STAN-CS-71-237 SU-SEL-71-050

A Generalization of the Divide-Sort-Merge Strategy for Sorting Networks

by

David C. Van Voorhis

August 1971

Technical Report No. 16

This work was conducted while the author was a National Science Foundation graduate fellow and was partially supported 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 and by the National Science Foundation under Grant GJ1180.

DIGITAL SYSTEMS LABORATORY

STANFORD ELECTRONICS LABORATORIES

STANFORD UNIVERSITY · STANFORD, CALIFORNIA

STAN-CS-71-237 SEL-71-050

A GENERALIZATION OF THE DIVIDE-SORT-MERGE STRATEGY FOR SORTING NETWORKS

by

David C. Van Voorhis

August 1971

Technical Report no. 16

DIGITAL SYSTEMS LABORATORY

Stanford Electronics Laboratories Computer Science Department

Stanford University

Stanford, California

This work was conducted while the author was a National Science Foundation graduate fellow and was partially supported 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 and by the National Science Foundation under grant GJ 1180.

A GENERALIZATION OF THE DIVIDE-SORT-MERGE STRATEGY FOR SORTING NETWORKS

by

David C. Van Voorhis

ABSTRACT

With a few notable exceptions the best sorting networks known have employed a "divide-sort-merge" strategy. That is, the N inputs are divided into 2 groups - - normally of size $\lceil \frac{1}{2}N \rceil$ and $\lfloor \frac{1}{2}N \rfloor$ * - - that are sorted independently and then "merged" together to form a single sorted sequence. An N-sorter network that uses this strategy consists of 2 smaller sorting networks followed by a merge network. The best merge networks known are also constructed recursively, using 2 smaller merge networks followed by a simple arrangement of $\lceil \frac{1}{2}N \rceil - 1$ comparators.

We consider a generalization of the divide-sort-merge strategy in which the N inputs are divided into $g \geq 2$ disjoint groups that are sorted independently and then merged together. The merge network that combines these g sorted groups uses $d \geq 2$ smaller merge networks as an initial subnetwork. The two parameters g and d together define what we call a " $[g,d]$ " strategy.

* Here $\lceil x \rceil$ denotes the smallest integer greater than or equal to x, whereas $\lfloor x \rfloor$ denotes the largest integer less than or equal to x.

A [g,d] N-sorter network consists of g smaller sorting networks followed by a [g,d] merge network. The initial portion of the [g,d] merge network consists of d smaller merge networks; the final portion, which we call the "f-network," includes whatever additional comparators are required to complete the merge. When $g = d = 2$, the f-network is a simple arrangement of $\lceil \frac{1}{2}N \rceil - 1$ comparators; however, for larger g,d the structure of the [g,d] f-network becomes increasingly complicated.

In this paper we describe how to construct [g,d] f-networks for arbitrary g,d. For $N > 8$ the resulting [g,d] N-sorter networks are more economical than any previous networks that use the divide-sort-merge strategy; for $N > 34$ the resulting networks are more economical than previous networks of any construction. The [4,4] N-sorter network described in this paper requires $\frac{1}{4} N(\log_2 N)^2 - \frac{1}{3} N(\log_2 N) + O(N)$ comparators, which represents an asymptotic improvement of $\frac{1}{12} N(\log_2 N)$ comparators over the best previous N-sorter. We indicate that special constructions (not described in this paper) have been found for $[2^r, 2^r]$ f-networks, which lead to an N-sorter network that requires only $.25\, N(\log_2 N)^2 - .372\, N(\log_2 N) + O(N)$ comparators.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The author is greatly indebted to Dr. Harold S. Stone for his prompt and careful attention to several versions of this paper, and for his many constructive suggestions that have been encorporated. The author also wishes to thank Dr. Robert W. Floyd and Dr. Donald E. Knuth, who each made several helpful suggestions.

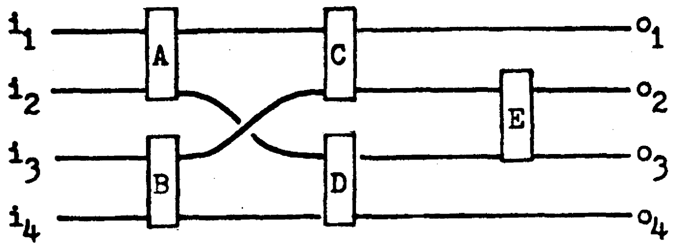
I. Introduction

A comparator network with 4 inputs is illustrated in Fig. 1(a). Each of the 5 comparators, labeled A, B, C, D, and E, compares its two inputs and emits the smaller on its higher output lead and the larger on its lower output lead. An abbreviated diagram for this comparator network is given in Fig. 1(b), where each comparator is replaced by a vertical line connecting the two comparands.

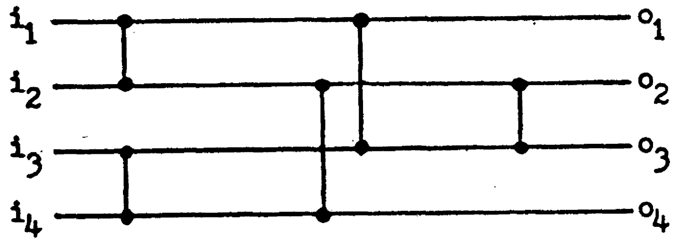
A comparator network with N input and output leads is called an N-sorter network, or simply an N-sorter, if for any multiset* of inputs $I = \{i_1, i_2, \ldots, i_N\}$, the resulting outputs $O = \{o_1, o_2, \ldots, o_N\}$ satisfy: 1) O is a permutation of I; and 2) $o_j \leq o_k$ if $j \leq k$. The network depicted in Fig. 1 is a 4-sorter, since comparators A through D move the smallest input to o_1 and the largest input to o_4, and then E orders the remaining two inputs.

From an engineering viewpoint it may be desirable to use as few comparators as possible when constructing a network to sort N inputs. (An alternative design objective would be to minimize the delay required to sort N items.) Let S(N) represent the minimum number of comparators required by an N-sorter network. R. W. Floyd and D. E. Knuth [2] have determined S(N) for $N \leq 8$ by proving a lower bound for S(N) that is precisely equal to the number of comparators actually contained in the most economical N-sorter network known. However, for $N > 8$, the value of S(N) and even the asymptotic behavior of the function remain an open question. The strongest lower bound known for S(N), proved by

* A multiset is like a set except that it may contain repetitions of elements. See D. E. Knuth [1].



(a)



(b)

Fig. 1. 4-sorter network.

D. Van Voorhis [3], increases as $N(\log_2 N)$, whereas the strongest upper bound known - - i.e. the number of comparators actually required by the most economical N-sorter known, designed by K. E. Batcher [4] and improved by M. W. Green [5] - - increases as $N(\log_2 N)^2$.

Batcher's N-sorter network contains $B(N)$ comparators, where

$$B(N) = \tfrac{1}{4}N(\log_2 N)^2 - \tfrac{1}{4}N(\log_2 N) + N + O(1). \qquad (1)$$

Although Green has been able to improve upon Batcher's networks, the net effect of Green's modification is simply to reduce the coefficient in the linear term of Equation (1) from unity to $\frac{13}{16}$. In this paper we present an extension of Batcher's constructions which reduces the coefficient of $N(\log_2 N)$ in (1) from $-\frac{1}{4}$ to $-\frac{1}{3}$. Our construction achieves an improvement of $\sim\frac{1}{12}N(\log_2 N)$ over the best previous networks, although the asymptotic growth is still $\tfrac{1}{4}N(\log_2 N)^2$. We indicate that a modification of our construction, which is too complicated to include here, reduces the coefficient of $N(\log_2 N)$ in (1) to $-.372$.

II. The Divide-sort-merge Strategy

It is not always easy to determine whether a given comparator network is an N-sorter. For example, it can be shown that the comparator network in Fig. 2(b) is a 4-sorter whereas that in Fig. 2(a) is not. One way to check a network is to see whether it will sort all $N!$ permutations of the numbers $1,2,\ldots,N$ as inputs. However, the following important theorem reduces to 2^N the number of input patterns required to test the design of an N-sorter network.

Theorem: (Zero-One Principle)

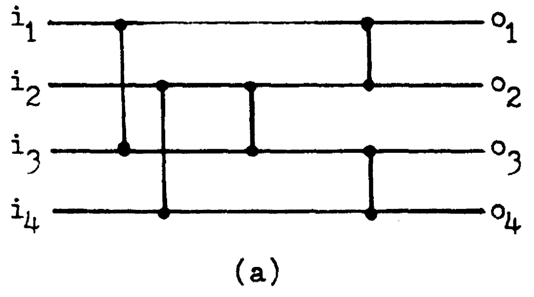
A comparator network with N inputs and N outputs is an N-sorter if and only if it will sort all 2^N combinations of N inputs for which each input is either 0 or 1. (See [2 , 5].)

Proof:*

The "only if" portion of the theorem is obvious; to prove the remainder of the theorem we show that if a comparator network C is not an N-sorter network, then there is at least one combination of 0's and 1's as inputs that C fails to sort.

Suppose that C is not an N-sorter network, so that for some multiset of inputs $I = \{i_1,i_2,\ldots,i_N\}$ it yields the incompletely ordered outputs $O(I) = \{o_1,o_2,\ldots,o_N\}$. This means that, although $O(I)$ is a permutation of I, $o_j > o_k$ for some indices satisfying $1 \leq j < k \leq N$. Now it is easily verified (by induction) that if $f(x)$ is any non-decreasing function (i.e. if $x \leq y$ implies that $f(x) \leq f(y)$,) then

* This proof was suggested to the author by D. E. Knuth.



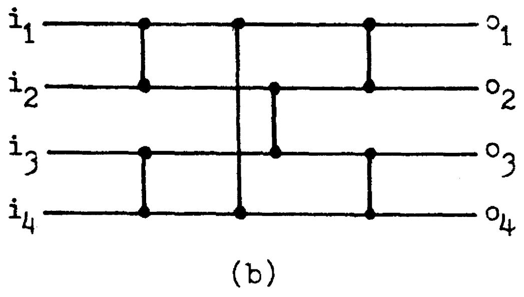
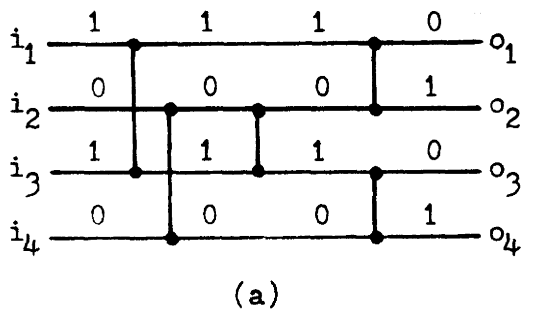


Fig. 2. Which of these is a 4-sorter network?



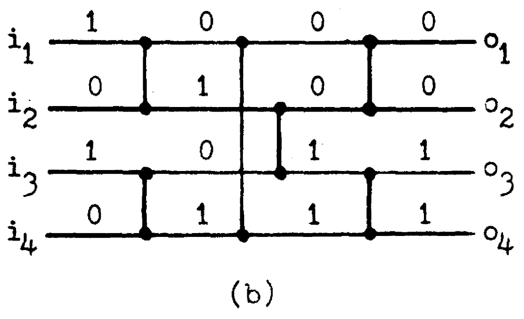


Fig. 3. Testing Comparator Networks.

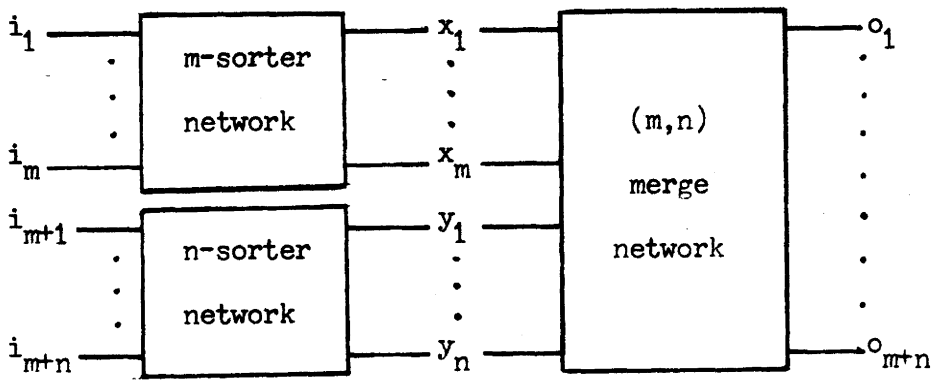


Fig. 4. (m+n)-sorter network T.

since $\max[f(x),f(y)] = f(\max[x,y])$,

$$O(\{f(i_1),f(i_2),\ldots,f(i_N)\}) = \{f(o_1),f(o_2),\ldots,f(o_N)\}. \qquad (2)$$

Therefore, using

$$f(x) = \begin{cases} 0, & x \leq o_k; \\ 1, & x > o_k, \end{cases} \qquad (3)$$

we obtain the inputs $I = \{f(i_1),f(i_2),\ldots,f(i_N)\}$, which is a combination of 0's and 1's that C fails to sort, since $f(o_j) = 1 > f(o_k) = 0$.

Q.E.D.

The theorem is illustrated in Fig. 3. The inputs $I = \{1,0,1,0\}$ are applied to the 4 input leads of each network in Fig. 2. The first network fails to arrange the inputs into non-decreasing order; therefore, it is not a 4-sorter network. The second network is a 4-sorter since it will order properly these inputs and also the other 15 combinations of 0's and 1's as inputs.

