 2$, and $v = n+2$.

for $n = 4$, $x = 2,3,4$ and $v = 6,7,8$.

for $n = 6$, $x = 2,3$ and $v = 8,9$.

Let us first give SBS(4,7), SBS(4,8), SBS(6,9).

SBS(4,7)

1	4	5	6
4	2	1	7
5	1	3	2
6	7	2	3

SBS(4,8)

1	3	4	6
3	1	5	7
4	5	2	8
6	7	8	2

SBS(6,9)

9	5	2	6	3	8
5	8	6	3	7	4
2	6	8	7	4	1
6	3	7	9	1	5
3	7	4	1	9	2
8	4	1	5	2	9

For $n \geq 4$, and $v = n+2$

$n^2 = 4(n-1) + (n-2)(n-2)$

Thus there are 4 elements with frequency $(n-1)$ and $(n-2)$

elements with frequency (n-2). The following algorithm constructs an SBS(n,n+2) ($n > 4$) correctly.

Algorithm 1:

Step 1: Place element 1 in the leftmost arc of the last row, i.e, arc (n-1 0). Also place 1 in arcs (i i), $1 < i < n-3$.

Step 2: Place element 2 in the arcs of row 1. Place element 3 in the arcs of row n-1. Place element 4 in the arcs of row n.

Step 3: Place an element in the unused arcs of each of the other rows. Also place an element in the remaining (n-2)/2 arcs of the last row.

Lemma 1 : For $n > 4$, algorithm 1 correctly constructs an SBS(n,n+2) in $O(n^2)$.

Proof : The correctness follows from the note 2 and note 4 of sec 2, the balance condition on f being obvious.

The complexity follows from the fact that the reference table has n(n+1)/2 entries organized as vertices and arcs and each entry is accessed exactly once in the construction process.

Ex: $n = 8$, $v = 10$, $f = 6$.

So 4 elements have frequency 7 and 6 elements have 8.

Using the reference table for n = 8 shown in fig-1 we get the following SBS(8,10).

2	9	5	3	6	4	7	1
	1	3	6	4	7	10	8
		1	4	7	10	8	2
			1	10	8	2	9
				1	2	9	5
					1	5	3
						3	6
							4

Case 3: $f = n-3$.

Let $v = n+x$, $x > 0$, then $n^2 = (n-3)(n+x) + 3n - xn + 3n$.

Now $f = n-3$ implies $3 \lesssim x \lesssim 3 + 9/(n-3)$.

Thus for $n > 12$, $x = 3$ and $v = n+3$.

For $n = 4$, $v = 9,10$.

SBS(4,9)

1	4	5	6
4	2	7	8
5	7	3	9
6	8	9	3

SBS(4,10)

1	5	6	7
5	2	8	9
6	8	3	10
7	9	10	4

For $n = 6$, $v = 10,11,12$

But SBS(6,11), and SBS(6,12) are not feasible.

SBS(6,10)

10	8	6	9	7	1
8	10	9	7	2	4
6	9	1	3	4	5
9	7	3	2	5	8
7	2	4	5	3	6
1	4	5	8	6	10

For $n = 8$, $v = 11,12$

SBS(8,12)

2	7	3	9	4	11	5	12
7	3	9	4	10	5	12	6
3	9	4	10	5	12	6	1
9	4	10	6	11	7	1	8
4	10	5	11	7	1	8	2
11	5	12	7	1	8	2	9
5	12	6	1	8	2	10	3
12	6	1	8	2	9	3	11

The case SBS(8,11) is described below.

For $n = 10$, $v = 13,14$. But SBS(10,14) is not feasible.

For $n = 12$, $v = 15,16$. But SBS(12,16) is not feasible.

For $n \geq 8$, $v = n+3$ implies,

$n^2 = 9(n-2) + (n-6)(n-3)$.

Thus 9 elements have frequency $(n-2)$ and $(n-6)$ elements have frequency $(n-3)$.

For $n \geq 8$, the following algorithm will construct an SBS(n,n+3).

Algorithm 2:

Step 1: Place elements 1,2,3 in the arcs of rows 1,2,3 respectively, leaving out the arcs (i i) in each row.

