STAN-CS-73-388 SEL-73-032

Interconnections for Parallel Memories To Unscramble p-Ordered Vectors

by

Roger C. Swanson

May 1973

Technical Report No. 74

This work was supported by the National Science Foundation Graduate Fellowship Program, by the National Science Foundation under Grant GJ-1180, and by the Joint Services Electronics Program: U. S. Army, U. S. Navy, and U. S. Air Force under Contract N-00014-67-A-0112-0044.

DIGITAL SYSTEMS LABORATORY

STANFORD ELECTRONICS LABORATORIES

STANFORD UNIVERSITY • STANFORD, CALIFORNIA

STAN-W-73-388 SEL 73-032

INTERCONNECTIONS FOR PARALLEL MEMORIES TO UNSCRAMBLE p-ORDERED VECTORS

by

Roger C. Swanson

Technical Report no. 74

May 1973

DIGITAL SYSTEMS LABORATORY

Dept. of Electrical Engineering Dept. of Computer Science

'Stanford University

Stanford, California

This work was supported by the National Science Foundation Graduate Fellowship Program, by the National Science Foundation under Grant GJ-1180, and by the Joint Services Electronics Program: U. S. Army, U. S. Navy, and U. S. Air Force under contract N-00014-67-A-0112-0044.

ABSTRACT

Several methods are being considered for storing arrays in a parallel memory system so that various useful partitions of an array can be fetched from the memory with a single access. Some of these methods fetch vectors in an order scrambled from that required for a computation. This paper considers the problem of unscrambling such vectors when the vectors belong to a class called p-ordered vectors and the memory system consists of a prime number of modules.

Pairs of interconnections are described that can unscramble p-ordered vectors in a number of steps that grows as the square root of the number of memories. Lower and upper bounds are given for the number of steps to unscramble the worst case vector. The upper bound calculation that is derived also provides an upper bound on the minimum diameter of a star polygon with a fixed number of nodes and two interconnections. An algorithm is given that has produced optimal pairs of interconnections for all sizes of memory that have been tried. The algorithm appears to find optimal pairs for all memory sizes, but no proof has yet been found.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

I. INTRODUCI'ION

Many of the large computers that are being used and designed today have a parallel memory system that consists of several memory modules. Each memory module operates independently so that accesses to different modules can be overlapped to improve the overall performance of the computer. One example of a computer that uses such a memory system is the ILLIAC IV [Barnes et al., 1968; Kuck, 1968]. Much work has been done on how data should be stored in this kind of memory so that algorithms for ILLIAC IV can be executed efficiently. Some of the proposed storage schemes provide efficient access to the data, but when data is fetched from the memory, it is not in the order required by the algorithm. Thus, there must be some means to unscramble the data so that it is in the correct order.

This paper considers a computer architecture that is modeled after that of the ILLIAC IV and studies the problem of unscrambling the vectors of data that can be fetched from the memory with a single memory access. Storage structures that have been proposed are shown to give rise to a class of scrambled vectors called p-ordered vectors. Interconnections are presented that can unscramble all vectors in this class. These interconnections are studied in detail to determine the best ones to use for a given number of memory modules.

Section II gives a brief description of the computer model under consideration and presents several ways to store two dimensional arrays in the parallel memory. The examples show the appearance of p-ordered vectors in typical situations. Section III discusses the problem of unscrambling p-ordered vectors and introduces the k-apart interconnection.

We show that a single k-apart interconnection is sufficient to unscramble all p-ordered vectors as long as k is chosen appropriately. Each time a vector is routed along the interconnection, all elements of the vector are transferred in parallel. A number of such routings take place in sequence in order to unscramble a particular p-ordered vector. In Section IV we consider using two different k-apart interconnections to speed up the unscrambling process. We are interested in minimizing the number of routings required by a vector in the worst case. We give lower and upper bounds on this number of routings and show that both bounds are of the order of $\sqrt{N}$, where N is the number of memory modules. The upper bound is shown to be close to the optimal number of routings known for certain values of N. Finally, in Section V an algorithm is derived that gives an optimal pair of interconnections for many values of N. We conjecture -that the algorithm provides an optimal pair for any value of N.

II. &ORDERED VECTORS

We first briefly describe the computer model treated here and then show how two dimensional arrays might be stored in such an architecture. We see that p-ordered vectors arise naturally from these storage structures. In order to define p-ordered vectors for all values of p, we find that we must restrict the number of memory modules to be a prime number.

The computer model used throughout this paper is shown in Figure 2.1. This is the same architecture as the ILLIAC IV with the addition of the interconnection network between the -processing elements and their memories. The single instruction stream is read and decoded by the control unit. Instructions meant for the control unit are executed there. Processing element instructions are sent on to the processing elements,, and all processing elements execute the same instruction simultaneously. Each processing element has an index register and can add the contents of this register to the address in the instruction to obtain the address of the operand in its own memory. Thus, a single load instruction causes each processing element to fetch some word from its memory. The net result is that a vector of data can be fetched in one memory access. The interconnection network consists of a register for each processing element. Each register is connected to the memory buffer register of the corresponding memory unit and to some set of registers in the processing element so that values can be transferred between a processing element and its memory through this register. We are interested in defining the interconnections between the registers themselves so that vectors fetched from the memory can be unscrambled efficiently.

Several people have studied the problem of storing arrays in parallel

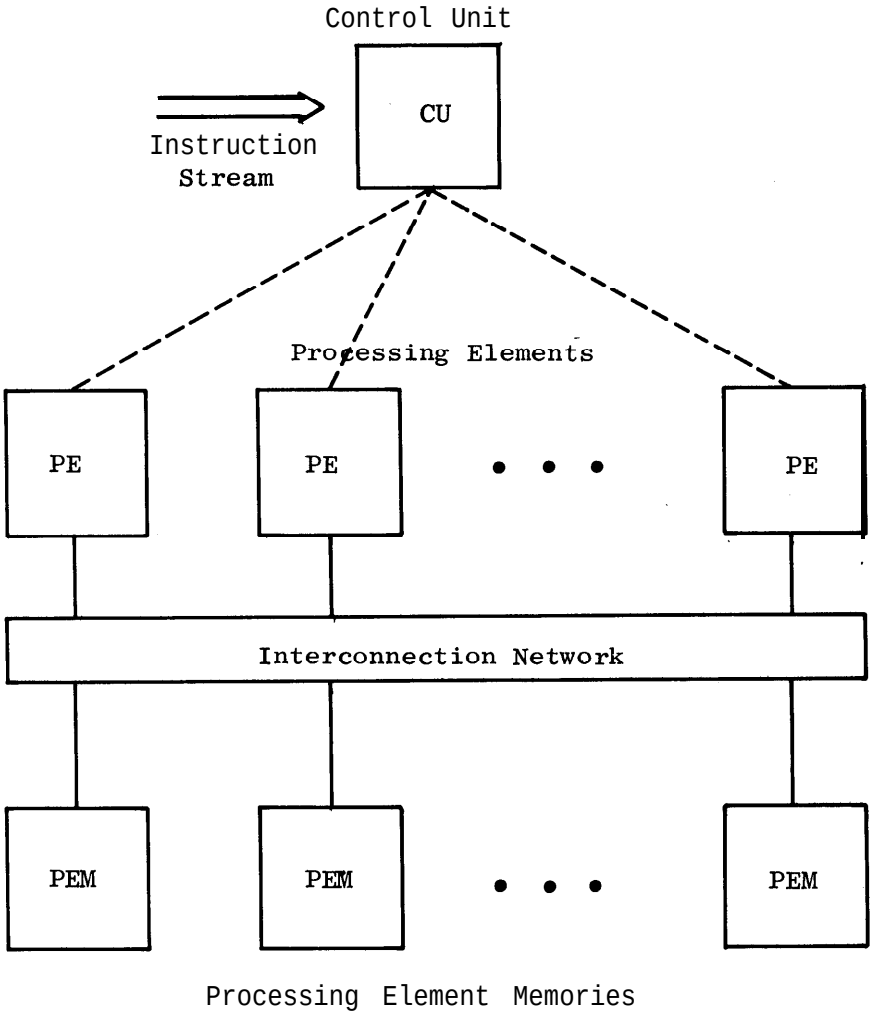

Figure 2.1

Computer Model

memories [Kuck, 1968; Muraoka, 1969; Knowles et al., 1967; Stevens, 1970; Budnik and Kuck, 1971]. The simplest way to store a two dimensional array is to use straight storage allocation as illustrated in Figure 2.2. The rows of the array are spread across the processing element memories, but each column is contained entirely within a single memory. Clearly, any row can be fetched from the memory with a single access as can the main diagonal, but a column requires an access for each element in the column.