Although 2^N grows much more slowly than N!, it is not feasible to test large networks for 2^N different combinations of inputs. Therefore, if we desire large sorting networks, we must build them in such a way that we can prove "by construction" that they will arrange all combinations of inputs into non-decreasing order. The Zero-One Principle is helpful in developing such proofs.

The most successful strategy for designing large sorting networks, suggested by R. C. Bose and R. J. Nelson [6], has been to build them out of smaller sorting networks. The inputs are divided into two groups that are sorted separately and then combined, or <u>merged</u>, to form a single sorted multiset. This <u>divide-sort-merge</u> strategy is illustrated in Fig. 4 by the N-sorter network T, which consists of:

i) an m-sorter network that operates on the inputs $\{i_1, i_2, \ldots, i_m\}$ to produce the sorted multiset $X = \{x_1, x_2, \ldots, x_m\}$; and

ii) an n-sorter network, where $n = N - m$, that transforms the inputs $\{i_{m+1}, i_{m+2}, \ldots, i_N\}$ into the sorted multiset $Y = \{y_1, y_2, \ldots, y_n\}$; followed by

iii) an <u>(m,n) merge network</u> that combines X and Y into the single sorted multiset $O = \{o_1, o_2, \ldots, o_N\}$.

We can use the divide-sort merge strategy recursively to achieve N-sorter networks for arbitrary N, provided we can construct the necessary merge networks. Bose and Nelson suggested building an (m,n) merge network out of three smaller merge networks arranged in a pattern resembling the final three comparators of the 4-sorter in Fig. 1. For example, when m and n are both even and $m \leq n$, Bose and Nelson's (m,n) merge network consists of the following. (See Fig. 5.)

BN1: a $(\frac{1}{2}m, \frac{1}{2}n)$ merge network that determines the smallest $\frac{1}{2}m$ members of O, namely $o_1, o_2, \ldots, o_{\frac{1}{2}m}$; and

BN2: a $(\frac{1}{2}m, \frac{1}{2}n)$ merge network that determines the largest $\frac{1}{2}m$ members of O; followed by

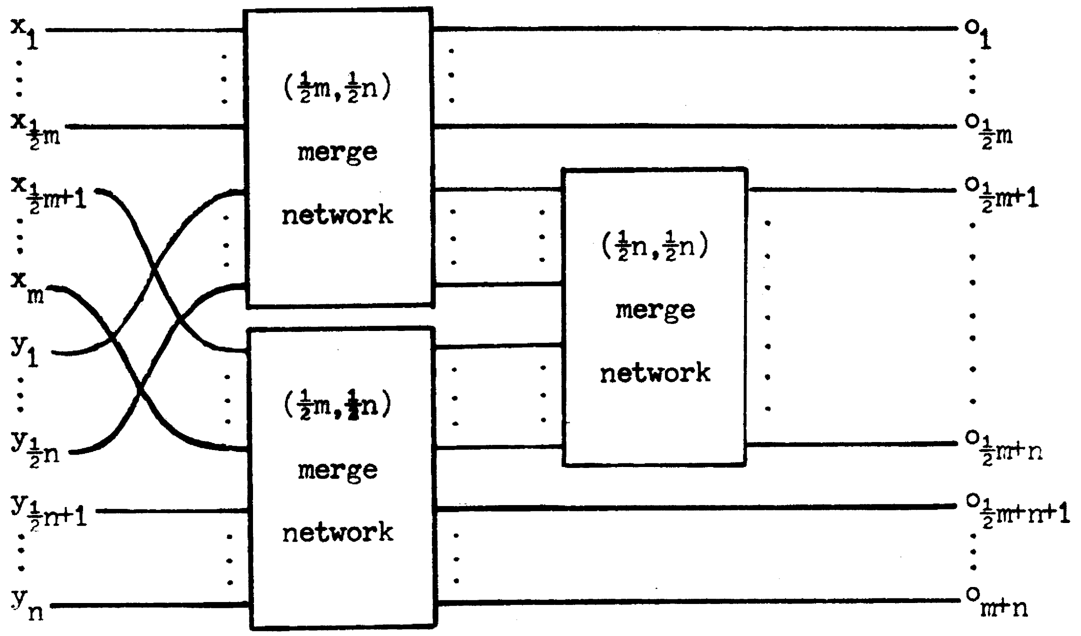


Fig. 5. Bose and Nelson's (m,n) merge network.

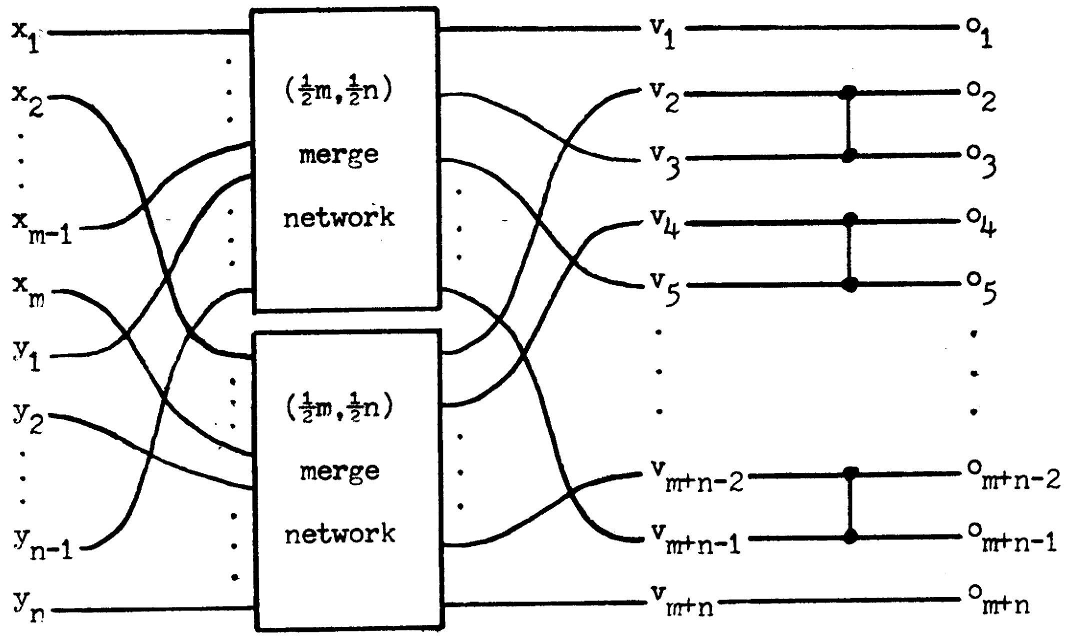


Fig. 6. Batcher's (m,n) merge network.

BN3: a $(\frac{1}{2}n,\frac{1}{2}n)$ merge network that determines the remaining n members of O.

K. E. Batcher [4] proposed a different merging strategy which is more economical than Bose and Nelson's, and which has not been improved upon. The general (m,n) merge network is defined recursively, beginning with the $(1,1)$ merge network, which is a single comparator. When m and n are both even integers greater than one, Batcher's (m,n) merge network consists of the following. (See Fig. 6.)

B1: a $(\frac{1}{2}m,\frac{1}{2}n)$ merge network that combines the odd members $X_o = \{x_1,x_3,\ldots,x_{m-1}\}$ and $Y_o = \{y_1,y_3,\ldots,y_{n-1}\}$ to form the odd members of an intermediate multiset V, namely $V_o = \{v_1,v_3,\ldots,v_{m+n-1}\}$; and

B2: a $(\frac{1}{2}m,\frac{1}{2}n)$ merge network that merges the even members X_e and Y_e to form $V_e = \{v_2,v_4,\ldots,v_{m+n}\}$; followed by

B3: the $\frac{1}{2}(m+n)-1$ comparators $v_{2k+2}:v_{2k+3}$, $0 \leq k < \frac{1}{2}(m+n)-1$.

Since Batcher's (m,n) merge network is the simplest example of a more general strategy described in the next two sections, it is instructive to work through the proof that the network described above and depicted in Fig. 6 leaves the outputs $O = \{o_1,o_2,\ldots,o_{m+n}\}$ sorted.

Suppose that the network T depicted in Fig. 4 consists of any m-sorter network, any n-sorter network, and Batcher's (m,n) merge network. Clearly the (m,n) merge network orders O iff t is an $(m+n)$-sorter network. Therefore, the Zero-One Principle guarantees that the (m,n) merge network orders O iff T sorts all combinations of $m+n$ 0's and 1's as inputs.

For any combination of 0's and 1's as inputs to T, the m-sorter sorts X while the n-sorter sorts Y. The sorted multiset X consists of r 0's followed by m-r 1's, and Y contains s 0's followed by n-s 1's, where for different combinations of inputs to T, r and s assume all combinations of the values $0 \leq r \leq m$; $0 \leq s \leq n$. Let n_o represent the number of 0's that go into V_o, that is, the number of 0's in X_o plus the number in Y_o. Let n_e represent the number of 0's that go into V_e. Then

$$n_e \leq n_o \leq n_e + 2, \tag{4}$$

since each of the two sorted multisets X and Y contributes either the same number of 0's to V_o and V_e or else one more 0 to V_o.

After the odd and even members of X and Y have been merged to form V_o and V_e, the following situation exists:

1) V_o and V_e are each ordered.

2) The first $2n_e$ elements of V are, therefore, all 0.

3) The remaining $m + n - 2n_e$ elements are:

   a) all 1 if $n_o = n_e$; or

   b) 0 followed by 1's if $n_o = n_e + 1$; or

   c) 010 followed by 1's if $n_o = n_e + 2$.

The elements of V are sorted except in Case c) which requires an additional comparator for the adjacent pair $v_{2n_e+2} : v_{2n_e+3}$. For different combinations of inputs to T, n_o and n_e will assume all of the values $0,1,\ldots,\frac{1}{2}(m+n)$. Case c) can occur for each of the possible values of n_e such that $n_o = n_e + 2 \leq \frac{1}{2}(m+n)$. Therefore, the comparators listed in B3 above are both necessary and sufficient to complete the merge.

Batcher's merge strategy is illustrated by the 8-sorter network in Fig. 7. The 10 comparators in Part A comprise two 4-sorters that order X and Y. (Note that each 4-sorter consists of two 2-sorters, i.e. comparators, followed by a (2,2) merge network.) The three comparators in Part B merge $X_o = \{x_1,x_3\}$ and $Y_o = \{y_1,y_3\}$ to form $V_o = \{v_1,v_3,v_5,v_7\}$, while the three comparators in Part C comprise a (2,2) merge network for X_e and Y_e. The comparators in Part D are those called for in B3, which combine V_o and V_e to form O.

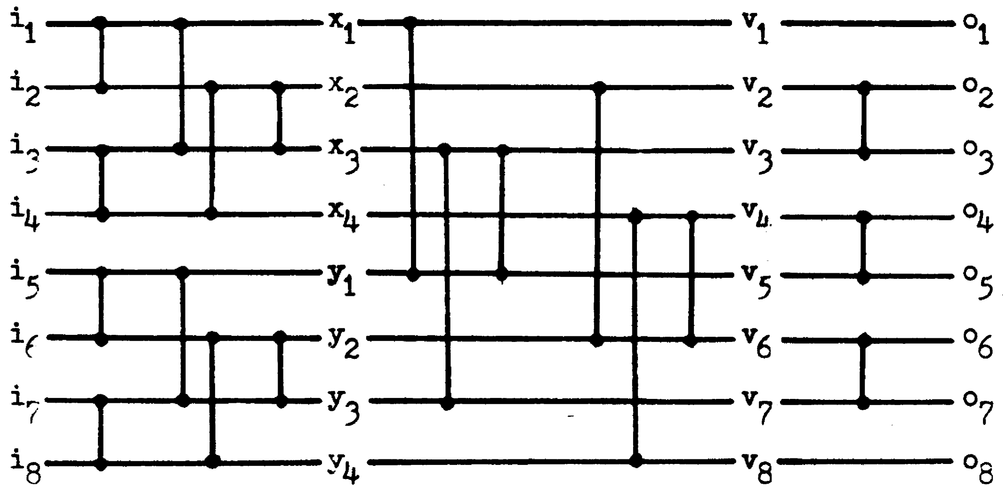


Fig. 7. Batcher's 8-sorter network.