Step 2: Starting from the rightmost arc of 4-th row, place elements numbered 4 onwards, one by one, each in $(n-2)/2$ consecutive arcs.

Step 3: Place element $(3+n/2)$ in arc (1 1), element $(2+n/2)$ in arc (2 2), element $(1+n/2)$ in arc (0 0).

Lemma 2: The above algorithm correctly constructs an SBS(n,n+3) for $n \geq 8$ in time $O(n^2)$.

Proof: The complexity consideration is similar to Lemma 1.

The correctness is proved as follows.

First note that elements placed by step 2 have frequency either $(n-3)$ or $(n-2)$.

Now the l.c.m of $n/2$ and $(n-2)/2$ is their product. Each row contains $n/2$ arcs and we place an element in $(n-2)/2$ consecutive arcs. Thus we use up exactly $(n-2)/2$ rows to place $n/2$ elements.

Thus the total number of elements placed by the algorithm is $3 + n/2 + n/2 = n + 3$, using up $3 + (n-2)/2 + (n-2)/2 = (n+1)$, i.e, all rows of the reference table.

Also starting from the 4-th row, after placing $n/2$ elements numbered 4 to $3 + n/2$, we will have used up rows 4 to $n/2 + 2$.

The structure of rows n/2, n/2 + 1, n/2 + 2 is as follows.

n/2-1 (n/2-2   n/2)   } (2 n-4)   (1 n-3)   (0 n-2) $\boxed{(n\ n-1)}$

n/2   (n/2-1 n/2+1)   (3 n-3)} {(2 n-2)   (1 n-1) $\boxed{(0\quad n)}$

{n/2+1 $\boxed{(2\quad n)}$} {(1   0)

where the boxed arcs have been left out and the braces indicate how elements 1+n/2, 2+n/2, 3+n/2 have been placed.

Thus in step 2 element 1+n/2 have not been placed anywhere in row 0 and column 0. Element 2+n/2 have not been placed anywhere in row 2 and column 2. Element 3+n/2 have not been placed anywhere in row 1 and column 1.

So step 3 does not violate the first balance condition on the square and by note 4 of the previous section neither does step 2.

(Note that here $\left\lfloor \frac{n}{12} \right\rfloor = 0$)

Also after step 3 elements 1+n/2, 2+n/2, 3+n/2 each have frequency n-2 in the square, since after step 2 each had frequency n-3 and step 3 increases the frequency by 1.

Also note that elements 1,2,3,4,n/2+4 and n+3 each have frequency (n-2) in the square. So 9 elements have frequency (n-2) and hence (n-6) have frequency (n-3), since a total of (n+3) elements have been placed by the algorithm.

Thus the second balance condition on the square is also satisfied.

Symmetry of the square follows from note 2 of section 2.

This establishes the correctness of the algorithm.

Ex: n = 10, v = 13, f = 7

So 9 elements have frequency 8 and 4 elements have frequency 7.

0	$\boxed{(10\ \ 1)}$	(9 2)	(8 3)	(7 4)	(6 5)
1	(0 2)	$\boxed{(10\ \ 3)}$	(9 4)	(8 5)	(7 6)
2	(1 3)	(0 4)	$\boxed{(10\ \ 5)}$	(9 6)	(8 7)
3	(2 4)	(1 5)	(0 6)	$\boxed{(10\ \ 7)}$	(9 8)
4	(3 5)	(2 6)	(1 7)	(0 8)	$\boxed{(10\ \ 9)}$
5	(4 6)	(3 7)	(2 8)	(1 9)	$\boxed{(\ 0\ \ 10)}$
6	(5 7)	(4 8)	(3 9)	$\boxed{(\ 2\ \ 10)}$	(1 0)
7	(6 8)	(5 9)	$\boxed{(\ 4\ \ 10)}$	(3 0)	(2 1)
8	(7 9)	$\boxed{(\ 6\ \ 10)}$	(5 0)	(4 1)	(3 2)
9	$\boxed{(\ 8\ \ 10)}$	(7 0)	(6 1)	(5 2)	(4 3)
$\boxed{10}$	(9 0)	(8 1)	(7 2)	(6 3)	(5 4)

Fig-2 : Reference table for n = 10

Using the above reference table we get the following SBS(10,13)

```
 6    7    2    9    3   10    4   12    5   13
      8    9    3   10    4   12    5   13    6
           7   10    4   11    5   13    6    1
                5   11    6   13    7    1    8
                     6   12    7    1    8    2
                          7    1    8    2    9
                               8    2    9    3
                                   10    3   11
                                        11    4
                                             12
```

Case 4: f < n-3

If SBS(n,v) is feasible then the following algorithm constructs one.