Skewed storage allocation can be used to overcome this problem and is illustrated in Figure 2.3. This method of storage allocation also spreads rows of the array across the processing element memories, but the first element of each row is displaced one memory unit from the first element of the previous row. We can still fetch any row with a single memory access by setting the index registers appropriately. Note that it is not possible to fetch the main diagonal with a single access.

These two methods of storage allocation waste space if the number of columns in the array is not the same as the number, of processing element memories. Figure 2.4(a) shows the same two dimensional array packed into the memory with no wasted space. Each new row begins immediately after the previous one. We can still fetch any row with a single memory access by setting the index registers properly, and a study of the figure also shows that any column can be fetched with a single memory access. Figure 2.4(b) shows the contents of the memory buffer -register of each processing element if the first column is fetched. This is the first example of a vector which may not be in the correct order. Suppose that we want the inner product of this column and a row from another matrix that is stored in the same way. We must first align

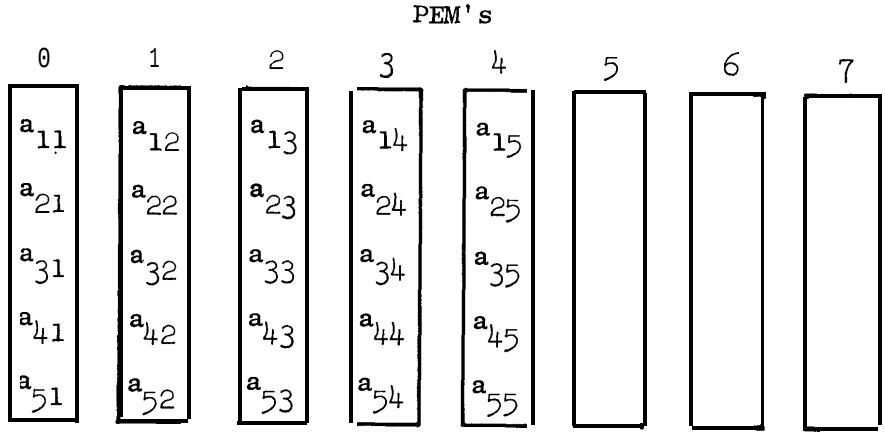

Figure 2.2

Straight Storage Allocation for 5X5 Array

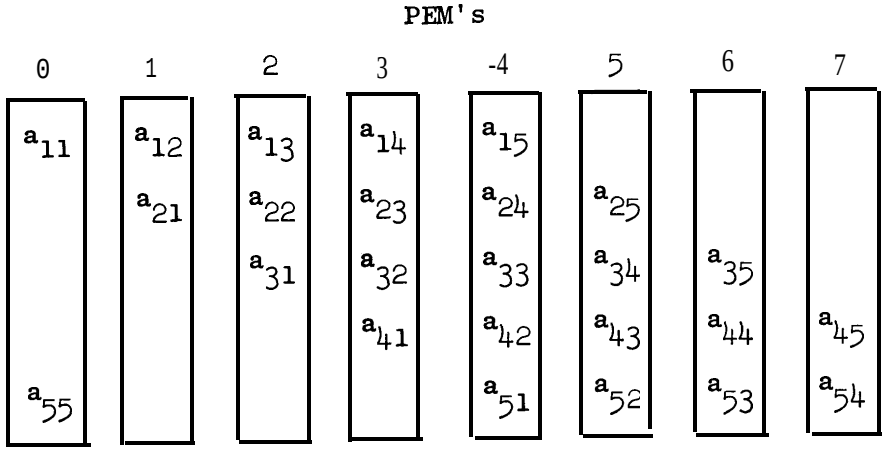

Figure 2.3

Skewed Storage Allocation for 5X5 **Array**

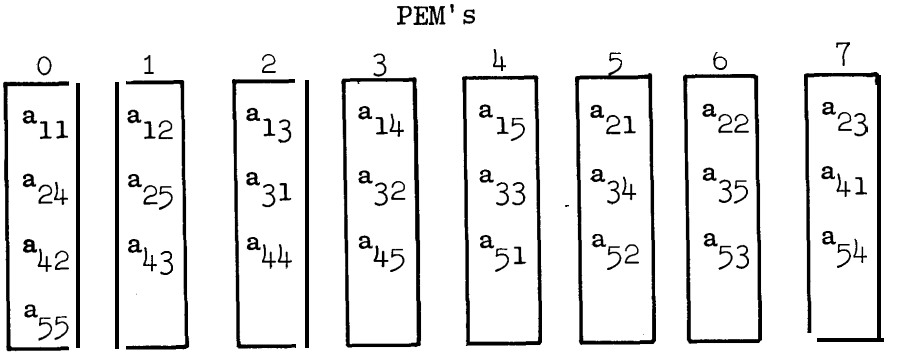

Figure 2.4(a)

Packed Storage Allocation for 5X5 Array

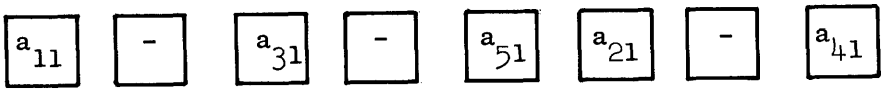

Figure 2.4(b)

Contents Of Memory Buffer Registers after Fetching First Column

the fetched column so that corresponding elements of the row and column are in the same processing element. Then all multiplications can be done in parallel. Note that the second element in the desired order of the column (a_{21}) is five elements away from the first element (all). Similarly, the third element (a_{31}) is five elements from the second. This is an example of a p-ordered vector with p = 5. These p-ordered vectors are characterized quite simply in general. The mathematical definition which follows the next example shows that elements that should be adjacent after the vector has been-unscrambled are p elements apart in the p-ordered vector.

Budnik and Kuck [1971] have looked at the problem of storing arrays so that rows, columns, main diagonal, and square subarrays can all be fetched with a single memory access. Their results place certain restrictions on the number of modules in the memory system. One useful memory size is $2^{2L} + 1$ memories, and they give an example of storing a 4X4 array in five memories. This is shown in Figure 2.5. A study of this example shows that any row or column, the main diagonal, and all 2X2 subarrays can be fetched with one memory access by setting the index registers correctly. Note that if a column is fetched, it is not in the proper order. Elements that should be adjacent are two apart in the fetched vector so it is a 2-ordered vector. Similarly, if the main diagonal is fetched, a j-ordered vector results, since elements that should be adjacent are three elements apart.

With these examples of p-ordered vectors in mind we proceed to give the general definition. We can associate a control vector with any vector fetched from the memory. This control vector specifies how the fetched vector is to be ordered. If X is a vector fetched from a memory with N modules (X has N elements and is called an N-vector),

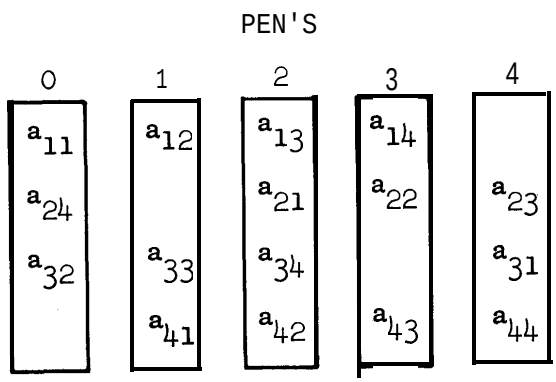

Figure 2.5

Storing 4X4 Array in $2^{2L} + 1$ Memories with Skewing of 2^{L} and L = 1

and A is the control vector, then A(1) = J, $0 \le I, J \le N-1$, means that the value X(1) should be in position J after the vector has been reordered. The vector X is p-ordered if its control vector is p-ordered as given in the following definition.