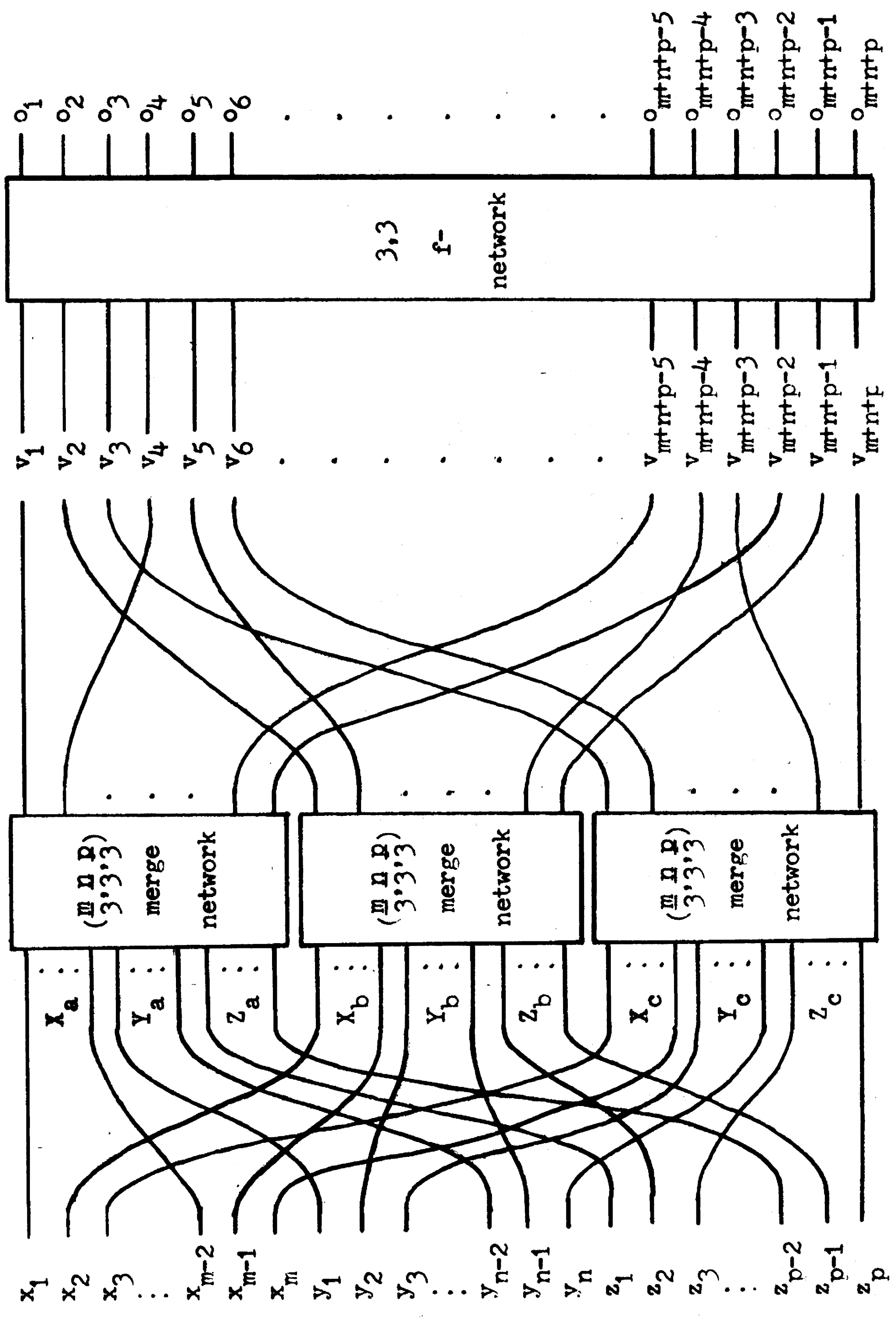
III. The [3,3] Merge Strategy

An obvious extension of the divide-sort-merge strategy as described above is to partition the N inputs into $g \geq 2$ groups that are sorted separately and then merged together. An N-sorter network that uses this g-way divide-sort-merge strategy consists of g sorting networks of size $N_1, N_2, \ldots, N_g$, where $\Sigma_{i=1}^{g} N_i = N$, followed by an $(N_1, N_2, \ldots, N_g)$ merge network. As an extension of Batcher's merge strategy, we can design g-way merge networks that begin with $d \geq 2$ smaller g-way merge networks, where d is a common divisor of $N_1, N_2, \ldots, N_g$.

The two parameters g and d together define what we shall call the [g,d] merge strategy. We say, then, that Batcher's networks described in the last section use the [2,2] strategy.

A [g,d] $(N_1, N_2, \ldots, N_g)$ merge network consists of d $(N_1/d, \ldots, N_g/d)$ merge networks followed by whatever additional comparators are required to complete the merge. We shall call the network comprising these final additional comparators the [g,d] f-network. The [2,2] f-network, namely the comparators listed in part B3 of Batcher's merge network, is particularly simple. In the remainder of this section we illustrate a procedure for designing [g,d] f-networks for arbitrary g,d, by considering the case $g = d = 3$.

Suppose that we wish to design an (m,n,p) merge network that will combine the three sorted multisets $X = \{x_1, x_2, \ldots, x_m\}$, $Y = \{y_1, y_2, \ldots, y_n\}$, and $Z = \{z_1, z_2, \ldots, z_p\}$ into the single sorted multiset $O = \{o_1, o_2, \ldots, o_{m+n+p}\}$. If m, n, and p are all multiples of 3, then the [,3] merge network consists of the following. (See Fig. 8.)



x_1 x_2 x_3 ... x_{m-2} x_{m-1} x_m
y_1 y_2 y_3 ... y_{n-2} y_{n-1} y_n
z_1 z_2 z_3 ... z_{p-2} z_{p-1} z_p
X_a Y_a Z_a X_b Y_b Z_b X_c Y_c Z_c
$(\frac{m}{3},\frac{n}{3},\frac{p}{3})$ merge network
v_1 v_2 v_3 v_4 v_5 v_6
$v_{m+n+p-5}$ $v_{m+n+p-4}$ $v_{m+n+p-3}$ $v_{m+n+p-2}$ $v_{m+n+p-1}$ v_{m+n+p}
3,3 f- network
o_1 o_2 o_3 o_4 o_5 o_6
$o_{m+n+p-5}$ $o_{m+n+p-4}$ $o_{m+n+p-3}$ $o_{m+n+p-2}$ $o_{m+n+p-1}$ o_{m+n+p}

Fig. 8. [3,3] (m,n,p) merge network.

M31: an $(m/3,n/3,p/3)$ merge network that combines $X_a = \{x_1,x_4,\ldots,x_{m-2}\}$, $Y_a = \{y_1,y_4,\ldots,y_{n-2}\}$, and $Z_a = \{z_1,z_4,\ldots,z_{p-2}\}$ to form $V_a = \{v_1,v_4,\ldots,v_{m+n+p-2}\}$;

M32: an $(m/3,n/3,p/3)$ merge network that combines $X_b = \{x_2,x_5,\ldots,x_{m-1}\}$, $Y_b = \{y_2,y_5,\ldots,y_{n-1}\}$, and $Z_b = \{z_2,z_5,\ldots,z_{p-1}\}$ to form $V_b = \{v_2,v_5,\ldots,v_{m+n+p-1}\}$;

M33: an $(m/3,n/3,p/3)$ merge network that combines $X_c = \{x_3,x_6,\ldots,x_m\}$, $Y_c = \{y_3,y_6,\ldots,y_n\}$, and $Z_c = \{z_3, z_6,\ldots,z_p\}$ to form $V_c = \{v_3,v_6,\ldots,v_{m+n+p}\}$; followed by

M34: the [3,3] f-network that we have yet to define.

Now the Zero-One Principle guarantees that, without loss of generality, we may assume that all members of X, Y, and Z are either 0 or 1. (To see that this is so, consider an $(m+n+p)$-sorter network that consists of: an m-sorter that produces the sorted multiset X; an n-sorter that produces Y; and a p-sorter that produces Z; followed by an (m,n,p) merge network.) When all members of X, Y, and Z are either 0 or 1, we find that the number of 0's in V_a, V_b, and V_c satisfies

$$n_c \le n_b \le n_a \le n_c + 3. \tag{5}$$

Therefore, after the three 3-way merges described by M31 through M33, the following situation exists:

1) V_a, V_b, and V_c are each ordered.

2) The first $3n_c$ elements of V are all 0.

3) If $n_a = n_b = n_c$, then the remaining elements of V are all 1; otherwise, the remaining elements exhibit one of the following patterns followed by 1's.

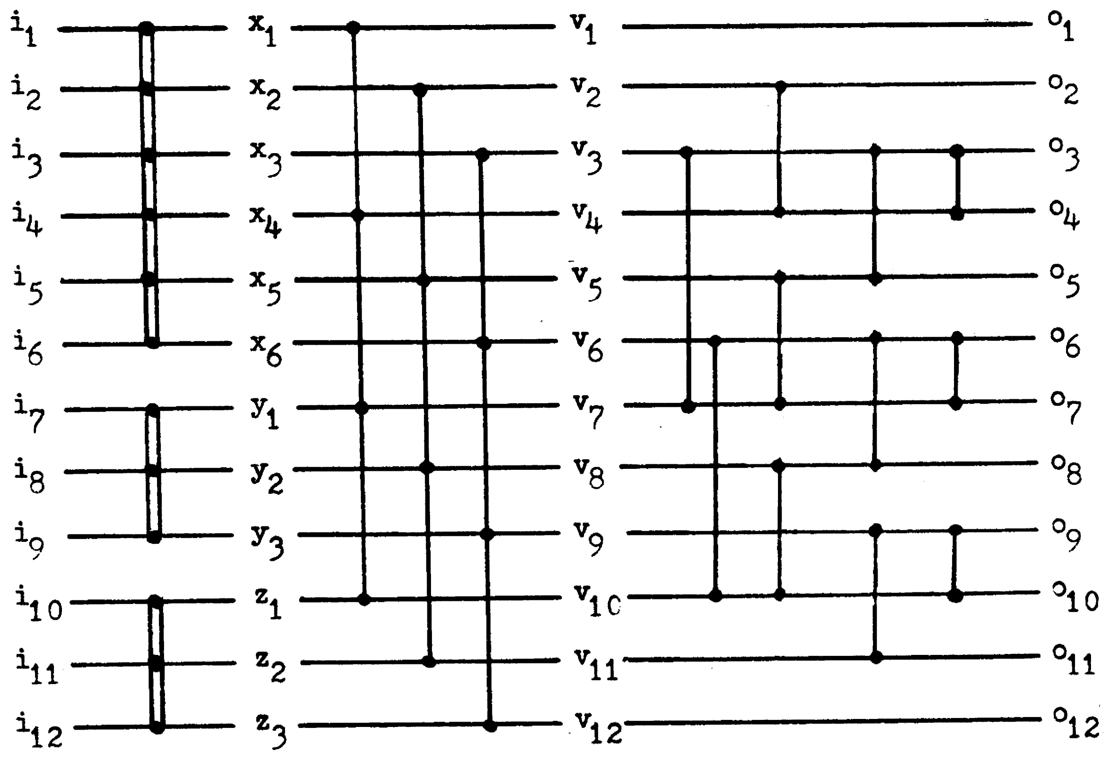
a)	0	if $n_a = n_b + 1 = n_c + 1$;
b)	00	if $n_a = n_b = n_c + 1$;
c)	0110	if $n_a = n_b + 2 = n_c + 2$;
d)	0010	if $n_a = n_b + 1 = n_c + 2$;
e)	00100	if $n_a = n_b = n_c + 2$;
f)	0110110	if $n_a = n_b + 3 = n_c + 3$;
g)	0010110	if $n_a = n_b + 2 = n_c + 3$;
h)	0010010	if $n_a = n_b + 1 = n_c + 3$;
i)	00100100	if $n_a = n_b = n_c + 3$.

It is readily verified that patterns c) through i) are all sorted by the following sequence of comparators.

$$\begin{aligned}
&v_{3n_c+3}:v_{3n_c+7}, && 0 \le n_c \le t-2;\\
&v_{3n_c+2}:v_{3n_c+4}, && 0 \le n_c \le t-1;\\
&v_{3n_c+3}:v_{3n_c+5}, && 0 \le n_c \le t-1;\\
&v_{3n_c+3}:v_{3n_c+4}, && 0 \le n_c \le t-1,
\end{aligned} \qquad (6)$$

where $t = (m+n+p)/3$. These comparators constitute the [3,3] f-network.

The [3,3] strategy is illustrated by the 12-sorter in Fig. 9. The inputs are initially partitioned into the three multisets $\{i_1, i_2, i_3, i_4, i_5, i_6\}$, $\{i_7, i_8, i_9\}$, and $\{i_{10}, i_{11}, i_{12}\}$ that are sorted separately.



[3,3] f-network

Fig. 9. [3,3] 12-sorter network.

The networks required to sort these three multisets are each abbreviated by double vertical lines connecting the appropriate comparands. The (6,3,3) merge network begins with three (2,1,1) merge networks that form V_a, V_b, and V_c. These merge networks, which are abbreviated by a single vertical line, are simply 4-sorters without the initial comparator connecting the pair from X. The remaining 11 comparators constitute the [3,3] f-network defined by (6).

IV. [g,d] Sorting Networks

For every pair of integers $g,d \geq 2$ we can construct N-sorter networks using g small sorting networks followed by a [g,d] merge network which, by definition, begins with d small merge networks. A sorting network that begins with g sorting networks and d merge networks will be called a _[g,d] sorting network_, even if the g sorting networks and d merge networks do not employ the [g,d] strategy internally. For example, the 12-sorter in Fig. 9 is called a [3,3] sorting network regardless of the construction of the initial 6-sorter and the small merge networks.

In order to facilitate the general discussion of [g,d] sorting networks, we adopt the following conventions.

1) The purpose of an N-sorter network is to accept as input the unordered multiset $I = \{i_1, i_2, \ldots, i_N\}$ and to produce as output the sorted multiset $O = \{o_1, o_2, \ldots, o_N\}$, where O is a permutation of I and $o_1 \leq o_2 \leq \cdots \leq o_N$. The Zero-One Principle allows us to assume, without loss of generality, that all members of I are either 0 or 1. We make this assumption throughout the remainder of this paper.