Algorithm 3:

Step 1: Calculate the number of positions to be filled on the diagonal to account for the even parts. [An even part of an element with frequency f is f-1 if f is odd and is f if f is even]. Occupy these diagonals by necessarily placing elements with even frequency, taking care that no vertex is repeated for the element placed using the diagonals and the left most consecutive arcs of the last row.

Step 2: Place elements, one by one, starting from the right most end of the first row, in the required number of arcs, placing an element with odd frequency if it has to be placed in the arc (i i),else it is placed with even frequency. In placing elements we skip arcs (i i) that have been used up in step 1.

Lemma 3: Algorithm 3 correctly constructs a feasible SBS(n,v) with $f < n-3$ in time O(n).

Proof: Complexity follows as in lemma 1.

Let g be f or f+1, depending on which is odd and let u be the number of elementas with frequency g.

After step 1 exactly u arcs of the form (i i) are left unused. Using up these arcs in step 2, implies that exactly u elements are placed with odd frequency, i.e, with frequency g. Since elements are placed with frequencies f or f+1, and all entries of the reference table have been used up, it follows that exactly (v-u) elements have been placed with even frequency. Thus a total of v elements have been placed by the algorithm.

Also since the frequencies of the placed elements are f or f+1, the second balance condition is maintained. The first balance condition follows from note 4 of section 2. And by note 2 of section 2 the square constructed is symmetric.

Thus the algorithm is correct.

Ex: $n = 10$, $v = 17$, $f = 5$

Two elements have frequency 5 and 15 elements have frequency 6.

Using the reference table for $n = 10$ shown in fig-2 we get the following SBS(10,17).

```
  2  11   5  12   6  14   7  16   8   1
      2  12   6  14   7  16   8   1   9
          1  13   8  15   9  17  10   4
              1  15   9  17  10   3  11
                  2  17  10   3  11   5
                      2   3  12   4  13
                          2   4  13   6
                              2   5  14
                                 15   7
                                     16
```

Note that element 1 has been placed in arcs (9 0) and (8 11) as well as in diagonals (2 2) and (3 3). Also element 2 has been placed completely in the diagonal. These placings were done in step 1.

From the above discussion we get the following theorem.

Theorem 1: For n even and v > n, an SBS(n,v), if feasible, can be constructed in time O(n).

The algorithm for constructing SBS(n,v) for all values of n and v, have been implemented on the VAX 8650 system. The algorithm used has been the one discussed in [1] with modifications as described above.

4. Reference

1. Dutta T.K. and Roy B.K. Construction of Symmetric Balanced Squares, I.S.I Technical Report No ASC/93/16, May '93.

CHAPTER 3

CONSTRUCTION OF NEARLY BALANCED UNIFORM REPEATED MEASUREMENT DESIGNS

1. Introduction

In repeated measurement designs (RMD) each experimental unit is exposed to a number of treatments applied sequentially over periods. If there are p periods $0,\dots,p-1$; t treatments $0,\dots,t-1$; and n experimental units, then an RMD(t,n,p) is an $n \times p$ array, say $D = (d_{ij})$ where d_{ij} denotes the treatment assigned to the i-th unit in the j-th period $i = 1,\dots,n$ and $j = 0,\dots,p-1$. An RMD is called uniform if in each period the same number of units is assigned to each treatment and on each unit each treatment appears in the same number of periods.