Definition: An N-vector A is p-ordered, $1 \le p \le N-1$, if its contents are described by

$$A[pi \bmod N] = i \qquad (2.1)$$

where $0 \le i \le N-1$.

Since we are not interested in the actual values of the vector elements fetched from the memory but in their relative positions, all vectors mentioned in the rest of the paper are control vectors. If a control vector can be brought into numerical order, then the same operations applied to the fetched vector will bring it into the desired order. Figure 2.6 shows a 2-ordered vector with N = 7.

The definition in (2.1) requires one restriction. Some value must be assigned to each of the N elements of A. Thus, given any j, $0 \le j \le N-1$, we must be able to find some value of i, $0 \le i \le N-1$, such that

$$pi \bmod N = j.$$

This can also be written as

$$pi \equiv j \pmod N.$$

Such a linear congruence has a solution for i only if gcd(p,N) divides j, where gcd(p,N) is the greatest common divisor of p and N [Andrews, 1971, page 60]. If some i_0 satisfies the congruence, then any multiple of N added to or subtracted from i_0 will also satisfy it. Therefore, if the congruence has a solution, we know that we can find some solution, i, that satisfies $0 \le i \le N-1$. We want a solution for any j, but the only number that divides all integer values from 0 to N-1 is 1. Thus,

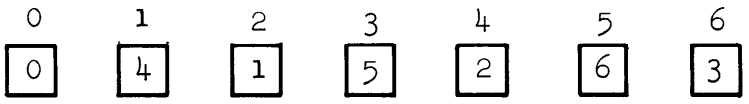

Figure 2.6

2-ordered Vector with N = 7

gcd(p,N) = 1 so that p must be chosen relatively prime to N. Since we want to unscramble all p-ordered vectors with $1 \leq p \leq N-1$, we must restrict N to be prime. Therefore, in the remainder of the paper the number of memory modules, N, is prime.

III. UNSCRAMBLING P-ORDERED VECTORS

Having seen that p-ordered vectors occur naturally when certain storage allocation schemes are used for two dimensional arrays in a parallel memory, we proceed to study the problem of unscrambling these efficiently. We propose an interconnection called a k-apart interconnection and show that a single k-apart interconnection is sufficient to unscramble all p-ordered vectors if k is chosen appropriately.

The most obvious way to unscramble a p-ordered vector is to have the registers containing the vector interconnected in such a way that there is a direct connection from each vector element to the register representing its final position. This idea is generalized in the following definition of a k-apart interconnection.

Definition: N registers are interconnected with a k-apart interconnection, $1 \leq k \leq N-1$, if the contents of register $(ki \bmod N)$ can be transferred directly to register i, with $0 \leq i \leq N-1$. The notation for such an interconnection is

$$\text{reg}\,[ki \bmod N] \rightarrow \text{reg}\,[i].$$

As with the definition of p-ordered vectors, we must restrict k to be relatively prime to N, but remember that N has already been restricted to a prime number. When a vector contained in the registers is to be routed along the interconnection path, all registers transfer their contents simultaneously. If the registers contain a p-ordered vector with $p = k$, then one transfer is sufficient to unscramble the vector to a l-ordered vector. The relationship between a k-apart interconnection and arbitrary p-ordered vectors is given in the next theorem, which states that a single k-apart interconnection can be used to unscramble all p-ordered vectors. Figure 3.1 illustrates this with seven registers

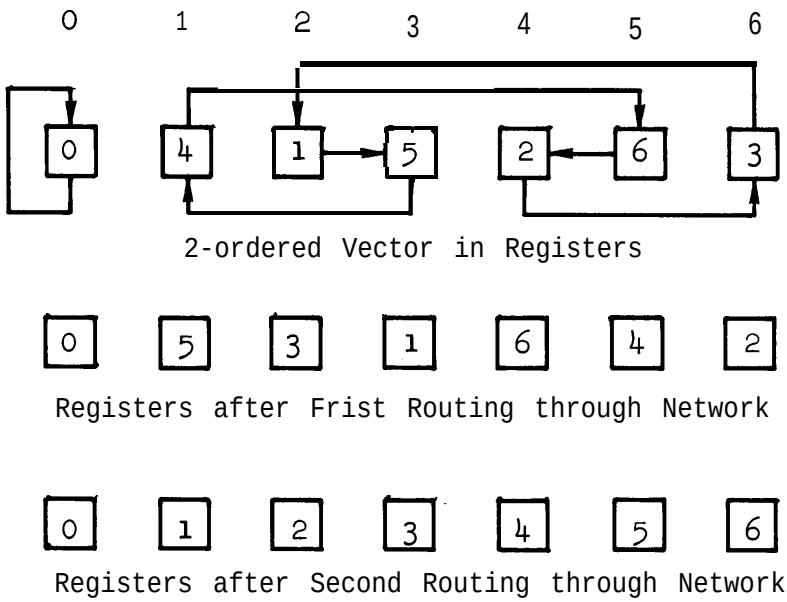

Figure 3.1

Unscrambling a 2-ordered Vector with a 3-apart Interconnection of Seven Registers

interconnected with a j-apart interconnection. The contents of the registers is a 2-ordered vector, and the figure also shows that after two routings along the interconnection path the vector is in the correct order.

Since the proof of the theorem and later discussion use results from group theory, we first describe the notation that is used.

1. A_N is the additive group of integers under addition modulo N. The symbol $+_N$ is used to denote the group operator. This is a cyclic, abelian group, and every element that is relatively prime to N is a generator of the group. The order of the group is N.

2. The notation $(ia)_N$ with i any positive integer and $a \in A_N$ denotes a sum containing i occurrences of the element a. For example,

$$(3a)_N = a +_N a +_N a.$$

If i is a negative integer, then the notation represents a sum containing $|i|$ occurrences of the element -a, which is the inverse of the element a. If i = 0, we define $(ia)_N$ to be 0.

3. M_N is the multiplicative group of integers under multiplication modulo N. The symbol $\bullet_N$ is used for the group operator. In order for this structure to actually be a group, N must be prime. This group is also cyclic and abelian. Since 0 is not in the group, the order of the group is N-l.

Theorem 3.1: Given N registers, with N prime, which are interconnected with a k-apart interconnection, if k is a generator of the group M_N, then any p-ordered vector contained in these registers can be converted to a l-ordered vector by a finite number of routings along the interconnection path.

Proof: Since N is prime, we can replace the mod notation in the

previous definitions of p-ordered vectors and k-apart interconnections with multiplication in the group M_N. Note that register 0 always contains 0 and can be ignored in the proof.

It is straightforward to show that after j routings through the interconnection network the contents of the registers are given by

$$\text{reg}\,[i] = p^{-1} \cdot_N (k^j \cdot_N i), \quad i \in M_N. \tag{3.3}$$

In order for the vector to become l-ordered after some number of routings, j, it is necessary for register i to contain the value i for all $i \in M_N$. From (3.3), j must-satisfy

$$p^{-1} \cdot_N (k^j \cdot_N i) = i, \quad i \in M_N$$

Then the necessary number of routings, j, must satisfy

$$k^j = p. \tag{3.4}$$

Since k is a generator of MN, and p is an element of M_N, such a j clearly exists in the range $0 \le j \le N-2$. Q.E.D.

We have shown that if the registers of the interconnection network are connected with a k-apart interconnection such that k is a generator of the group M_N, any p-ordered vector fetched from memory into these registers can be unscrambled to produce a l-ordered vector. This vector can then be used by the processing elements. If we let $N = 7$, we find that $k = 3$ is a generator of M_7. If we want to unscramble a 2-ordered vector with seven elements, according to (3.4) it will take j routings where j is given by $3^j = 2$. Since $3 \cdot_7 3 = 2$, two routings are required. This is precisely what was shown in Figure 3.1.

Since k is a generator of M_N, and the order of M_N is N-l, each value of k^3 is distinct for $0 \le j \le N-2$. As a result, there must be some value of p that requires a worst case of N-2 routings to unscramble the corresponding p-ordered vector. We can use more interconnections between the

registers to decrease the number of routings in this worst case. Suppose that we use two different k-apart interconnections described by interconnection distances k_1 and k_2. An argument similar to that used in deriving (3.4) shows that in order to unscramble a particular p-ordered vector the problem is to find i and j such that

$$k_1^i \bullet_N k_2^j = p. \tag{3.5}$$

Then i routings along the k_1-apart interconnection followed by j routings along the k_2-apart interconnection will unscramble the p-ordered vector. The interconnections are commutative, since M_N is an abelian group.