2) The g initial sorting networks, labeled $s_1, s_2, \ldots, s_g$, each operate on an integral multiple of d members of I. The outputs of these g sorting networks together form a partially ordered multiset $X = \{x_1, x_2, \ldots, x_N\}$, where x_1 is the smallest output from s_1, x_2 is the second smallest output from s_1, ..., and x_N is the largest output from s_g.

3) The j^{th} merge network, $1 \leq j \leq d$, operates on $x_{(i-1)d+j}$, $1 \leq i \leq N/d$ to produce $v_{(i-1)d+j}$, $1 \leq i \leq N/d$.

4) The [g,d] f-network operates on V to produce O.

The transformation from the unordered multiset I to the completely ordered multiset O may be summarized by

$$I \xrightarrow[\text{networks}]{\text{g sorting}} X \xrightarrow[\text{networks}]{\text{d merge}} V \xrightarrow{\text{f-network}} O.$$

The [g,d] f-network is defined informally to be any network that will complete the ordering of the intermediate multiset V achieved in the [g,d] N-sorter network, N = td. Before giving a formal definition, let us examine the partial ordering in V. It is convenient to consider V to be a t x d array, where

$$V_{(i,j)} = v_{(i-1)d+j}. \tag{7}$$

The t rows and d columns of V are given by

$$V_{(i,*)} = \bigcup_{1 \leq j \leq d} \left\{ V_{(i,j)} \right\}, \quad 1 \leq i \leq t; \tag{8}$$

$$V_{(*,j)} = \bigcup_{1 \leq i \leq t} \left\{ V_{(i,j)} \right\}, \quad 1 \leq j \leq d. \tag{9}$$

Note that the column $V_{(*,j)}$, $1 \leq j \leq d$, is completely ordered since its t members are the t outputs of m_j, the j^{th} merge network.

If the k^{th} initial sorting network s_k accepts n_k 0's as inputs, then the uniform distribution of the elements of X among the d merge networks guarantees that $\lfloor (n_k+d-j)/d \rfloor$ of these n_k 0's are passed to merge network m_j. Therefore, the total number of 0's that goes into m_j, and into $V_{(*,j)}$, is given by

$$n_{(*,j)} = \sum_{1 \le k \le g} \lfloor (n_k+d-j)/d \rfloor, \quad 1 \le j \le d. \tag{10}$$

We may use Equation (10) to show that

$$n_{(*,d)} \le n_{(*,d-1)} \le \cdots \le n_{(*,1)} \le n_{(*,d)} + g. \tag{11}$$

Equations (4) and (5) are special cases of (11).

We have seen that the d columns $V_{(*,j)}$ are each ordered. The following theorem specifies the remaining partial ordering in V.

Theorem 1:

Consider the Boolean multiset $V = \{v_1, v_2, \ldots, v_N\}$, where $N = td$. Suppose that the d columns $V_{(*,j)}$, given by (9), are each ordered. Then

a) the t rows $V_{(i,*)}$, given by (8), are also each ordered if and only if the number of 0's in $V_{(*,j)}$ satisfies

$$n_{(*,d)} \le n_{(*,d-1)} \le \cdots \le n_{(*,1)}; \tag{12}$$

and

b) the relation $n_{(*,1)} \le n_{(*,d)} + g$ implies that

$$V_{(i,d)} \le V_{(i+g,1)}, \quad 1 \le i \le t-g. \tag{13}$$

Proof:

a) The theorem is illustrated in Fig. 10. Since each column $V_{(*,j)}$ is ordered, the upper $n_{(*,j)}$ elements of $V_{(*,j)}$ - - that is, $V_{(i,j)}$, $1 \leq i \leq n_{(*,j)}$ - - are all 0 and the remaining $t - n_{(*,j)}$ elements are 1. If we draw a line from left to right in V representing the step function $h(j) = t - n_{(*,j)}$, then all elements of V above h are 0, whereas all elements below h are 1. Now the rows $V_{(i,*)}$ are all ordered iff no 1 appears to the left of a 0 in any row. Clearly this is the case iff the line h(j) separating 0's from 1's is non-decreasing, that is, iff

$$t-n_{(*,1)} \leq t-n_{(*,2)} \leq \cdots \leq t-n_{(*,d)}. \tag{14}$$

(See Fig. 10(b).) Equation (14) is equivalent to Equation (12).

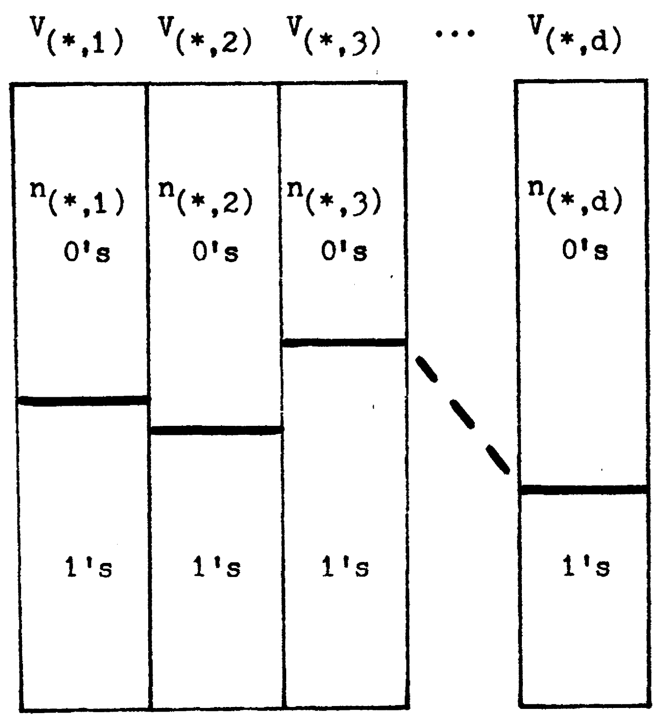
b) Since V is Boolean

$$V_{(i+g,1)} = 1 \quad \Rightarrow \quad V_{(i,d)} \leq V_{(i+g,1)}. \tag{15}$$

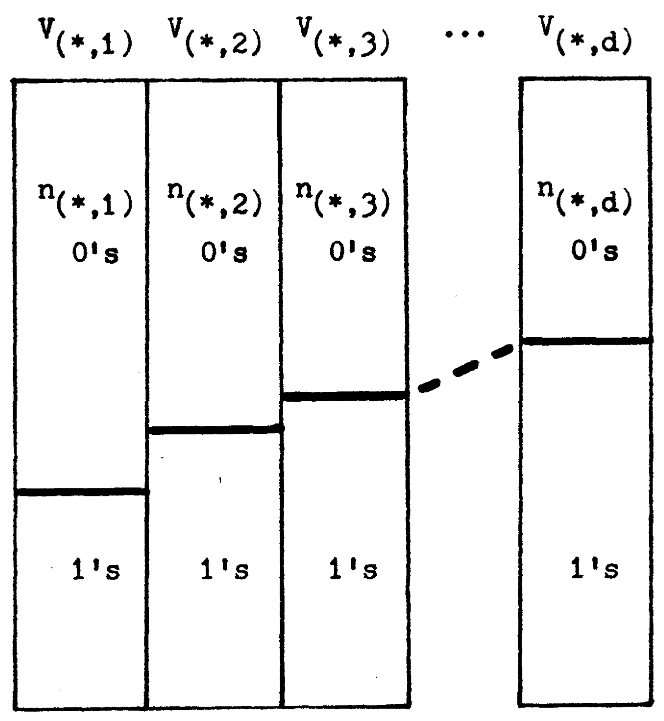
Also, since $V_{(*,j)}$ is ordered,

$$V_{(i,j)} = 0 \quad \Leftrightarrow \quad n_{(*,j)} \geq i. \tag{16}$$

Therefore, if $n_{(*,1)} \leq n_{(*,d)} + g$, then



(a) Columns Ordered.



(b) Rows and Columns Ordered.

Fig. 10. Location of zeros in V.

$$
\begin{aligned}
V_{(i+g,1)} = 0 \quad &\Rightarrow \quad n_{(*,1)} \geq i+g \\
&\Rightarrow \quad n_{(*,d)} \geq i \\
&\Rightarrow \quad V_{(i,d)} = 0 \\
&\Rightarrow \quad V_{(i,d)} \leq V_{(i+g,1)} \qquad (17)
\end{aligned}
$$

Together (15) and (17) imply that $V_{(i,d)} \leq V_{(i+g,1)}$.

Q.E.D.

Since the columns $V_{(*,j)}$ are ordered, and since $n_{(*,j)}$ satisfies (11), Theorem 1 and the transitivity of the relation "less than or equal to" imply the following corollary.

Corollary 1:

Let $V = \{v_1,v_2,\ldots,v_N\}$, $N = td$, be the intermediate multiset achieved by the [g,d] N-sorter network T. Then for any multiset of inputs to T, $V_{(i,j)} \leq V_{(r,s)}$ if

a) $r \geq i$ and $s \geq j$; OR

b) $r \geq i+g$.

The partial ordering in V is illustrated in Fig. 11, for the case $g = 3$, $d = 4$, $t = 6$, with an arrow from $V_{(i,j)}$ to $V_{(r,s)}$ representing the relation $V_{(i,j)} \leq V_{(r,s)}$. R. W. Floyd has pointed out* that the partial ordering in V is exactly characterized by Corollary 1 and Fig. 11.

* Private communication.

$$\begin{tikzcd}
V_{(1,1)} \arrow[r] \arrow[d] & V_{(1,2)} \arrow[r] \arrow[d] & V_{(1,3)} \arrow[r] \arrow[d] & V_{(1,4)} \arrow[d] \arrow[dddlll, bend right] \\
V_{(2,1)} \arrow[r] \arrow[d] & V_{(2,2)} \arrow[r] \arrow[d] & V_{(2,3)} \arrow[r] \arrow[d] & V_{(2,4)} \arrow[d] \arrow[dddlll, bend right] \\
V_{(3,1)} \arrow[r] \arrow[d] & V_{(3,2)} \arrow[r] \arrow[d] & V_{(3,3)} \arrow[r] \arrow[d] & V_{(3,4)} \arrow[d] \arrow[dddlll, bend right] \\
V_{(4,1)} \arrow[r] \arrow[d] & V_{(4,2)} \arrow[r] \arrow[d] & V_{(4,3)} \arrow[r] \arrow[d] & V_{(4,4)} \arrow[d] \\
V_{(5,1)} \arrow[r] \arrow[d] & V_{(5,2)} \arrow[r] \arrow[d] & V_{(5,3)} \arrow[r] \arrow[d] & V_{(5,4)} \arrow[d] \\
V_{(6,1)} \arrow[r] & V_{(6,2)} \arrow[r] & V_{(6,3)} \arrow[r] & V_{(6,4)}
\end{tikzcd}$$

Fig. 11. Partial ordering in V when $g = 3$, $d = 4$, $t = 6$.

By this we mean that if $\hat{V} = \{\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_N\}$ is any Boolean multiset that satisfies the partial ordering specified for V by Corollary 1, then there is at least one combination of inputs to T that achieves $V = \hat{V}$. The sublety of this observation is best illustrated by a partial ordering that is not exactly characterized. Consider the comparator network that results from removing comparators D and E from the 4-sorter in Fig. 1. The partial ordering in the multiset O does not include either $o_2 \leq o_3$ or $o_2 \leq o_4$, since $O(\{0,1,0,1\}) = \{0,1,0,1\}$ and $O(\{1,1,0,0\}) = \{0,1,1,0\}$. However, no combination of inputs will achieve $O = \{0,1,0,0\}$.

We have defined a [g,d] f-network informally as a network that will complete the ordering of the intermediate multiset V achieved in the [g,d] N-sorter network, N = td. The following is a more formal definition.

Definition 1:

A sequence of comparators is called a <u>[g,d] f-network</u> for N = td items if and only if it will complete the ordering of the multiset $V = \{v_1, v_2, \ldots, v_N\}$, when a) the columns $V_{(*,j)}$ of V, given by (9), are each ordered and b) $n_{(*,j)}$ satisfies (11).

We can construct [g,d] f-networks for arbitrary g,d by i) using (11) to determine what unsorted patterns of 0's and 1's remain in V; and ii) finding a sequence of comparators that will order these unsorted patterns. Following this procedure we have derived f-networks for $g,d \leq 4$; the best f-networks obtained are tabulated in Table 1.