The underlying model is called circular if in each unit the the residuals in the initial period are incurred from the last period. Under the circular model an RMD is called nearly balanced if $p < t$, and the collection of ordered pairs (d_{ij}, d_{ij+1}) $1 \leq i \leq n$, $0 \leq j \leq p-1$ (operation on the second suffix is modulo p), contains each ordered pair of treatments, either once or not at all. Nearly balanced uniform RMD(t,n,p) will be abbreviated to NBURMD(t,n,p).

Here construction of NBURMD(t,t,p) is considered using the method of differences and a proper difference vector. It is shown that if p be odd, then for $t = p+1$ and $t = p+2$, such vectors do not exist. In all the other cases proper difference vectors are defined.

2. Method of differences

Definition: Let G be the group Z_t. Consider the P-tuple $P : P_0 \dots P_{p-1}$, where,

1. P_i's are distinct and $P_i \in G$, $i = 0,\dots,p-1$.
2. Define $D_i = P_{i-1} - P_i$, $i = 1,\dots,p$, with $P_p = P_0$. Then D_i's are also distinct.

Then $\{P+g : g \in G\}$ arranged in t rows, forms NBURMD(t,t,p). P will be referred to as a difference vector.

Note 1: Henceforth P_p will be considered to be P_0 .

Note 2: The existence of a difference vector is a sufficient condition for the existence of NBURMD(t,t,p).

Note 3: Differences vectors cannot exist for $t = p$.

Note 4: After constructing a difference vector, it is possible to construct a NBURMD(t,t,p) in $p(t-1)$ number of steps. The constructions given for difference vectors, whenever they exist, require p number of steps. Thus whenever NBURMD(t,t,p) can be constructed by the method of differences, the algorithm requires pt number of steps.

Theorem 1: If p be odd, then for $t = p+1$ and $t = p+2$, difference vectors do not exist.

Proof: Let us first derive a necessary condition for the existence of a difference vector.

Let $P_0, \dots, P_{p-1}$ be any difference vector.

Then, $D_i = P_{i-1} - P_i \quad 1 \le i \le p$.

Summing over i from 1 to p, we get,

$$\sum_{i=1}^{p} D_i = \sum_{i=1}^{p} P_{i-1} - \sum_{i=1}^{p} P_i$$

$$= 0 \quad \text{, since } P_p = P_0 .$$

$\therefore \sum_{i=1}^{p} D_i = 0$ is a necessary condition for the existence of the corrosponding difference vector.

We will show that this condition is violated if $t = p+1$ and $t = p+2$.

Case $t = p+1$: Then $D_i \in \{1, \dots, p\}$, $1 \le i \le p$, since D_i cannot be zero for any i.

$$\therefore \sum_{i=1}^{p} D_i = \sum_{i=1}^{p} i$$

$$= \frac{p(p+1)}{2} \not\equiv 0 \bmod (p+1).$$

Since p is odd, $\frac{p}{2}$ is not an integer.

$\therefore$ Difference vectors cannot exist for $t = p+1$.

Case $t = p+2$: Then, $D_i \in \{1, \dots, p+1\}$. $1 \le i \le p$.

$$\therefore \sum_{i=1}^{p} D_i = \sum_{i=1}^{p} i + (p+1) - r', \text{ where } 1 \le r' \le p+1.$$

$$= \sum_{i=1}^{p+1} i + r, \text{ where } r = p+2-r' \text{ \& } 1 \le r \le p+1.$$

$$= \frac{(p+1)(p+2)}{2} + r.$$

$$\not\equiv 0 \bmod (p+2).$$

Since, p is odd, $(p+1)$ is even and so $(p+2)$ divides $\frac{(p+1)(p+2)}{2}$ but $(p+2) \nmid r$, since $1 \le r \le p+1$.

$\therefore$ Difference vectors cannot exist for $t = p+2$.

Hence the theorem.

3. Construction of Difference Vectors.

Lemma 1: Let $p = 2r$, with r even and $1<r<n$ if $t=2n$ and $1<r\leq n$ if $t=2n+1$. Then a corresponding difference vector exists.