IV. USE OF TWO K-APART INTERCONNECI'IONS

The use of two interconnections instead of one creates many new problems which are the subject of the rest of the paper. What conditions must be placed on the values of k_1 and k_2 to guarantee that all p-ordered vectors can be unscrambled by using these two k-apart interconnections? What lower and upper bounds can be derived for the total number of routings required to unscramble any p-ordered vector in the worst case? How should 'kl and k_2 be chosen to minimize the number of routings required to unscramble the worst case p-ordered vector? The first two questions are considered in detail in this section, and the third question is studied in the following section.

Rather than looking for generator pairs for the multiplicative group, M_N, it is simpler to deal with the additive group A_{N-1}. If we find a satisfactory pair, (a,b) in A_{N-1}, the interconnection distances that correspond to this pair are given by

$$k_1 = g^a,$$

$$k_2 = g^b,$$

where g is a generator of M_N.

We can then state the three questions posed at the beginning of this section in terms of the group A_{N-1}. What conditions must be placed on the pair (a,b) to guarantee that the pair is a generator pair for the group A_{N-1}? What lower and upper bounds can be derived for the number of a's and b's required in any sum in the worst case? How should the pair be chosen to minimize the number of terms required in any of the sums?

Stone [1970] considers these problems in a somewhat different form. He is interested in finding a star polygon with minimum diameter

for a specified number of nodes and connections. A star polygon is a directed graph with n nodes, each node having d outward directed edges. The edges of the graph are given by a connection set containing d elements. If the nodes are numbered from 0 to n-l, then each node i has an edge from itself to node $(i + s_j) \bmod n$, where s_j takes each value in the connection set. Stone considers the case with the number of nodes equal to a power of two and provides a lower bound on the diameter for various numbers of connections. He also gives the actual minimum diameter in certain cases, which he obtained by using an exhaustive search. The work that is presented here provides both a lower and an upper bound on the minimum diameter for star polygons with two connections and any number of nodes, although their application to unscrambling p-ordered vectors requires a prime number of nodes. A means of reducing the diameter below this upper bound is discussed in the next section.

The next theorem gives the condition on (a,b) that must be met in order for the pair to be a generator pair. In this and later proofs we implicitly use gcd as a commutative and associative binary operator. This leads to identities such as

$$\gcd(a,b,N-1) = \gcd(\gcd(a,b),N-1) = \gcd(\gcd(a,N-1),b)$$

Theorem 4.1: The pair (a,b) is a generator pair for the group A_{N-1} if and only if gcd(a,b,N-1) = 1.

Proof:

(a) Given gcd(a,b,N-1) = 1. Let h = gcd(a,b). Then positive integers m and n exist such that

$$a = mh, \qquad b = nh, \qquad (4.1)$$

and gcd(m,n) = 1.

From the hypothesis, h is relatively prime to N-l. Therefore, h must be a generator of A_{N-1}. Thus, for all $x \in A_{N-1}$, there exists i, $0 \leq i \leq N-2$, such that

$$x = (ih)_{N-1} = (ih) \bmod N-1.$$

Since m and n are relatively prime, integers j_1 and j_2 exist such that

$$jlm + j_2 n = 1.$$

Multiplying this by ih gives

$$ijlmh + ij_2 nh = ih.$$

Using (4.1) in the above and taking the result modulo N-l gives

$$x = (ih) \bmod N-1 = (ij_1 a + ij_2 b) \bmod N-1$$
$$= ((ij_1)a)_{N-1} +_{N-1} ((ij_2)b)_{N-1}.$$

Thus, any element of A_{N-1} can be written as a sum of a's and b's.

(b) Given that (a,b) is a generator pair for' A_{N-1}. Then for all $x \in A_{N-1}$, there exist $i,j \geq 0$ such that

$$x = (ia)_{N-1} +_{N-1} (jb)_{N-1}.$$

Leth = gcd(a,b) so that (4.1) can be used in the above to give

$$x = (i(mh))_{N-1} +_{N-1} (j(nh))_{N-1}$$
$$= (imh + jnh) \bmod N-1$$
$$= ((im + jn)h) \bmod N-1$$
$$= ((im + jn)h)_{N-1}.$$

This last equation says that h is a generator of A_N 1 so that h must be relatively prime to N-l. Thus, gcd(a,b,N-1) = 1. Q.E.D.

A lower bound on the number of terms required in the worst case to represent any element of A_{N-1} as a sum of a's and b's is given in the following theorem.

Theorem 4.2: Let $a, b \in A_N$ 1 be a generator pair for A_{N-} 1. If each element of A_{N-1} is written as a sum of a's and b's with as few terms as possible, then the number of terms, n, in those representations requiring the most terms is bounded as follows:

$$n \geq \left\lceil \frac{\sqrt{8N - 7} - 3}{2} \right\rceil .$$

Proof: This result is easily seen by considering Table 4.1. Row i of the table shows which elements of A_{N-1} can be represented as a sum of a's and b's containing i terms. Row n is the lowest numbered row such that all elements of A_{N-1} can be found in the table in the rows corresponding to n or fewer terms. Note that each row has one more entry than the previous row so that row i contains i + 1 entries, $0 \leq i \leq n$. We can now see that the number of entries in row 0 through row n must be at least N-l (the number of elements in A_{N-1}). This gives

$$\sum_{i=0}^{n} (i + 1) \geq N-1.$$

This can be written as

$$\sum_{i=1}^{n+1} i = \frac{(n + 1)(n + 2)}{2} \geq N-1,$$

or $n^2 + 3n + 2 \geq 2N - 2.$

This quadratic inequality can be solved for n to give

$$n \geq \left\lceil \frac{\sqrt{8N - 7} - 3}{2} \right\rceil .$$ Q.E.D.

In terms of unscrambling p-ordered vectors the last theorem says that if we use any two k-apart interconnections that can unscramble all p-ordered vectors, then some p-ordered vector will require a number of

Number of Terms in Sum	Elements of A_{N-1} Represented
0	0
1	a b
2	$a +_{N-1} a$ $a +_{N-1} b$ $b +_{N-1} b$
⋮	
n	$(na)_{N-1}$ $((n-1)a)_{N-1} +_{N-1} b$ . . . $(nb)_{N-1}$

Table 4.1

Representation of Elements of A_{N-1} as Sums of the Elements a and b

routings that is at least as great as the lower bound of the theorem, In order to determine how much greater we now construct an upper bound on the number of routings required to unscramble all p-ordered vectors.

We return to the group A_{N-1} and consider generator pairs of the form $(1,b)$. Such pairs will clearly generate all elements of A_{N-1}, since 1 itself is a generator of A_N 1. Let x be any element of A_{N-1}. Then x can be written in terms of $(1,b)$ as

$$x = (ib)_{N-1} +_{N-1} (j1)_{N-1}$$

$$= (ib)_{N-1} +_{N-1} j, \qquad i,j \geq 0.$$

We want to choose b so that i + j <the number of terms in the sum) is minimized for all x. An obvious way to choose i and j for any x is shown in Table 4.2.