Strategy	f-network for N-sorter, $N = td$	$\hat{f}_{[g,d]}(N)$
[2,2]	$V_{(i,2)}:V_{(i+1,1)}$, $1 \leq i \leq t-1$.	$\frac{1}{2}N - 1$
[2,3]	$V_{(i,2)}:V_{(i+1,1)}$, $1 \leq i \leq t-1$; $V_{(i,3)}:V_{(i+1,2)}$, $1 \leq i \leq t-1$; $V_{(i,3)}:V_{(i+1,1)}$, $1 \leq i \leq t-1$.	$N - 3$
[2,4]	$V_{(i,3)}:V_{(i+1,1)}$, $1 \leq i \leq t-1$; $V_{(i,4)}:V_{(i+1,2)}$, $1 \leq i \leq t-1$; $V_{(i,2)}:V_{(i,3)}$, $1 \leq i \leq t$; $V_{(i,4)}:V_{(i+1,1)}$, $1 \leq i \leq t-1$.	$N - 3$
[3,2]	$V_{(i,2)}:V_{(i+1,1)}$, $1 \leq i \leq t-1$; $V_{(i,1)}:V_{(i,2)}$, $2 \leq i \leq t-2$.	$N - 3$
[3,3]	$V_{(i,3)}:V_{(i+2,1)}$, $1 \leq i \leq t-2$; $V_{(i,2)}:V_{(i+1,1)}$, $1 \leq i \leq t-1$; $V_{(i,3)}:V_{(i+1,2)}$, $1 \leq i \leq t-1$; $V_{(i,3)}:V_{(i+1,1)}$, $1 \leq i \leq t-1$.	$\frac{4}{3}N - 5$

Table 1. Small f-networks

Strategy	f-network for N-sorter, $N = td$		$\hat{f}_{[g,d]}(N)$
[3,4]	$V_{(1,4)}:V_{(3,1)}$;		$2N-12$, $N=12$
	$V_{(t-2,4)}:V_{(t,1)}$;		$2N-11$, $N>12$
	$V_{(1,3)}:V_{(1,4)}$;		
	$V_{(3,1)}:V_{(3,2)}$;		
	$V_{(t-2,3)}:V_{(t-2,4)}$;		
	$V_{(t,1)}:V_{(t,2)}$;		
	$V_{(i,3)}:V_{(i+2,1)}$,	$2 \leq i \leq t-3$;	
	$V_{(i,4)}:V_{(i+2,2)}$,	$2 \leq i \leq t-3$;	
	$V_{(i,2)}:V_{(i+1,1)}$,	$1 \leq i \leq t-2$;	
	$V_{(i,4)}:V_{(i+1,3)}$,	$2 \leq i \leq t-1$;	
	$V_{(i,3)}:V_{(i+1,1)}$,	$1 \leq i \leq t-1$;	
	$V_{(i,4)}:V_{(i+1,2)}$,	$1 \leq i \leq t-1$;	
	$V_{(i,2)}:V_{(i,3)}$,	$2 \leq i \leq t-1$;	
	$V_{(i,4)}:V_{(i+1,1)}$,	$1 \leq i \leq t-1$.	
[4,2]	$V_{(i,2)}:V_{(i+2,1)}$,	$1 \leq i \leq t-2$;	$N - 3$
	$V_{(i,2)}:V_{(i+1,1)}$,	$1 \leq i \leq t-1$.	

Table 1. (cont) Small f-networks.

Strategy	f-network for N-sorter, $N = td$	$\hat{f}_{[g,d]}$
[4,3]	$V_{(1,3)}:V_{(4,1)}$; $V_{(t-3,3)}:V_{(t,1)}$; $V_{(i,2)}:V_{(i+2,1)}$, $2 \leq i \leq t-3$; $V_{(i,3)}:V_{(i+2,2)}$, $2 \leq i \leq t-3$; $V_{(i,3)}:V_{(i+2,1)}$, $1 \leq i \leq t-2$; $V_{(i,2)}:V_{(i+1,1)}$, $1 \leq i \leq t-2$; $V_{(i,3)}:V_{(i+1,2)}$, $2 \leq i \leq t-1$; $V_{(i,3)}:V_{(i+1,1)}$, $1 \leq i \leq t-1$; $V_{(2,1)}:V_{(2,2)}$; $V_{(t-1,2)}:V_{(t-1,3)}$.	$2N-12$, $N=12$; $2N-11$, $N>12$.
[4,4]	$V_{(i,3)}:V_{(i+2,1)}$, $1 \leq i \leq t-2$; $V_{(i,4)}:V_{(i+2,2)}$, $1 \leq i \leq t-2$; $V_{(i,2)}:V_{(i+1,1)}$, $1 \leq i \leq t-1$; $V_{(i,4)}:V_{(i+1,3)}$, $1 \leq i \leq t-1$; $V_{(i,3)}:V_{(i+1,1)}$, $1 \leq i \leq t-1$; $V_{(i,4)}:V_{(i+1,2)}$, $1 \leq i \leq t-1$; $V_{(i,2)}:V_{(i,3)}$, $2 \leq i \leq t-1$; $V_{(i,4)}:V_{(i+1,1)}$, $1 \leq i \leq t-1$.	$2N - 11$.

Table 1. (cont) Small f-networks.

Except for the [3,4] and [4,3] f-networks, each of the tabulated f-networks is completely described by a sequence of templates of the form $V_{(i,\alpha)}:V_{(i+\gamma,\beta)}$ -- where $1 \leq \alpha,\beta \leq d$, $\gamma \geq 0$, and $\alpha < \gamma d + \beta$ -- followed by a range for i, which is specified in terms of $t = N/d$. The [3,4] and [4,3] f-networks are described by several specific comparators, in addition to templates. Note that when $N = 12$, half of these specific comparators are redundant and may be eliminated. For example, the second comparator listed for the [3,4] f-network, namely $V_{(t-2,4)}:V_{(t,1)}$, becomes $V_{(1,4)}:V_{(3,1)}$, which is the same as the first comparator listed.

Let $f_{[g,d]}(N)$ represent the minimum number of comparators required by a [g,d] f-network for N items. (Note that this function is only defined when N is a multiple of d.) Since we have not proved that the tabulated f-networks are minimal, we have labeled the number of comparators that they require $\hat{f}_{[g,d]}(N)$. For each of the tabulated f-networks, except the [3,4] and [4,3] f-networks, we find that

$$\hat{f}_{[g,d]}(N) = a_{[g,d]}N - b_{[g,d]}, \tag{18}$$

where $a_{[g,d]}$ is $(1/d)$ times the number of templates and $b_{[g,d]}$ is constant. The tabulated [3,4] and [4,3] f-networks are also described by (18) for $N > 12$.

For large g,d it becomes increasingly difficult to derive an economical [g,d] f-network, since the number of patterns of 0's and 1's allowed by (11) increases rapidly. Let $P(g,d)$ represent the number of patterns of 0's and 1's consistent with (11), that is, the number of different combinations of values that $n_{(*,1)}, n_{(*,2)}, \ldots, n_{(*,d-1)}$ can assume for each value of $n_{(*,d)}$. With $n_{(*,j)}$ abbreviated by n_j, we observe that

$$P(g,d) = \sum_{n_d \le n_1 \le n_d+g} \; \sum_{n_d \le n_2 \le n_1} \cdots \sum_{n_d \le n_{d-1} \le n_{d-2}} 1. \tag{19}$$

We may obtain a recurrence relation for $P(g,d)$ by noting that

$$P(g,d) = \sum_{n_d \le n_1 \le n_d+g-1} \; \sum_{n_d \le n_2 \le n_1} \cdots \sum_{n_d \le n_{d-1} \le n_{d-2}} 1$$

$$+ \sum_{n_d \le n_2 \le n_d+g} \cdots \sum_{n_d \le n_{d-1} \le n_{d-2}} 1$$

$$= P(g-1,d) + P(g,d-1). \tag{20}$$

The solution to (20), with the boundary conditions $P(1,d) = d$, $P(g,1) = 1$, is simply

$$P(g,d) = \binom{g+d-1}{d-1}. \tag{21}$$

Note that (21) yields $P(2,2) = 3$ and $P(3,3) = 10$, which agrees with our analysis of the [2,2] and [3,3] merge networks.

When $N \geq gd$, the problem of designing an f-network that will order $P(g,d)$ patterns of 0's and 1's represents a considerable reduction of the original problem of designing an N-sorter network that will order 2^N different input patterns. However, for large g,d, we find that $P(g,d)$

becomes too large to permit an exhaustive test of a proposed design for a [g,d] f-network. Therefore, for large g,d we must build f-networks in such a way that we can prove "by construction" that they complete the ordering of V. Suitable procedures for constructing large [g,d] f-networks are given in the next section.

V. Constructing Large [g,d] f-networks

Our approach to the problem of deriving large sorting networks and large [g,d] merge networks is to build them out of smaller sorting networks and smaller merge networks. We use the same approach to the problem of designing large [g,d] f-networks. We will present two construction methods in the form of theorems. Theorem 2 below describes a procedure for constructing a [g,sd] f-network using d small [g,s] f-networks and one [g,d] f-network. Theorem 3 describes a similar procedure for building an [sg,d] f-network out of s small [g,d] and one [s,d] f-networks. We may use these constructions and the f-networks given in Table 1 to achieve f-networks for arbitrarily large g,d.

Before giving the theorems, we will describe an example. Suppose we desire to construct a [3,6] f-network for the [3,6] 18-sorter network. The partial ordering in the intermediate multiset V is depicted in Fig. 12(a). In Fig. 12(b) we have isolated the partial ordering in the even members of V. Clearly a [3,3] f-network will order V_e; similarly, another [3,3] f-network will order V_o.

The partial ordering depicted in 12(a) guarantees (by Theorem 1) that

$$n_{(*,6)} \leq n_{(*,5)} \leq \cdots \leq n_{(*,1)} \leq n_{(*,6)} + 3. \tag{22}$$

The number of 0's in V_o and V_e are given by

$$n_o = n_{(*,1)} + n_{(*,3)} + n_{(*,5)}; \tag{23}$$

$$n_e = n_{(*,2)} + n_{(*,4)} + n_{(*,6)},$$

$$\begin{tikzcd}
v_1 \arrow[r] \arrow[d] & v_2 \arrow[r] \arrow[d] & v_3 \arrow[r] \arrow[d] & v_4 \arrow[r] \arrow[d] & v_5 \arrow[r] \arrow[d] & v_6 \arrow[d] \\
v_7 \arrow[r] \arrow[d] & v_8 \arrow[r] \arrow[d] & v_9 \arrow[r] \arrow[d] & v_{10} \arrow[r] \arrow[d] & v_{11} \arrow[r] \arrow[d] & v_{12} \arrow[d] \\
v_{13} \arrow[r] & v_{14} \arrow[r] & v_{15} \arrow[r] & v_{16} \arrow[r] & v_{17} \arrow[r] & v_{18}
\end{tikzcd}$$

(a) V for [3,6] 18-sorter.

$$\begin{tikzcd}
v_2 \arrow[r] \arrow[d] & v_4 \arrow[r] \arrow[d] & v_6 \arrow[d] \\
v_8 \arrow[r] \arrow[d] & v_{10} \arrow[r] \arrow[d] & v_{12} \arrow[d] \\
v_{14} \arrow[r] & v_{16} \arrow[r] & v_{18}
\end{tikzcd}$$

(b) Partial ordering in V_e.

Fig. 12.

so that

$$0 \leq (n_{(*,5)} - n_{(*,6)}) + (n_{(*,3)} - n_{(*,4)}) + (n_{(*,1)} - n_{(*,2)})$$

$$= n_o - n_e \leq n_{(*,1)} - n_{(*,6)} \leq 3. \qquad (24)$$

Therefore, a [3,2] f-network will complete the ordering of V, once V_e and V_o have each been ordered.

Since the two small [3,3] and one full-sized [3,2] f-network will complete the ordering of V, they together constitute a [3,6] f-network. The resulting [3,6] 18-sorter network is given in Fig. 13.

For Theorems 2 and 3 below it is convenient to consider the multiset $V = \{v_1, v_2, \ldots, v_N\}$, $N = pqr$, to be a $p \times q \times r$ array, where

$$V_{(i,j,k)} = v_{(i-1)qr+(j-1)r+k}. \qquad (25)$$

Submultisets of V include the pq "rows," pr "columns," and qr "verticals" defined, respectively, by

$$V_{(i,j,*)} = \bigcup_{1 \leq k \leq r} \left\{ V_{(i,j,k)} \right\}, \quad 1 \leq i \leq p,\ 1 \leq j \leq q;$$

$$V_{(i,*,k)} = \bigcup_{1 \leq j \leq q} \left\{ V_{(i,j,k)} \right\}, \quad 1 \leq i \leq p,\ 1 \leq k \leq r; \qquad (26)$$

$$V_{(*,j,k)} = \bigcup_{1 \leq i \leq p} \left\{ V_{(i,j,k)} \right\}, \quad 1 \leq j \leq q,\ 1 \leq k \leq r.$$

Larger submultisets of V include the p $q \times r$ "planes", the q $p \times r$ planes, and the r $p \times q$ planes defined by