Proof: Define a p-tuple $P: P_0 \ldots P_{p-1}$ as follows.

$$\begin{aligned} P_i &= (t-1) - \tfrac{i}{2} \quad , i \text{ even}, \ 0 \leq i < 2r \\ &= \tfrac{i-1}{2} \ , i \text{ odd}, \ 1 \leq i < r. \\ &= t - 2r + \tfrac{i-1}{2}, \quad i \text{ odd}, \ r+1 \leq i \leq 2r-1. \end{aligned}$$

For even i, the P_i's form a strict decreasing sequence with minimum value $t-r$. For odd i, the P_i's form a strict increasing sequence with maximum value $t-r-1$. Hence P_i's are distinct.

Using $D_i = P_{i-1} - P_i$, $1 \leq i \leq p$, we get,

$$\begin{aligned} D_i &= t-i \quad , \ i \text{ odd} \\ &= i \quad , \ i \text{ even} \end{aligned} \Bigg\} \ 1 \leq i \leq r$$

$$\begin{aligned} &= 2r - i \ , \ i \text{ odd} \\ &= t-2r+i, \ i \text{ even} \end{aligned} \Bigg\} \ r+1 \leq i < 2r$$

$$= t-r \ , \quad i = 2r.$$

If $t = 2r$: For i odd, D_i's are odd and form a strict decreasing sequence.

For i even, and $2 \le i \le 2r-2$, D_i's are even and form a strict increasing sequence.

Also $D_p = D_{2r} = t - r$ is even, and,

$D_r = r < D_{2r} = t - r < D_{r+2} = t - r + 2$, since $2r < t$.

Thus D_i's are distinct.

If $t = 2r+1$: For $1 \le i \le r$, D_i's are even and distinct. For $r+1 \le i < 2r$, D_i's are odd and distinct. Also for $r+1 \le i < 2r$, if i is odd, D_i's form a decreasing sequence from $r-1$ downto 1. If i is even, D_i's form an increasing sequence from $t-r+2$ upto $t-2$.

Now, $D_{2r} = t - r$ is odd. and

$t - 2 > t - r > r - 1$ for $r > 2$.

for $r = 2$, $r - 1 = 1$
and $r + 2 = 2r$.

Thus D_i's are distinct.

$\therefore$ P is a proper difference vector.

Lemma 2: Let $p = 2r$, with r odd and $1 \leq r < n$, if $t = 2n$, and $1 \leq r \leq n$, if $t = 2n+1$. Then a corresponding difference vector exists.

Proof: Define a p-tuple P; $P_0 \dots P_{p-1}$ as follows.

$$\begin{aligned} P_i &= t - 1 - \tfrac{i}{2}, \quad i \text{ even}. \\ &= \tfrac{i-1}{2}, \quad i \text{ odd and } i < r-1. \\ &= t - 2r + \tfrac{i-1}{2}, \quad i \text{ odd and } r \leq i \leq 2r-1. \end{aligned}$$

using $D_i = P_{i-1} - P_i$ we get.

$$\begin{aligned} D_i &= \left.\begin{aligned} &t - i, \quad i \text{ odd} \\ &= i, \quad i \text{ even} \end{aligned}\right\} 1 \leq i \leq r-1. \\ &= \left.\begin{aligned} &2r - i, \quad i \text{ odd} \\ &= t - 2r + i, \quad i \text{ even} \end{aligned}\right\} r \leq i < 2r \\ &= t - r \quad i = 2r. \end{aligned}$$

Then as in Lemma 1, we can show that P_i's are distinct from each other, and also D_i's are distinct from each other. Hence P is a proper difference vector.

Lemma 3 : Let $p = 2r+1$, with r even, and $1 < r < n-1$, if $t = 2n$ or $t = 2n+1$. Then a corresponding difference vector exists.

Proof : Define a p-tuple $P: P_0 \ldots P_{p-1}$ as follows,

$$
\begin{aligned}
P_i &= \frac{i-1}{2}, \quad i \text{ odd} \\
&= t-1-\frac{i}{2}, \quad i \text{ even}, \quad 0 \le i \le r \\
&= 2r+1-\frac{i}{2}, \quad i \text{ even} \quad r < i \le 2r.
\end{aligned}
$$

Then using $D_i = P_{i-1} - P_i$, we get,

$$
\begin{aligned}
D_i &= \left.\begin{aligned} &t-i, && i \text{ odd} \\ &i, && i \text{ even} \end{aligned}\right\} 1 \le i \le r+1. \\
&= \left.\begin{aligned} &2r+2-i, && i \text{ odd} \\ &t-2r+i-2, && i \text{ even} \end{aligned}\right\} r+1 < i \le 2r \\
&= r+2, \quad i = 2r+1.
\end{aligned}
$$

Then as in lemma 1, we can show that the P_i's are distinct from each other. Also D_i's are distinct from each other.