The dotted lines partition the table into consecutive rows that have the same value of i. Each partition (except possibly the last) contains b rows. The maximum value of i + j in the table occurs in the last row of the last full partition (a partition with b rows). The number of full partitions is $\lfloor (N-1)/b \rfloor$, and so the value of i in the last full partition is

$$i = \lfloor (N-1)/b \rfloor - 1.$$

The value of j in the last row of any full partition is clearly b-l. Thus, the maximum value of i + j in the table is

$$(i + j)_{max} = \lfloor (N-1)/b \rfloor - 1 + b - 1$$

$$= \lfloor (N-1)/b \rfloor + b - 2. \qquad (4.2)$$

We want to choose b so that $(i + j)_{max}$ is minimized. If we minimize the function $f(b) = (N-1)/b + b - 2$, we find that $b = \sqrt{N-1}$ gives the minimum value. Since b must be an integer, we choose $b = \lfloor \sqrt{N-1} \rfloor$ or $b = \lceil \sqrt{N-1} \rceil$

x	i	j	i + j
0	0	0	0
1	0	1	1
⋮	⋮	⋮	⋮
b-1	0	b-1	b-1
b	1	0	1
⋮	⋮	⋮	⋮
$(2b)_{N-1} - 1$	1	b-1	b
$(2b)_{N-1}$	2	0	2
⋮	⋮	⋮	⋮
$(3b)_{N-1} - 1$	2	b-1	b+1
⋮	⋮	⋮	⋮
N-2			

Table 4.2

Representation of Elements of A_{N-1} as $x = (ib)_N\ 1 +_N 1\ j$

depending on which minimizes $i + j)_{max}$. As long as N is not a perfect square we can show that

$$\left\lfloor \frac{N-1}{\lfloor\sqrt{N-1}\rfloor} \right\rfloor + \lfloor\sqrt{N-1}\rfloor - 2 = \left\lfloor \frac{N-1}{\lceil\sqrt{N-1}\rceil} \right\rfloor + \lceil\sqrt{N-1}\rceil - 2.$$

Since N has been restricted to a prime number, we see that it makes no difference which of the two values of b is chosen. In summary, we have constructed an upper bound on the number of terms required to form any element of A_{N-1} as a sum of 1's and b's. We choose $b = \lfloor\sqrt{N-1}\rfloor$ (or $\lceil\sqrt{N-1}\rceil$ and then the upper bound is

$$UB = \lfloor (N-1)/b \rfloor + b - 2. \qquad (4.9)$$

The second and third columns of Table 4.3 give the lower bound of Theorem 4.2 and the upper bound of (4.9) for various values of N. The fourth column of the table was obtained from a computer program that considered all possible pairs of k-apart interconnections for a given value of N. For each pair the program determined how many routings would be required to unscramble the worst case p-ordered vector. The smallest value in this set of numbers is reported in column four of the table and corresponds to an optimal pair of interconnections. Note that the constructed upper bound, which provides a pair (1,b) that meets this bound, is very close to the optimal solution in all the cases shown.

N	Lower Bound	Upper Bound	Optimal Value
5	2	2	2
7	2	3	3
11	3	4	4
13	4	5	5
17	5	6	5
19	5	6	6
23	6	7	7
31	7	9	8
61	10	13	12
257	22	30	27*

Table 4.3

Lower and Upper Bounds on Number of Routings to Unscramble Worst-case p-ordered Vectors Compared to Optimal Number of Routings in Worst Case

*This value is not the result of an exhaustive search but is the best value found in a partial search and appears to be optimal.

V. IMPROVING THE UPPER BOUND

In this section we again consider the problem of representing all elements of A_{N-1} as a sum of elements from a generator pair (a,b) and construct an upper bound on the number of terms needed in any of these sums. This upper bound is different from the one of the previous section and not quite as good. It does, however, lead to an algorithm that will produce an optimal generator pair for those values of N shown in Table 4.3. We have been unable to prove that the algorithm will produce an optimal generator pair for all N, but the empirical evidence suggests that that is the case.

We begin the derivation by considering some element, a, of A_N 1. We denote by (a) the subgroup generated by a. The order of this subgroup is n, the value of which is given in the following theorem. Since this result is well known, the theorem is stated without proof.

Theorem 5.1: If a is any element of A_{N-1}, then the order of (a) is given by

$$n = (N-1)/\gcd(a,N-1). \tag{5.1}$$

The number of distinct cosets of (a) in A_{N-1} is given in Lang [1968, page 27]. This is denoted by m and is given by

$$m = (N-1)/n. \tag{5.2}$$

Now we construct a table of the elements of A_{N-1} in the following manner. Row zero of the table starts with 0, and each of the other elements of the row is formed by adding a to the previous element. Thus, this row is just the subgroup (a). Choosing an element b from A_{N-1}, we form additional rows by adding b to each element of the previous row. Of course, all addition is done modulo N-1. The rows of the table are just the cosets of (a) in A_{N-1}. If b is chosen so that the first m

rows correspond to the m distinct cosets, then the table with m rows and n columns contains precisely the elements of A_{N-1}. Since (a,b) must be a generator pair for the group A_N-1, Theorem 4.1 says that b must be chosen such that gcd(a,b,N-1) = 1. The following theorem shows that this same restriction gives the m distinct cosets as the first m rows of the table. The theorem uses the fact that two cosets are either equal or have no elements in common.

Theorem 5.2: If gcd(a,b,N-1) = 1, then for all i, $1 \le i \le m-1$,

$$(ib)_{N-1} \notin (a) +_{N-1} (jb)_{N-1}$$

for all j, $0 \le j < i$.

Proof: By using (5.1) and (5.2) we see that gcd (a,N-1) = m. Then we write the hypothesis as

$$\gcd(m,b) = 1. \tag{5.3}$$

Using proof by contradiction, we assume the contrary of the conclusion of the theorem. Thus, there exists i_0, $1 \le i_0 \le m-1$, such that

$$(i_0 b)_{N-1} \in (a) +_{N-1} (jb)_{N-1}$$

for some j, $0 \le j < i_0$.

This statement says that integers k_0, $0 \le k_0 \le n-1$, and j_0, $0 \le j_0 < i_0$, exist such that

$$(i_0 b)_{N-1} = (k_0 a)_{N-1} +_{N-1} (j_0 b)_{N-1}.$$

We can write this equation in terms of regular integer arithmetic as

$$i_0 b - p(N-1) = k_0 a + j_0 b - q(N-1)$$

for some $p,q \ge 0$.

Rearranging the equation, we obtain

$$(i_0 - j_0)b = k_0 a + (p-q)(N-1).$$

Since gcd(a,N-1) = m and N-l = nm, m clearly divides the right-hand side of this equation. Therefore, m must divide the left-hand side as well, but since the hypothesis states that m and b are relatively prime, m must divide the term $(i_0 - j_0)$. If we look at the ranges in which i_0 and j_0 must lie, we see that

$$1 \leq (i_0 - j_0) \leq m-1.$$

It is impossible for m to divide $(i_0 - j_0)$ so we have reached a contradiction, thus proving the theorem. Q.E.D.

Figure 5.1 presents such a table for the group A_{30} with a = 3 and b = 2. The order of the subgroup (a) is given by (5.1) as n = 10, and the number of distinct cosets is given by (5.2) as m = 3. Diagonal lines have been drawn in Figure 5.1 so that those elements on the same diagonal can be represented by a sum of a's and b's with the same number of terms. For example, the elements 4, 5, and 6 are all formed from sums with two terms. If we count the diagonals, starting from 0, we find that all the elements of A_{30} can be represented as sums with no more than eleven terms. Note that this particular upper bound is not as good as that given in Table 4.3, which shows an upper bound of 9 for N = 31.

In the general case, the number of diagonals needed to cover a table like that in Figure 5.1 provides an upper bound on the number of terms needed in any sum if we want to write all elements of A_{N-1} as a sum of a's and b's. We count the diagonals starting with zero, since the element 0 requires no terms in its sum, to obtain the general upper bound of

$$UB = n + m - 2. \qquad (5.4)$$

Obviously, the choice of a determines the values of n and m so we want to choose a to make this bound as small as possible.