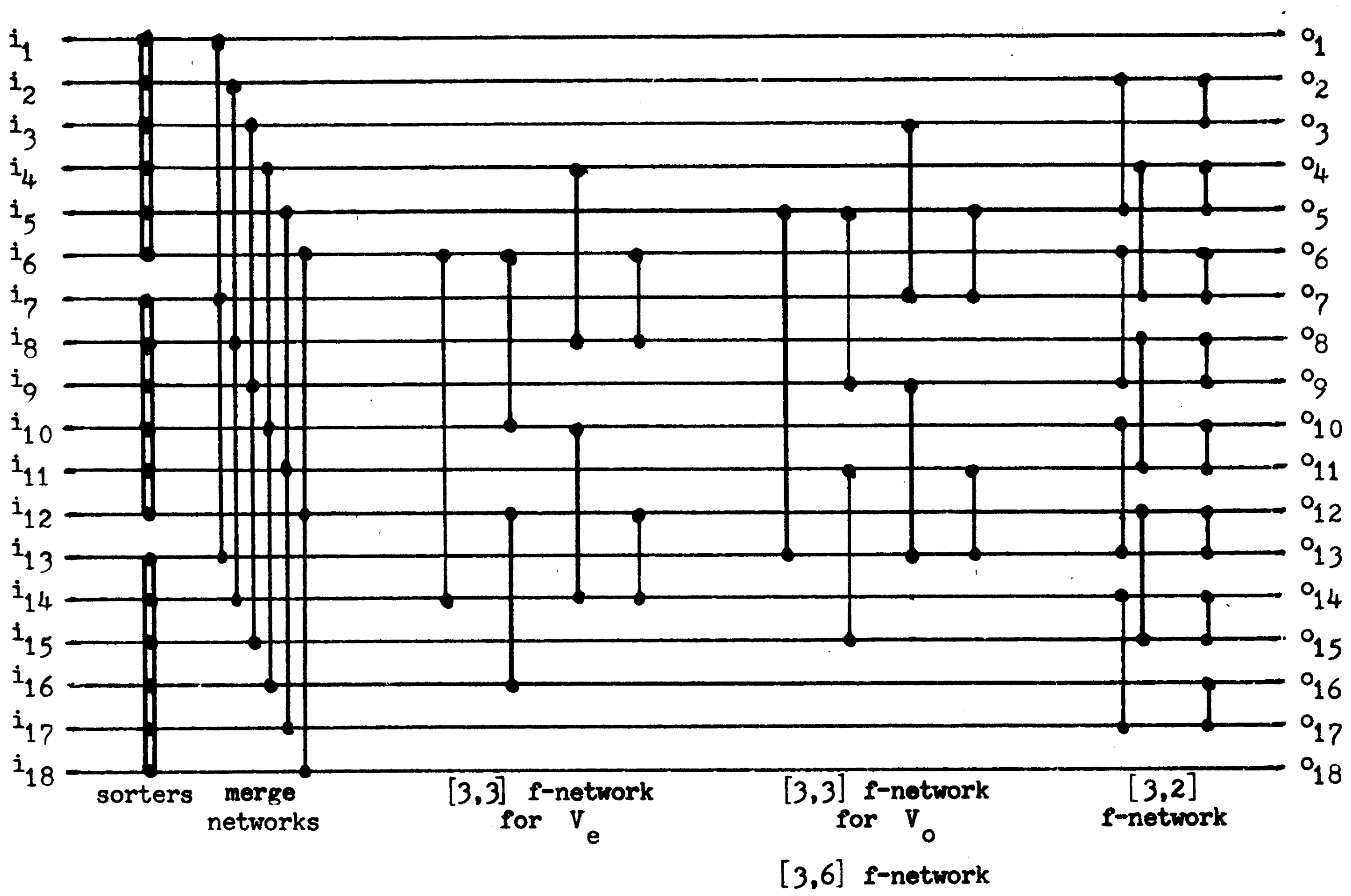


Fig. 13. [3,6] 18-sorter network.

$$V_{(*,*,k)} = \bigcup_{1 \leq i \leq p} \; \bigcup_{1 \leq j \leq q} \left\{ V_{(i,j,k)} \right\}, \quad 1 \leq k \leq r;$$

$$V_{(*,j,*)} = \bigcup_{1 \leq i \leq p} \; \bigcup_{1 \leq k \leq r} \left\{ V_{(i,j,k)} \right\}, \quad 1 \leq j \leq q; \qquad (27)$$

$$V_{(i,*,*)} = \bigcup_{1 \leq j \leq q} \; \bigcup_{1 \leq k \leq r} \left\{ V_{(i,j,k)} \right\}, \quad 1 \leq i \leq p.$$

For example, if we consider the intermediate multiset V for the [3,6] 18-sorter (Fig. 12) to be a $3 \times 3 \times 2$ array, then

$$V_{(1,2,1)} = v_3;$$

$$V_{(*,2,1)} = \{v_3, v_9, v_{15}\}; \qquad (28)$$

$$V_{(*,*,1)} = V_o.$$

We are now ready for Theorems 2 and 3.

Theorem 2:

Let the multiset $V = \{v_1, v_2, \ldots, v_N\}$, where $N = tsd$, be considered a $t \times s \times d$ array. Then the following small f-networks together constitute a [g,sd] f-network for V.

i) d [g,s] f-networks for $V_{(*,*,k)}$, $1 \leq k \leq d$; followed by

ii) one [g,d] f-network for V.

Proof:

According to Definition 1, the sequence of comparators represented by i) and ii) is a [g,sd] f-network for V if and only if it will complete the ordering of V given that: a) the planes $V_{(*,j,k)}$, $1 \leq j \leq s$, $1 \leq k \leq d$, are ordered; and b) $n_{(*,j,k)}$ satisfies

$$n_{(*,s,d)} \leq n_{(*,s,d-1)} \leq \cdots \leq n_{(*,s,1)} \leq n_{(*,s-1,d)} \leq$$
$$n_{(*,s-1,d-1)} \leq \cdots \leq n_{(*,1,1)} \leq n_{(*,s,d)} + g. \qquad (29)$$

Let us assume that the partial ordering in V satisfies conditions a) and b). Then since the submultisets $V_{(*,j,k)}$ of $V_{(*,*,k)}$ are ordered and since $n_{(*,j,k)}$ satisfies (29) a [g,s] f-network will order $V_{(*,*,k)}$. Now the number of 0's in $V_{(*,*,k)}$ (both before and after the application of the [g,s] f-network) is given by

$$n_{(*,*,k)} = \sum_{1 \leq j \leq s} n_{(*,j,k)} . \qquad (30)$$

For any two indices k_1, k_2 satisfying $1 \leq k_1 < k_2 \leq d$, we may use (29) to show that

$$0 \leq \sum_{1 \leq j \leq s} (n_{(*,j,k_1)} - n_{(*,j,k_2)}) = n_{(*,*,k_1)} - n_{(*,*,k_2)}$$
$$\leq n_{(*,1,1)} - n_{(*,s,d)} \leq g . \qquad (31)$$

Therefore, once the [g,s] f-networks have ordered the planes $V_{(*,*,k)}$, (31) guarantees that a [g,d] f-network will complete the ordering of V.

We have seen that if the partial ordering in V satisfies conditions a) and b), then the d $[g,s]$ f-networks in i) followed by the $[g,d]$ f-network in ii) will complete the ordering of V. Therefore, i) and ii) together constitute a $[g,sd]$ f-network.

Q.E.D.

Theorem 3:

Let V be as in Theorem 2. Then the following small f-networks together constitute an $[sg,d]$ f-network for V.

i) s $[g,d]$ f-networks for $V_{(*,j,*)}$, $1 \leq j \leq s$: followed by

ii) one $[s,d]$ f-network for V.

Proof:

The proof of Theorem 3 is similar to that for Theorem 2 and is given in Appendix A.

The partial ordering in the intermediate multiset V for the $[3\cdot 2,3]$ 18-sorter is given in Fig. 14. The construction method described by Theorem 3 requires a $[3,3]$ f-network connecting the three odd rows of V $(V_{(*,1,*)})$ and a $[3,3]$ f-network for the even rows $(V_{(*,2,*)})$, followed by a $[2,3]$ f-network. The resulting $[6,3]$ 18-sorter network is given in Fig. 15.

We many count the comparators required by the f-networks constructed according to Theorems 2 and 3 to obtain the following important corollary.

$$\begin{tikzcd}
v_1 \arrow[r] \arrow[d] & v_2 \arrow[r] \arrow[d] & v_3 \arrow[d] \\
v_4 \arrow[r] \arrow[d] & v_5 \arrow[r] \arrow[d] & v_6 \arrow[d] \\
v_7 \arrow[r] \arrow[d] & v_8 \arrow[r] \arrow[d] & v_9 \arrow[d] \\
v_{10} \arrow[r] \arrow[d] & v_{11} \arrow[r] \arrow[d] & v_{12} \arrow[d] \\
v_{13} \arrow[r] \arrow[d] & v_{14} \arrow[r] \arrow[d] & v_{15} \arrow[d] \\
v_{16} \arrow[r] & v_{17} \arrow[r] & v_{18}
\end{tikzcd}$$

Fig. 14. V for the [6,3] 18-sorter.

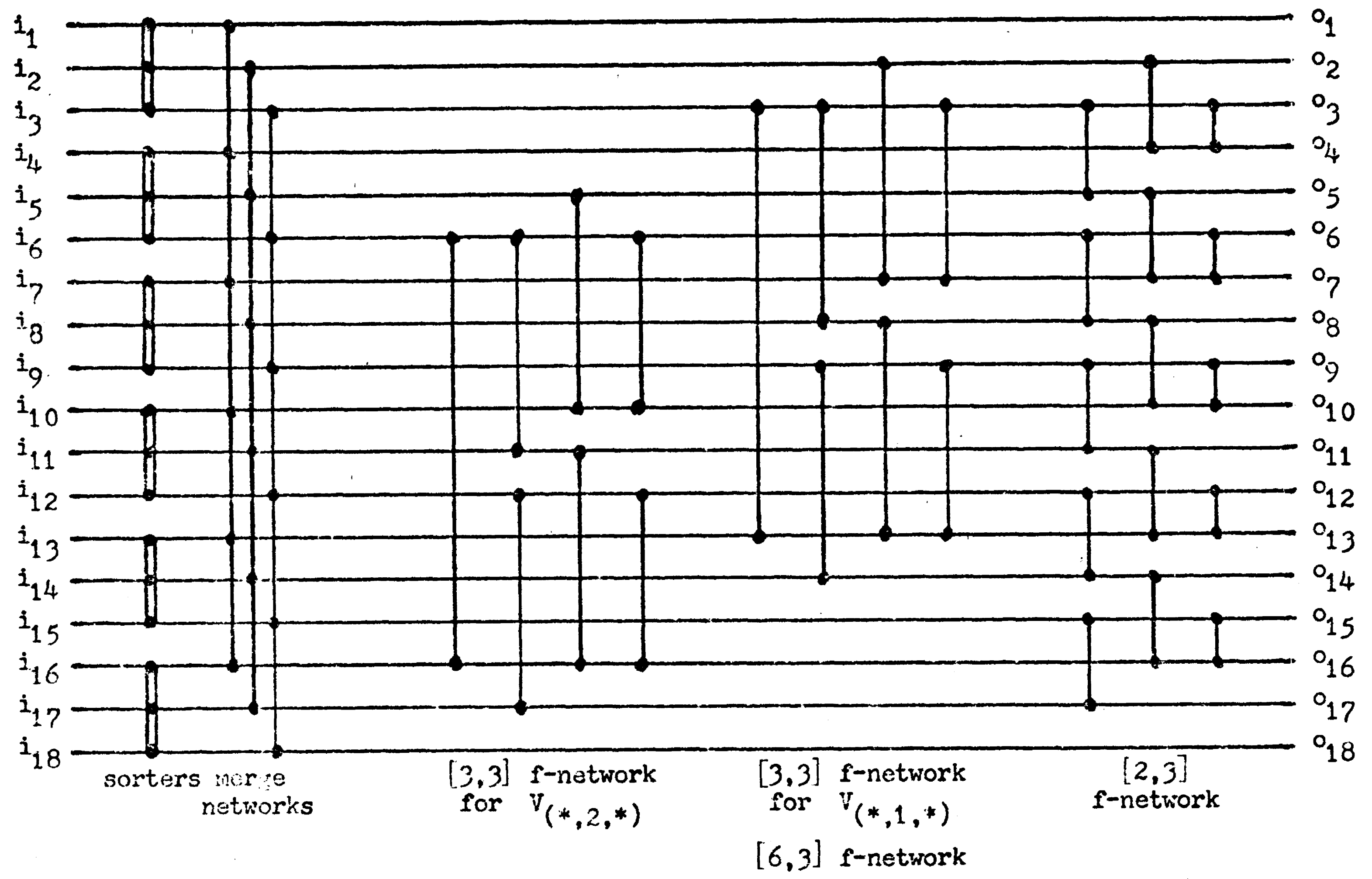


Fig. 15. [6,3] 18-sorter network.

Corollary 2:

$$f_{[g,sd]}(N) \leq d \cdot f_{[g,s]}(N/d) + f_{[g,d]}(N); \qquad (32)$$

$$f_{[sg,d]}(N) \leq s \cdot f_{[g,d]}(N/s) + f_{[s,d]}(N). \qquad (33)$$

The inequality is required in Corollary 2 since we have no guarantee that an f-network constructed using Theorem 2 or Theorem 3 is minimal. In fact, the [3,6] f-network exhibited by the 18-sorter in Fig. 13 is not the most economical [3,6] f-network known. We may use Theorem 2 with s = 2, d = 3, to build a [3,6] f-network out of three [3,2] and one [3,3] f-networks. Using Table 1 we see that this f-network requires

$$\hat{f}_{[3,6]}(N) = 3\,\hat{f}_{[3,2]}(N/3) + \hat{f}_{[3,3]}(N) = \frac{7}{3}N - 14 \qquad (34)$$

comparators, whereas the f-network in Fig. 13 requires $\frac{7}{3}N - 13 = 29$ comparators. (However, a slight modification of the [6,3] f-network illustrated in Fig. 13 reduces the number of comparators to $\frac{7}{3}N - 15$.)