Hence P is a proper difference vector.

Lemma 4: Let $p = 2r+1$, with n odd and $1 \leq r < n-1$, if $t = 2n$ or $t = 2n+1$. Then a corrosponding difference vector exists.

Proof: Define a p-tuple $P : P_0 \ldots P_{p-1}$ as follows.

$$
\begin{aligned}
P_i &= \frac{i-1}{2}, \quad i \text{ odd} \\
&= t-1-\frac{i}{2} \quad i \text{ even}, \; 0 \leq i \leq r-1. \\
&= 2r+1-\frac{i}{2} \quad i \text{ even} \quad r < i \leq 2r.
\end{aligned}
$$

using $D_i = P_{i-1} - P_i$, we get,

$$
\begin{aligned}
D_i &= t-i, \; i \text{ odd} \\
&= i, \; i \text{ even}
\end{aligned}
\Bigg\} \; 1 \leq i \leq r
$$

$$
\begin{aligned}
&= 2r+2-i, \quad i \text{ odd} \\
&= t-2r+i-2, \quad i \text{ even}
\end{aligned}
\Bigg\} \; r < i \leq 2r
$$

$$
= r+2, \quad i = 2r+1.
$$

Then, as in lemma 1, we can prove the distinctness of P_i's and D_i's.

Thus P is a proper difference vector.

From the above discussions, we get the following theorem,

Theorem 2: NBURMD (t, t, p) exists for

(a) t even and $1 \leq p < t-1$.

(b) t odd and $1 \leq p \leq t-3$ and $p = t-1$.

Proof: Lemmas 1 to 4 give the construction of difference vectors for the above cases. Therefore by the discussion in section 2, it follows that the corresponding NBURMD (t, t, p)'s exists.
Thus the proof of the theorem follows.

Note 1: The constructions given in lemmas 1 to 4 can be computed in p steps and hence by note 4 of section 2, NBURMD (t, t, p) when it exists can be computed in pt steps.

Examples: $t = 24$, $p = 12$.

P : 23 0 22 1 21 2 20 15 19 16 18 17

D : 23 2 21 4 19 6 5 20 3 22 1 18.

$t = 13, \; p = 12.$

P : 12 0 11 1 10 2 9 4 8 5 7 6

D : 12 2 10 4 8 6 5 9 3 11 1 7

$t = 24, \; p = 14.$

P : 23 0 22 1 21 2 20 13 19 14 18 15 17 16

D : 23 2 21 4 19 6 7 18 5 20 3 22 1 17.

$t = 13, \; p = 10$

P : 12 0 11 1 10 5 9 6 8 7

D : 12 2 10 4 5 9 3 11 1 8.

$t = 24, \; p = 13.$

P : 23 0 22 1 21 2 20 3 9 4 8 5 7.

D : 23 2 21 4 19 6 17 18 5 20 3 22 8.

$t = 13, \; p = 9$

P : 12 0 11 1 10 2 6 3 5

D : 12 2 10 4 8 9 3 11 6.

$t=24$, $p=15$.

P: 23 0 22 1 21 2 20 3 11 4 10 5 9 6 8

D: 23 2 21 4 19 6 17 16 7 18 5 20 3 22 9.

$t=13$, $p=7$.

P: 12 0 11 1 5 2 4

D: 12 2 10 9 3 11 5.

4. Reference

1. Dutta T.K. and Roy B.K. Construction Of Strongly Balanced Uniform Repeated Measurements Designs: A New Approach. Sankhya: The Indian Journal of Statistics 1992, Special Volume 54, pp 147-153. Dedicated to the memory of R.C. Bose.