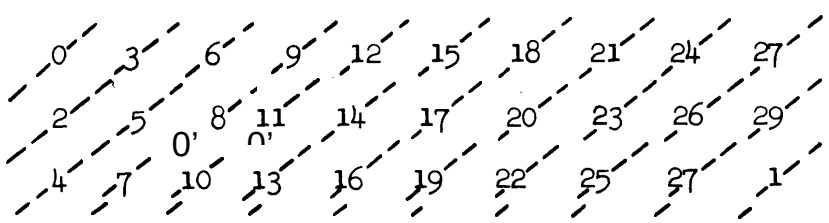

Figure 5.1

Coset Table for the Group A_{30} with a = 3 and b = 2

The results so far say that if we pick some element a and an element b that satisfies (5.3), then all the elements of A_{N-1} can be put into a table with m rows, each row containing a distinct coset of (a) in A_{N-1}. We can cover this table, except the 0 element, with a number of diagonals given by (5.4). If we are careful in selecting b, however, the table can in general be covered with fewer diagonals. If we generate additional rows in the same manner and call each set of m consecutive rows a partition, then each partition consists of the same cosets, but the elements in each coset are cyclically shifted from their positions in partition zero. This is easily seen by writing the element in row i, $0 \le i \le m-1$, and column j, $0 \le j \le n-1$, as

$$x_{ij} = (ib)_{N-1} +_{N-1} (ja)_{N-1}.$$

Then an element in row i of partition q, $q \ge 0$, is given by

$$x_{i+qm,j} = ((i + qm)b)_{N-1} +_{N-1} (ja)_{N-1}$$

$$= (ib)_{N-1} +_{N-1} ((qm)b)_{N-1} +_{N-1} (ja)_{N-1}.$$

If the element $((qm)b)_{N-1}$ can be shown to be an element of (a), then this last equation says that any element in row i + qm also occurs in row i. In addition, each element of row i + qm is formed by adding a to the previous element so that row i + qm is just a cyclic shift of row i. We show that $((qm)b)_{N-1}$ is an element of (a) by showing that m is an element of (a). Since m is in A_{N-1}, we have that $((qm)b)_{N-1} = ((qb)m)_{N-1}$. Therefore, if m is in (a), then $((qb)m)_{N-1}$ is in (a). From (5.1) and (5.2) we see that

$$m = \gcd(a,mn).$$

If m is given, then the element a is given by

$$a = km \tag{595}$$

where $1 \le k \le n-1$

and gcd (k,n) = 1.

The set of positive integers less than n and relatively prime to n forms a group under the operation of multiplication modulo n. Therefore, k is in this group and has an inverse element, k^{-1}, such that

$$(k^{-1}k) \bmod n = 1.$$

Using a result from Knuth [1969, page 39], we can write the last equation as

$$(k^{-1}km) \bmod mn = m.$$

Now we apply (5.5) and use mn = N-l to obtain

$$(k^{-1}a)_{N-1} = m,$$

which gives the result that m is an element of (a).

Figure 5.2 shows additional rows added to the table of Figure 5.1 to give four partitions. Note that each partition consists of cosets that are cyclically shifted two places left from their positions in the previous partition. The diagonals have been extended to cover the new partitions. It is clear from this example that no fewer than eleven diagonals are sufficient, since the element 1 first occurs on the eleventh diagonal.

Suppose we examine the table formed if we use b = 4 along with a = 3. This is shown in Figure 5.3. Now only 'nine diagonals are needed to make certain that every element of the group occurs on some diagonal. The elements within the outlined boundary fall below the ninth diagonal in partition zero, but in partition two, these same elements lie on or above the ninth diagonal.

This procedure to reduce the upper bound can be generalized. We assume that the element a is given so thatnand m are known. The element, L, in the lower right-hand corner of partition zero has a value that depends on which element is--ultimately chosen for b. Since L is in row

m-l and column n-l, its value is

$$L = ((m-1)b)_{N-1} +_{N-1} ((n-1)a)_{N-1},$$

or using the mod notation, we obtain

$$L = (mb - b + na - a) \bmod N-1. \tag{5.6}$$

From (5.4) we know that the occurrence of L in partition zero lies on diagonal n + m - 2. Suppose we specify that in partition q, q ≥ 1, L is to lie on diagonal n + m - 2 - x, where x ≥ 0. Then the occurrence of L in partition q has moved up x diagonals from the occurrence of L in partition zero. Therefore, the position of L in partition q is qm rows down from and qm + x columns to the left of its position in partition zero. As a result we can also write I, as

$$L = ((qm + m - 1)b)_{N-1} +_{N-1} ((n - 1 - qm - x)a)_{N-1}.$$

If we equate this with (5.6) and simplify, we find that b must satisfy

$$qmb \equiv (qm + x)a \pmod{N-1},\ q \geq 1. \tag{5.7}$$

This last result tells how b must be chosen if x is known, but how should x be chosen. We would like to choose x as large as possible and still guarantee that all elements of the group lie on diagonals 0 through n + m - 2 - x. For any value of x the only elements that do not satisfy this requirement are those in partition zero that lie below diagonal n + m - 2 - x. In the last row of partition zero there are exactly x elements below that diagonal. We know, however, that if b can be chosen to satisfy (5.7), then L will lie on diagonal n + m - 2 - x in partition q. If there is room for the other x-l elements to the left of L in partition q then all the elements of the last row will satisfy the requirement. Note that since the relative positions of the elements are the same in each partition, the elements which are below diagonal

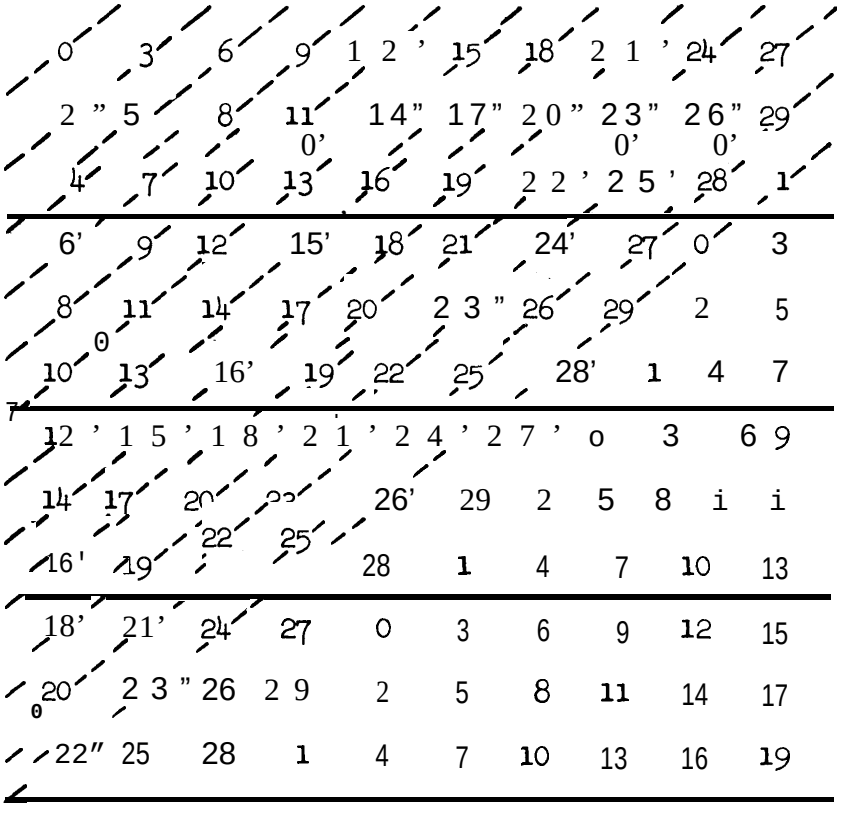

Figure 5.2

Extension to Coset Table for the Group A_{30} with a = 3 and b = 2

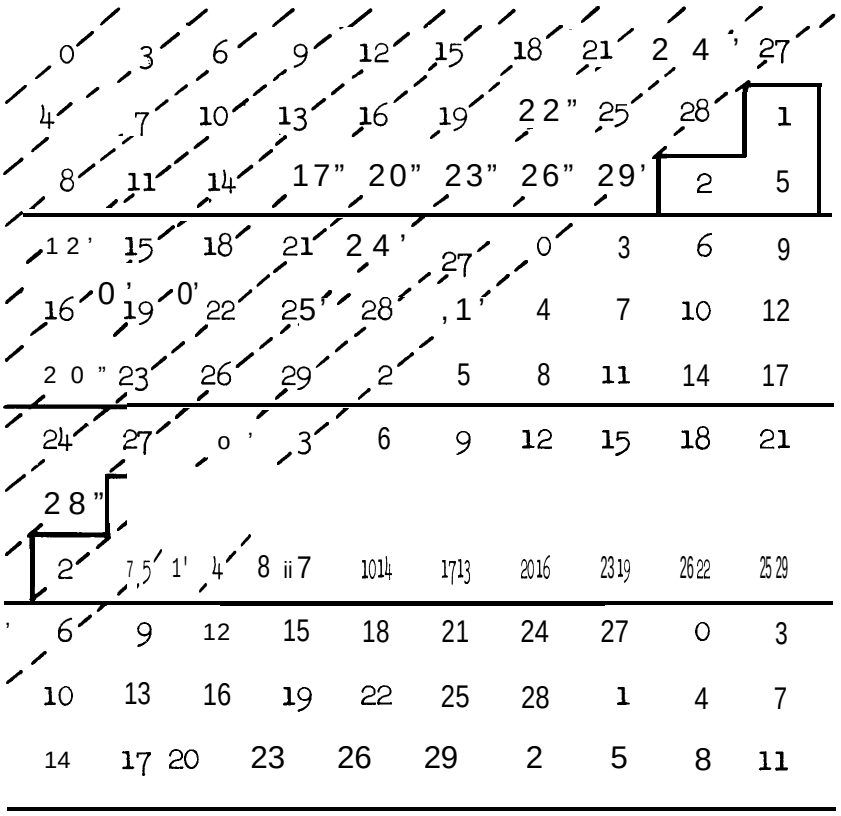

Figure 5.3

Coset Table for the Group A_{30} with a = 3 and b = 4

$n + m - 2 - x$ in the other rows of partition zero will be above that diagonal in partition q. Since column j, $0 \le j \le n-1$, has precisely j columns on its left, and since L is in column $n - 1 - qm - x$ in partition q, the requirement is satisfied if the following is true:

$$x - 1 \le n - 1 - qm - x.$$

This simplifies to

$$x \le \lfloor (n - qm)/2 \rfloor, \quad q \ge 1.$$

Remembering that we wanted to make x as large as possible, we take $q=1$ to obtain

$$x5 \lfloor (n - m)/2 \rfloor. \qquad (5.8)$$

If $n = m$, then the maximum value of x is zero. In that case the upper bound cannot be improved by considering partitions other than partition zero. Then we can choose any value of b that will form a generator pair with the value of a given by (5.5), since we do not care where the elements lie in the other partitions. A convenient choice is $b = 1$.

Now that we have restricted q and know how x should be chosen, we look more carefully at (5.7), which is used to determine b. Setting $q = 1$ and $N-1 = nm$ and using (5.5) we obtain

$$mb \equiv (m + x)km \pmod{nm},$$

where $1 \le k \le n-1$, and $\gcd(k,n) = 1$.

Since $\gcd(m,nm) = m$, and m divides $(m + x)km$, we know that this congruence has m mutually incongruent solutions for b [Andrews, 1971, page 60]. One solution is obtained immediately by using Knuth [1969, page 39]. The result is

$$b_0 = (m + x)k \bmod n$$

where $1 \le k \le n-1$ and $\gcd(k,n) = 1$.

All the solutions are then given in terms of this one and are given by

$$b_t = ((m + x)k) \pmod n + nt, \qquad (5.9)$$

where $1 \le k \le n-1$

$\gcd(k,n) = 1$

$0 \le t \le m-1.$

Any of the m values of b that satisfy (5.9) can be used along with $a = km$ to form a table of the type that has been discussed. The element L will lie on diagonal $n + m - 2 - x$ in partition one, and all the elements that occur in partition zero will be covered by diagonals 0 through $n + m - 2 - x$. We require, however, that each partition contain all elements of the group A_{N-1}, that is that (a,b) be a generator pair of A_{N-1}. As a result, b is restricted by (5.3) to be relatively prime to m so that only certain solutions of (5.9) can be used. The next theorem says that we need only consider values of n, m, and x that have no common factor. For such values of n, m, and x, the theorem says that we can use $k = 1$ in (5.9) and be guaranteed that some value of b_t exists that satisfies (5.9) and that is relatively prime to m.

Before stating the theorem we show how (5.9) is simplified by letting $k = 1$. We want x to be non-negative, and we have already discussed the case of $x = 0$. Using (5.8), we see that we can restrict m to be strictly less than n. This result along with (5.8) leads to the following relation:

$$m + x \le m + \lfloor (n - m)/2 \rfloor$$

$$\le m + (n - m)/2 = (m + n)/2$$

$$< (n + n)/2 = n.$$

Letting $k = 1$ and using this last relation in (5.9) gives

$$b_t = m + x + nt, \quad 0 \le t \le m-1. \qquad (5.10)$$

Theorem 5.3: Given positive integers m, n, and x from which m values are determined by

$$b_t = m + x + nt, \quad 0 \leq t \leq m-1,$$

then a value of t exists such that $\gcd(b_t, m) = 1$ if and only if gcd (m,n,x) = 1.

Proof:

(a) Given a value of t such that $\gcd(b_t, m) = 1$. Assume the contrary of the conclusion that n, m, and x have a common factor f > 1. Then from the manner in which b_t is defined, it is clear that f must divide b_t. This contradicts the hypothesis so that gcd(m,n,x) = 1.

(b) Given gcd(m,n,x) = 1. Let $f = \gcd(m,n)$, and let d = m/f. Then $\gcd(f,x) = 1$ from the hypothesis so that integers u and v exist such that

$$uf + vx = 1. \tag{5.11}$$

We first show that a solution to (5.11) exists such that v is relatively prime to d. If u_0, v_0 is one solution to (5.11), then all solutions are given by [Andrews, 1971, page 24]

$$u = u_0 - tx \tag{5.12}$$
$$v = v_0 + tf$$

where t takes on all integer values.

Note that values for u_0 and v_0 can be found by using the Euclidean algorithm to find gcd(f,x) [Niven and Zuckerman, 1972, page 7]. If v_0 is relatively prime to d, then we are finished. On the other hand, suppose that v_0 and d are not relatively prime. Then the set of all prime numbers that divide d can be partitioned into two disjoint sets described by

Y = {the set of all prime numbers
that divide both d and v_0}

Z = (the set of all prime numbers
that divide d but not v_0}.

Clearly Y is not empty. Set t equal to the product of all of the elements of Z. If Z is empty, set t = 1. Let u', v' be the result of using such a value for t in (5.12). We now show that v' is relatively prime to d. If we assume the contrary, then any prime number that is common to both v' and d must be an element of either Y or Z. Suppose that $y \in Y$ is common to both v' and d. Then since $y \in Y$, y divides v_0, and from (5.12) y divides tf. Now y and f are relatively prime, since if they had a common factor greater than 1, it would divide v_0 as well, but v_0 is a solution to (5.11) and must be relatively prime to f. As a result y must divide t, but t is either 1 or a product of the elements of Z. In either case y cannot divide t. If Z is not empty, the only remaining possibility is that $z \in Z$ is common to both v' and d. Since $z \in Z$, we know that z divides t so from (5.12) it is clear that z divides $v_{0'}$ which again is a contradiction since $z \in Z$. Thus, we have shown the existence of a solution to (5.11) in which v' is relatively prime to d.

Since f = gcd(m,n), integers r and s exist such that

rm + sn = f.

Using this and the solution u',v' in (5.11) gives

u'rm + u'sn + v'x = 1.

This can be written as

u'rm + u'sn + v'x + v'm - v'm + v'in - v'in = 1

with i some integer. Rearranging gives

$$v'(m + x + in) + (u'r - v')m + (u's - v'i)n = 1. \quad (5.13)$$

We now show that i can be chosen so that u's - v'i is a multiple of d. This requires that an integer j exists such that

$$u's - v'i = jd.$$

This can be written as

$$jd + iv' = u's,$$

which is a linear Diophantine equation in unknowns j and i.

We have shown that $\gcd(v',d) = 1$, so the equation clearly has a solution. If i_0 and j_0 satisfy the equation then so do

$$i = i_0 + wd,$$

$$j = j_0 - wv'$$

where w takes on all integers values.

Since d divides m, it is clear that a value of i exists in the inclusive range 0 to m-l. Let i' be such a value and let j' be the corresponding value of j. Using these results along with n = ef and m = df, we see that (5.13) becomes

$$v'(m + x + i'n) + (u'r - v')m + j'def =$$

$$v'(m + x + i'n) + (u'r - v' + j'e)m = 1.$$

Now the second factor of the first term is just $b_{i'}$, so that $b_{i'}$ is relatively prime to m. Q.E.D.

In the discussion so far we have assumed that we are given the value of n, from which we can calculate m, x, and a generator pair (a,b). In order to complete the procedure we must specify how n is chosen. Since n represents the order of a subgroup of the group $A_{N-l'}$ only those values of n that divide N-l (the order of the group A_N-1) can be considered. From (5.8) we see that in order to have x be non-negative, we must have

$n \leq m$. Thus, from (5.2), which defines m in terms of n, we finally conclude that we must limit n to those integer values that are greater than or equal to $\sqrt{N-1}$ and that also divide N-1. We then choose that value that minimizes the upper bound of $n + m - 2 - x$ and that also leads to a generator pair (a,b).