The number of comparators required by the best f-network that can be constructed out of smaller f-networks using the construction of Theorem 2 and/or Theorem 3 is neatly summarized by

$$\hat{f}_{[g,d]}(N) = \min_{\substack{1 \leq q < g \\ g \bmod q = 0}} \quad \min_{\substack{1 \leq p < d \\ 2 < q + p \\ d \bmod p = 0}} \left\{ F(g,d,N,q,p) \right\}, \qquad (35$$

where

$$F(g,d,N,q,p) = q \cdot p \cdot \hat{f}_{[g/q,d/p]}(N/(q \cdot p)) + \hat{f}_{[q,p]}(N)$$
$$+ q \cdot \hat{f}_{[g/q,p]}(N/q) + p \cdot \hat{f}_{[q,d/p]}(N/p) . \qquad (36)$$

Note that $\hat{f}_{[g,1]}(N) = \hat{f}_{[1,d]}(N) = 0$, so that: a) if $q = 1$, then (36) describes a construction that uses only Theorem 2; b) if $p = 1$ then (36) describes the use of Theorem 3 alone; and c) if $p,q > 1$, then (36) describes a network built using both theorems. The case $p = q = 1$ is not allowed, since it would reduce (35) to an identity.

We may use the construction methods of Theorems 2 and 3, along with the f-networks in Table 1, to achieve [g,d] f-networks for all g,d of the form $2^i 3^j$.* When g and/or d have prime factors greater than 3, we may construct a [g,d] f-network as follows: Using Batcher's general method we obtain a [2,d] (2d)-sorter network. This (2d)-sorter will, of course, exhibit all of the templates required by any [2,d] network. We may use the [2,d] f-network and Theorem 3 to derive a $[2^i,d]$ f-network, where $g \leq 2^i$. From Definition 1 it is clear that a $[2^i,d]$ f-network is also a [g,d] f-network for all values of $g \leq 2^i$. (This is because the unsorted patterns remaining in the intermediate set V for the [g,d] sorter are a subset of those remaining in the $[2^i,d]$ network, if $g \leq 2^i$.) Therefore, we may construct [g,d] f-networks for arbitrary g,d.

We will conclude this section by calculating the number of comparators required by the $[g^i,d^j]$ f-network constructed using Theorems 2 and 3. From Equation (35) we obtain

* Note that these construction techniques are illustrated by networks in Table 1: the [2,4] f-network illustrates Theorem 2, while the [4,2] f-network illustrates Theorem 3.

$$\hat{f}_{[g^i,d^j]} = \min_{0 \le r < i} \; \min_{\substack{0 \le s < j \\ 0 < r+s}} \left\{ F(g,d,N,g^r,d^s) \right\}. \tag{37}$$

Since $\hat{f}_{[g,d]}(N)$ is linear in N for all of the tabulated f-networks, we expect a solution to (37) of the form

$$\hat{f}_{[g^i,d^j]}(N) = a_{[g^i,d^j]} N - b_{[g^i,d^j]}. \tag{38}$$

Using (36) and (38) in (37) we obtain

$$a_{[g^i,d^j]} = \min_{0 \le r < i} \; \min_{\substack{0 \le s < j \\ 0 < r+s}} \left\{ a_{[g^{i-r},d^{j-s}]} + a_{[g^r,d^s]} + a_{[g^{i-r},d^s]} + a_{[g^r,d^{j-s}]} \right\}; \tag{39}$$

$$b_{[g^i,d^j]} = \max_{0 \le r < i} \; \max_{\substack{0 \le s < j \\ 0 < r+s}} \left\{ g^r d^s b_{[g^{i-r},d^{j-s}]} + b_{[g^r,d^s]} + g^r b_{[g^{i-r},d^s]} + d^s b_{[g^r,d^{j-s}]} \right\}. \tag{40}$$

Equations (38)-(40) describe the number of comparators required by a $[g^i,d^j]$ f-network built out of smaller f-networks using Theorem 2 and/or Theorem 3. Most of the best $[g^i,d^j]$ f-networks known exhibit this construction and are, therefore, described by (38)-(40).

For many values of g,d, all of the best $[g^r,d^s]$ f-networks known $(0 \leq r < i\ ,\ 0 \leq s < j,\ 0 < r + s)$ are constructed from the $[g,d]$ f-network by repeated use of Theorems 2 and 3. In this case the solutions to (39) and (40) are

$$a_{[g^i,d^j]} = i \cdot j \cdot a_{[g,d]} \ ; \tag{41}$$

$$b_{[g^i,d^j]} = \frac{(g^i-1)(d^j-1)}{(g-1)(d-1)} b_{[g,d]} \ . \tag{42}$$

From Table 1 we observe that $a_{[4,4]} = 2$, $b_{[4,4]} = 11$, whereas from Equations (41) and (42) (evaluated with $g = d = i = j = 2$) we find that the $[4,4]$ f-network constructed from the $[2,2]$ f-network according to Theorems 2 and 3 requires $a_{[4,4]} = 4a_{[2,2]} = 2$, $b_{[4,4]} = 9b_{[2,2]} = 9$. The $[4,4]$ f-network given in Table 1 is the smallest example of a special procedure which has been discovered for constructing $[2^k,2^k]$ f-networks [7]. The special procedure is too complicated to include in this paper. Basically, it requires; a) determining the templates required by the $[2^k,2^k]$ f-network derived using Theorems 2 and 3; and b) reordering these templates in such a manner that, although the resulting network still orders V, some of the comparators have become "redundant" and may be removed. Since the special construction does not reduce the number of templates, $a_{[2^k,2^k]}$ is given by (41), evaluated with $g = d = 2$ and $i = j = k$. However, the constant term is increased to

$$b_{[2^k,2^k]} = \frac{4}{3} 4^k - 3\ 2^k + \frac{5}{3} , \tag{43}$$

which represents a reduction (since $b_{[g,d]}$ in (38) is preceded by a minus sign) of $\sim\frac{1}{3}4^k$ comparators. When $g = d = 2$ and $i \neq j$, the best $[2^i,2^j]$ f-networks known use the special $[2^k,2^k]$ f-networks as building blocks for the construction methods described by Theorems 2 and 3. The coefficient $b_{[2^i,2^j]}$ is obtained from (40).

VI. The Economy of [g,d] N-sorter Networks

We have defined a [g,d] N-sorter network to consist of g sorting networks of size $N_1,N_2,\ldots,N_g$, where N_i is an integral multiple of d and $\Sigma_{i=1}^{g} N_i = N$, followed by a $[g,d]$ $(N_1,N_2,\ldots,N_g)$ merge network. Since N_i is required to be a multiple of d, we cannot construct a [g,d] N-sorter network unless N is a multiple of d. This limitation, which was included since it greatly simplifies the description of [g,d] merge networks and [g,d] f-networks, can be removed. In [3 , 5] a procedure is given for <u>pruning</u> an N-sorter network, that is eliminating one input lead, one output lead, and several comparators, to achieve an (N-1)-sorter network. For arbitrary N we can use the [g,d] strategy to achieve an N-sorter network by 1) deriving the $[g,d]$ $(d\lceil N/d\rceil)$-sorter network and 2) pruning as necessary. If we extend the definition of a [g,d] N-sorter network to include the sorting networks achieved by pruning a [g,d] sorting network, then for all values of N <u>except</u> $N = 10,13,14,15,16$, or 18, the most economical N-sorter known is a [g,d] sorting network.

We can also use pruning to achieve a $[g,d]$ $(N_1,N_2,\ldots,N_g)$ merge network when not all of the N_i are integral multiples of d. Let $M_{[g,d]}(N_1,N_2,\ldots,N_g)$ represent the number of comparators contained in the $(N_1,N_2,\ldots,N_g)$ merge network achieved by pruning (if necessary) the $[g,d]$ $(d\lceil N_1/d\rceil,d\lceil N_2/d\rceil,\ldots,d\lceil N_g/d\rceil)$ merge network. Then the minimum number of comparators required by a $(N_1,N_2,\ldots,N_g)$ merge network constructed using any [g,d] strategy is given by

$$M_g(N_1,N_2,\ldots,N_g) = \min_{d} M_{[g,d]}(N_1,N_2,\ldots,N_g). \qquad (44)$$

It is instructive to ask which values of g and d yield the most economical N-sorter networks. Let $S_g(N)$ represent the number of comparators required by the most economical N-sorter that uses g sorting networks followed by a $[g,d]$ merge network. In order to permit a valid comparison of networks achieved with different values of g, we will require that each of the g initial sorting networks must itself use the g-way divide-sort-merge strategy, so that $S_g(N)$ satisfies the recurrence relation

$$S_g(N) = \min_{\substack{N_1+\ldots+N_g=N \\ N_i \geq 1}} \{ M_g(N_1,\ldots,N_g) + \sum_{1 \leq i \leq g} S_g(N_i)\}. \qquad (45)$$

We have calculated $S_g(N)$ for $g = 2,3,$ and 4 and $N \leq 36$; the results are given in Table 2. The last column, labeled $\hat{S}(N)$, gives the number of comparators contained in the most economical N-sorter known of any construction. An asterisk indicates those values of $\hat{S}(N)$ which represent an improvement* over the most economical networks previously reported [5].

From Table 2 we observe that $S_3(N)$ is only occasionally smaller than Batcher's result, $B(N) = S_2(N)$. However, $S_4(N) < S_2(N)$ for all $N > 8$, and the $[4,d]$ N-sorter networks are more economical than any previous N-sorter, for $N > 34$.

* The improved 18-sorter, which does not use a [g,d] strategy, is given in Fig. 16. The improved 26-,27-,28-, and 34-sorters all use two initial sort units, one of them the particularly efficient 16-sorter designed by M. W. Green, followed by Batcher's [2,2] merge network. The best 35-sorter is achieved by pruning one lead from the [4,9] 36-sorter; the best 36-sorter uses the [3,12] strategy.

N	g = 2	g = 3	g = 4	$\hat{S}(N)$
2	1	1	1	1
3	3	3	3	3
4	5	6	5	5
5	9	9	9	9
6	12	12	12	12
7	16	17	16	16
8	19	21	19	19
9	26	25	25	25
10	31	32	30	29
11	37	37	35	35
12	41	42	39	39
13	48	51	47	46
14	53	57	52	51
15	59	62	57	56
16	63	70	61	60
17	74	76	73	73
18	82	81	80	79*

Table 2. $S_g(N)$ for $g \leq 4$, $N \leq 36$.

N	g = 2	g = 3	g = 4	$\hat{S}(N)$
19	91	93	89	88
20	97	101	95	93
21	107	108	104	103
22	114	117	110	110
23	122	125	118	118
24	127	131	123	123
25	138	141	135	134
26	146	148	143	141*
27	155	154	151	150*
28	161	168	157	156*
29	171	178	168	166
30	178	187	174	172
31	186	197	182	180
32	191	207	187	185
33	207	214	203	203
34	219	226	214	213*
35	232	234	225	225*
36	241	241	233	232*

Table 2 (cont.)

* This value of $\hat{S}(N)$ describes an N-sorter network that is more economical than any N-sorter previously reported [5].

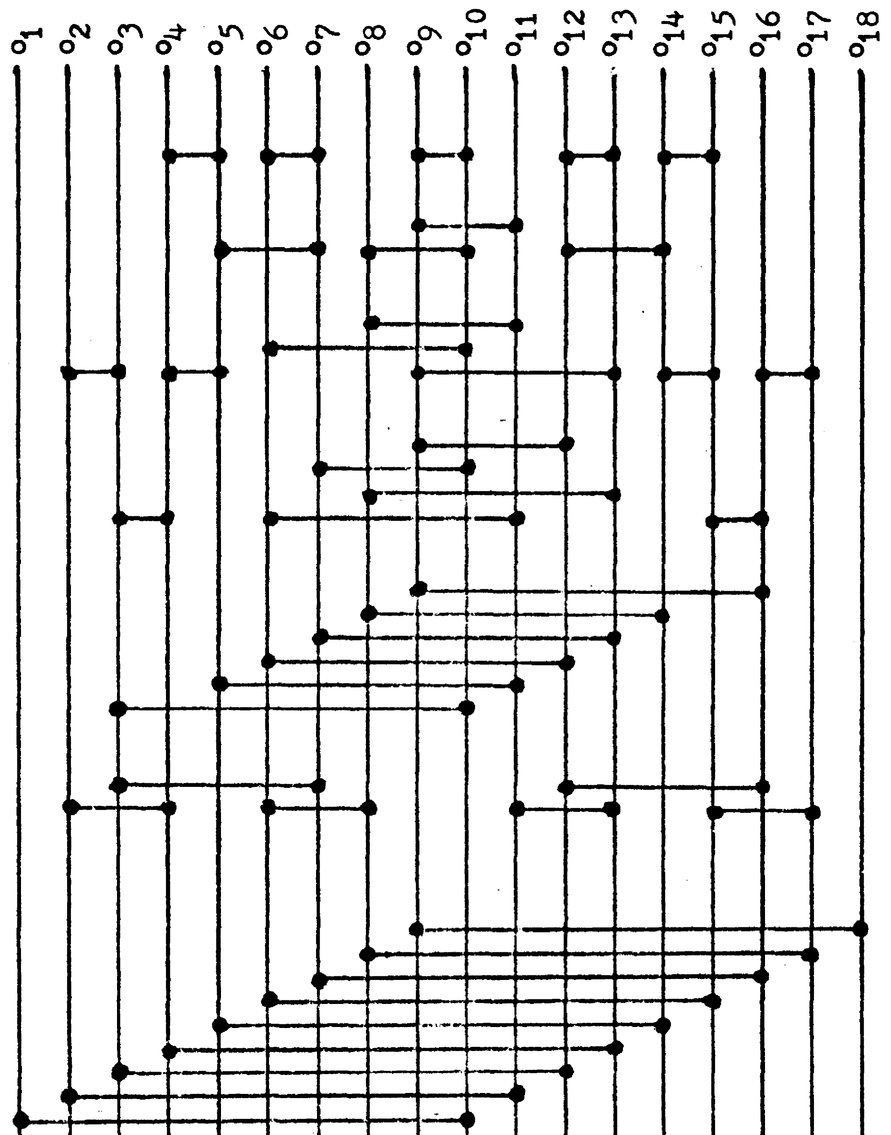


Fig. 16. Best 18-sorter network known.