An example should clarify the use of the&procedure that has been developed in the preceding discussion. We consider the group A_{30}, which was used in examples at the beginning of this section, to see if the upper bound of nine depicted in Figure 5.3 can be improved. Figure 5.4 shows the calculations that give the best value for n. The first column contains those integers that are greater than or equal to $\sqrt{30}$ and that divide 30. The second column is given by (5.2), the third column is the maximum value for x as given by (5.8), and the last column gives the upper bound. The smallest value of the upper bound is 8 so we should use $n = 10$, $m = 3$, and $x = 3$. From (5.5) we find that a value for a is $a = 3$. Now we use (5.10) to obtain a value for b. Substituting the known values of the parameters, we obtain

$$b_t = 3 + 3 + 10t,$$

$$\text{or } b_t = 6 + 10t.$$

Noting that b must be relatively prime to m, we let $t = 1$ and obtain $b = 16$. Figure 5.5 shows the coset table that results from the generator pair (3,16). Note that eight diagonals do indeed cover all the elements of the group and that eight was the optimal number of routings needed to unscramble the worst case p-ordered vector when $N = 31$ (see Table 4.3).

We conclude this section by summarizing this algorithm, assuming that the value of N is given and that we want an "optimal" generator pair for the group A_{N-1}.

n	m	x	n+m-2-x
6	5	0	9
10	3	3	8
15	2	6	9
30	1	14	15

Figure 5.4

Values of n and Corresponding Values
for m, x, and Upper Bound for N-l = 30

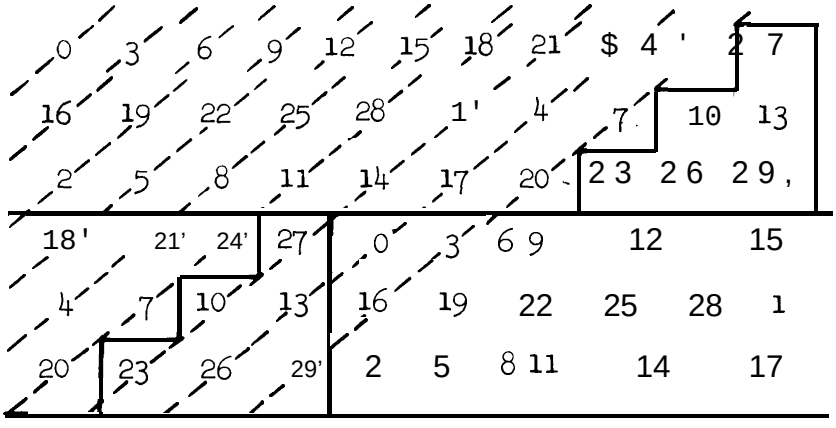

Figure 5.5

Coset Table for the Group A_{30} with a = 3 and b = 16

1. For each value of n such that $n \geq \sqrt{N-1}$ and n divides N-1, calculate $m = (N-1)/n$ and calculate all values of x such that $0 \leq x \leq \lfloor (n-m)/2 \rfloor$.
2. For each of the triples in Step 1 calculate $n + m - 2 - x$. Let n', m', and x' be the values that minimize $n + m - 2 - x$ such that n', m', and x' have no common factor or such that $x' = 0$.
3. If $x' = 0$, use $a = m'$ and $b = 1$ as the generator pair.
4. Otherwise, let $a = m'$ and choose b to satisfy the equation

$$b_t = m' + x' + n't, \quad 0 \leq t \leq m' - 1,$$

such that $\gcd(b,m') = 1$.

Step 4 as stated might require generating several values of b_t until one that is relatively prime to m' is found. The proof of Theorem 5.3, however, does give an explicit procedure to find a value of t such that b_t is relatively prime to m'. This algorithm has been applied to all of the values of N given in Table 4.3. The generator pairs that are found can generate all elements of the group A_{N-1} as sums containing no more terms than the optimal value given in Table 4.3. Therefore, the algorithm appears optimal, but as mentioned before, no proof has been found.

VI. CONCLUSION

We began this discussion by giving examples of ways to store two dimensional arrays in a parallel memory like that used in ILLIAC IV. Certain of the structures give efficient access to the important partitions of the arrays, but some partitions are not ordered correctly after they have been fetched. The examples showed that these scrambled vectors belong to the class of p-ordered vectors. In order to allow p-ordered vectors for any value of p, it was necessary to restrict the number of memory modules, N, to be a prime number. Then k-apart interconnections were defined, and we showed that a single k-apart interconnection of the interconnection registers was sufficient to unscramble all p-ordered vectors as long as k was chosen as a generator of the group M_N. Next we considered the problem of choosing two different k-apart interconnections in such a way that all p-ordered vectors can be unscrambled in as few a number of routings as possible. This problem is isomorphic to the problem of finding a generator pair (a,b) for the group A_N-1 such that all elements of the group can be represented as a sum of a's and b's with a minimum number of terms in the longest sum. If such a generator pair can be found, then the best pair of interconnections is $k_1 = g^a$ and $k_2 = g^b$, where g is a generator of M_N. Lower and upper bounds were derived for the number of terms needed in the longest sum. Both of these bounds are of the order of $\sqrt{N}$, and the constructed upper bound, which gives a generator pair of the form (l,b), gave results that were- very close to optimal for the values of N that were studied. The final problem that was studied was the problem of reducing the upper bound as much as possible to obtain an optimal generator pair. An algorithm was given that produces a generator pair

that should actually be optimal. Although a proof of this has not been found, the algorithm has produced optimal pairs in all the cases tried so far. It should be pointed out that this algorithm is applicable for **any** group A_{N-1}, regardless of the value of N. In order to apply the results to unscrambling p-ordered vectors, however, it is necessary to restrict N to be prime.

There are several problems that remain unanswered. The most striking is to determine if the restriction of a prime number of memories can be removed. Perhaps other data structures or other kinds of interconnections can alleviate the problems that the methods presented here are intended to solve. The question of whether the algorithm to find optimal generator pairs always finds an optimal pair should be considered, since that problem is an interesting mathematical problem in its own right, regardless of its application to computer architecture. Finally, what can be said about the problem of using more than two interconnections in the interconnection network. Can the results here be extended, or must new tools be developed.

REFERENCES

Andrews, George E., Number Theory, W. B. Saunders Company, Philadelphia, Pennsylvania, 1971.

Barnes, George H. et al., "The ILLIAC IV Computer", IEEE Transactions on Computers, vol. C-17, No. 8, pp. 746-757, August 1968.

Budnik, Paul and David J. Kuck, 'The Organization and Use of Parallel Memories", IEEE Transactions on Computers, Vol. C-20, No. 12, pp. 1566-1569, December 1971.

Knowles, Michael et al., "Matrix Operations on ILLIAC IV', Report No. 52, Department of Computer Science, University of Illinois, Urbana, Illinois, March 1967.

Knuth, Donald E., The Art of Computer Programming, Vol. 1, Addison-Wesley Publishing Company, Reading, Massachusetts, 1969.

Kuck, David J., "ILLIAC IV Software and Application Programming", IEEE Transactions on Computers, Vol. C-17, No. 8, pp. 758-770, August 1968.

Lang, Serge, Algebraic Structures, Addison-Wesley Publishing Company, Reading, Massachusetts, 1968.

Muraoka, Yoichi, "Storage Allocation Algorithms in the Tranquil Compiler", Report No. 159, Department of Computer Science, University of Illinois, Urbana, Illinois, January 1969.

Niven, Ivan and Herbert S. Zuckerman, An Introduction to the Theory of Numbers, John Wiley and Sons Inc., New York, 1972.

Stevens, James E., "Matrix Multiplication Algorithms for ILLIAC IV, Document No. 231, Department of Computer Science, University of Illinois, August 1970.

Stone, Harold S., "The Organization of High-Speed Memory for Parallel Block Transfer of Data", IEEE Transactions on Computers, Vol. C-19, No. 1, pp. 47-53, January 1970.