We may discover the asymptotic growth of $S_g(N)$ by considering the case $N = g^{k+1}$. For all values of g tried we have found that the minimum in the right-hand-side of (45) occurs when $N_1 = N_2 = \dots = g^k$, so that

$$S_g(g^{k+1}) = g\, S_g(g^k) + M_g(g^k, g^k, \dots, g^k). \tag{46}$$

We have also found that the most economical $[g,d]$ $(g^k, g^k, \dots, g^k)$ merge network known is achieved using $d = g$, so that

$$\begin{aligned} M_g(g^k, \dots, g^k) &= M_{[g,g]}(g^k, \dots, g^k) \\ &= g\, M_g(g^{k-1}, \dots, g^{k-1}) + \hat{f}_{[g,g]}(g^{k+1}). \end{aligned} \tag{47}$$

The solutions to the recurrence relations given in (47) and (46), with the boundary condition $S_g(g) = M_g(1,1,\dots,1) = \hat{S}(N)$, are

$$M_g(g^k, \dots, g^k) = (2\alpha_g k + \alpha_g + \beta_g)\, g^{k+1} + (g-1)\gamma_g; \tag{48}$$

$$S_g(g^k) = (\alpha_g k^2 + \beta_g k + \gamma_g)\, g^k - \gamma_g, \tag{49}$$

where

$$\alpha_g = \tfrac{1}{2} a_{[g,g]}, \tag{50}$$

$$\beta_g = g^{-1}\, \hat{S}(g) - \tfrac{1}{2} a_{[g,g]} - g^{-1}(g-1)^{-1}\, b_{[g,g]}, \tag{51}$$

$$\gamma_g = (g-1)^{-2} b_{[g,g]}. \tag{52}$$

The asymptotic growth of $S_g(N)$ may be obtained from (49); it is given by

$$\begin{aligned} S_g(N) &= \alpha_g N(\log_g N)^2 + \beta_g N(\log_g N) + \gamma_g N + O(1) \\ &= \alpha_g (\log_2 g)^{-2} N(\log_2 N)^2 + \\ &\qquad \beta_g (\log_2 g)^{-1} N(\log_2 N) + \gamma_g N + O(1). \end{aligned} \tag{53}$$

We may obtain the coefficients $a_{[g,g]}$ and $b_{[g,g]}$ from Table 1 and use them in Equations (49)-(53) to show that

$$\begin{aligned} S_2(N) &= \tfrac{1}{4} N(\log_2 N)^2 - \tfrac{1}{4} N(\log_2 N) + N + O(1); \\ S_3(N) &= .265\, N(\log_2 N)^2 - .315\, N(\log_2 N) + 1.25N + O(1); \\ S_4(N) &= \tfrac{1}{4} N(\log_2 N)^2 - \tfrac{1}{3} N(\log_2 N) + \tfrac{11}{9} N + O(1). \end{aligned} \tag{54}$$

Since the leading coefficient for $S_3(N)$ exceeds that for both $S_2(N)$ and $S_4(N)$, it is not surprising that $S_3(N)$ is generally larger than the other two. Also, although the leading coefficient is $\frac{1}{4}$ for both $S_2(N)$ and $S_4(N)$, the coefficient for the term $N(\log_2 N)$ is smaller for $S_4(N)$. This explains why $S_4(N)$ is smaller than $S_2(N)$ for sufficiently large N, that is, for $N > 8$.

From (50) and (53) we observe that the coefficient for the term $N(\log_2 N)^2$ in the expansion of $S_g(N)$ is

$$\alpha_g(\log_2 g)^{-2} \quad = \quad \tfrac{1}{2}(\log_2 g)^{-2}\ a_{[g,g]}. \tag{55}$$

When $g = 2^r$, we may use (41) to show that (55) reduces to $\frac{1}{4}$. However, for all $[g,g]$ f-networks known, $a_{[g,g]} > \frac{1}{2}(\log_2 g)^2$ if g is not a power of 2. Therefore, the leading coefficient in the expansion of $S_g(N)$ is minimized - - and its value is $\frac{1}{4}$ - - if and only if g is a power of 2.

In view of the above observations we conclude that the most economical N-sorter networks are achieved when g is a power of 2. Furthermore, we might hope to achieve successive reductions in the asymptotic growth of $S_g(N)$ by choosing $g = 2^r$ (which maintains a leading coefficient of $\frac{1}{4}$), for successively larger r. Therefore, we use (50)-(53), along with $a_{[8,8]}$, $b_{[8,8]}$, $a_{[16,16]}$, and $b_{[16,16]}$ given by (41) and (43), and with $\hat{S}(8) = 19$ and $\hat{S}(16) = 60$ obtained from Table 2, to derive

$$\begin{aligned} S_8(N) &= \frac{1}{4} N(\log_2 N)^2 - \frac{1}{3} N(\log_2 N) + \frac{9}{7} N + O(1). \\ S_{16}(N) &= \frac{1}{4} N(\log_2 N)^2 - \frac{1}{3}(\frac{71}{64}) N(\log_2 N) + \frac{59}{45} N + O(1). \end{aligned} \tag{56}$$

Comparing these results with (54) we observe that for sufficiently large N, $S_8(N) > S_4(N) > S_{16}(N)$. And, trying $g = 2^r$ for successively larger values of r, we find that successive improvements occur only when r is itself a power of 2, so that the first improvement over $S_{16}(N)$ occurs when $r = 2^3 = 8$, or $g = 2^8 = 256$.

The most economical (2^{2r})-sorter network known, when $r = 2^s \geq 4$, uses the $[2^r,2^r]$ strategy, so that $\hat{S}(2^{2r}) = S_{2^r}(2^{2r})$. (See [7].) We may use this observation to eliminate $\hat{S}(2^{2r})$ from the right-hand-side of (51), evaluated with $g = 2^{2r}$, thereby obtaining

$$\beta_{2^{2r}} = 2^{-2r} S_{2^r}(2^{2r}) - \tfrac{1}{2}a_{[2^{2r},2^{2r}]} - 2^{-2r}(2^{2r} - 1)^{-1} b_{[2^{2r},2^{2r}]}. \qquad (57)$$

Using (41), (49), (50), (52), and some algebra we can reduce (57) to the following recurrence relation for the coefficient β_{2^r}.

$$\beta_{2^{2r}} = 2\beta_{2^r} + (1 - 2^{-2r})\{ (2^r - 1) b_{[2^r,2^r]} - (2^{2r} - 1) b_{[2^{2r},2^{2r}]} \}. \qquad (58)$$

The solution to (58), with $b_{[2^k,2^k]}$ given by (43) and with the boundary condition $\beta_{16} = 71/48$ obtained from (51), is

$$\beta_{2^r} = - (\tfrac{17}{64} + \sigma_s) r, \qquad (59)$$

when $r = 2^s$ and

$$\sigma_s = \frac{1}{6} \sum_{0 \leq \ell < s} 2^{-(2^\ell+\ell)}. \qquad (60)$$

From (43) and (52) we obtain

$$\gamma_{2^r} = (2^r - 1)^{-2} \left(\frac{4}{3} 4^r - 3\, 2^r + \frac{5}{3}\right). \tag{61}$$

Since σ_s rapidly converges to .107 and since $\gamma_{2^r} \approx (4 - 2^{-r})/3$, we may achieve a growth rate for $\hat{S}(N)$ of

$$\hat{S}(N) = .250\, N(\log_2 N)^2 - .372\, N(\log_2 N) + 1.333\, N + O(1). \tag{62}$$

Equation (62) represents an improvement of order $N(\log_2 N)$ over $B(N) = S_2(N)$, which exhibits the smallest growth rate known previously. Furthermore, the minimum growth rate of (62) is nearly achieved by $S_{16}(N)$, since the coefficient of $N(\log_2 N)$ in (56) is $-.370$.

VII. Conclusion

The strategy used by most of the best previous N-sorter networks is to divide the N inputs into 2 groups, sort these groups separately, and then merge the results. The best merge networks, suggested by K. E. Batcher [4], partition each sorted multiset into 2 divisions, merge each division of the first sorted multiset with one of the second, and then use $\frac{1}{2}N-1$ comparators to resolve the remaining ambiguities.

Our results demonstrate that greater economy can be achieved in N-sorter networks by dividing the N inputs into $g > 2$ groups that are sorted separately, and by partitioning each sorted multiset into $d > 2$ divisions to be combined by d merge networks. In particular, we have shown that by using $g = d = 4$, we can achieve N-sorter networks that are more economical than Batcher's for $N > 8$ and that are more economical than any networks previously designed for $N > 34$. We have indicated that even greater savings can be achieved by using $g = d = 2^r$, where $r = 2^s > 4$; however, these constructions are only applicable for $N \geq g \cdot d = 4^r$.

Our N-sorter networks require order $N(\log_2 N)$ fewer comparators than the best previous networks. However, we have not been able to improve upon the asymptotic growth rate of $\frac{1}{4}N(\log_2 N)^2$ achieved with Batcher's construction. As noted above, the coefficient for the term $N(\log_2 N)^2$ with our construction is given by

$$\alpha_g(\log_2 g)^{-2} = \tfrac{1}{2}(\log_2 g)^{-2}\, a_{[g,g]}. \qquad (63)$$

Since $a_{[g,g]}$ is $(1/g)$ times the number of templates required by the

$[g,g]$ f-network, we could reduce the coefficient of the term $N(\log_2 N)^2$ by constructing an improved $[g,g]$ f-network that required fewer than $\frac{1}{2}(\log_2 g)^2$ templates. However, in Appendix B we show that

$$f_{[g,d]}(N) \geq (1 - d^{-1}) N - (d - 1), \qquad (64)$$

so that $a_{[g,g]} > 0$ and $\alpha_g > 0$. Therefore, the $[g,d]$ strategy must require order $N(\log_2 N)^2$ comparators.

Appendix A: Proof of Theorem 3

The proof of Theorem 3 is facilitated by the following two lemmas.

Lemma 1:

If the rows $V_{(i,*)}$ of an $r \times d$ array V are ordered, then the columns $V_{(*,j)}$ are also ordered if and only if

$$n_{(r,*)} \leq n_{(r-1,*)} \leq \cdots \leq n_{(1,*)} . \tag{65}$$

Proof: Lemma 1 follows from Theorem 1 by symmetry.

Lemma 2:

Suppose that the $t \times s$ planes $V_{(*,*,k)}$ of the $t \times s \times d$ array V are ordered. Then if we sort the t d planes $V_{(*,j,*)}$, the $t \times s$ planes remain ordered.

Proof:

Assume that $V_{(*,*,k)}$ is ordered, $1 \leq k \leq d$. Then the columns $V_{(i,*,k)}$, $1 \leq i \leq t$, and the verticals $V_{(*,j,k)}$, $1 \leq j \leq s$, are also ordered, since they are all submultisets of $V_{(*,*,k)}$. Therefore, by Theorem 1,

$$n_{(*,s,k)} \leq n_{(*,s-1,k)} \leq \cdots \leq n_{(*,1,k)} . \tag{66}$$

Summing (66) for $1 \leq k \leq d$ we find that

$$n_{(*,s,*)} \leq n_{(*,s-1,*)} \leq \cdots \leq n_{(*,1,*)} . \tag{67}$$

Now suppose we sort the $t \times d$ planes $V_{(*,j,*)}$, $1 \leq j \leq s$. We need to show that this operation leaves the planes $V_{(*,*,k)}$ ordered. Clearly sorting $V_{(*,j,*)}$ does not alter the number of 0's in $V_{(*,j,*)}$; therefore, $n_{(*,j,*)}$ satisfies (67) after the $t \times d$ planes are sorted. Also, once $V_{(*,j,*)}$ is sorted, the $n_{(*,j,*)}$ 0's are divided among the rows $V_{(i,j,*)}$ according to

$$n_{(i,j,*)} = \begin{cases} 0 & \text{if } n_{(*,j,*)} \in [0,(i-1)d]; \\ n_{(*,j,*)} - (i-1)d & \text{if } n_{(*,j,*)} \in [(i-1)d,id]; \\ d & \text{if } n_{(*,j,*)} \in [id,td]; \end{cases} \qquad (68)$$

Equations (67) and (68) together imply that

$$n_{(t,s,*)} \leq n_{(t,s-1,*)} \leq \cdots \leq n_{(t,1,*)} \leq n_{(t-1,s,*)} \leq n_{(t-1,s-1,*)} \leq \cdots \leq n_{(1,1,*)}. \qquad (69)$$

We can consider V to be a $ts \times d$ array with ts "rows" $V_{(i,j,*)}$, $1 \leq i \leq t$, $1 \leq j \leq s$, and with d "columns" $V_{(*,*,k)}$, $1 \leq k \leq d$. Sorting $V_{(*,j,*)}$ orders the "rows" $V_{(i,j,*)}$, because they are all sub-multisets of $V_{(*,j,*)}$. If $V_{(*,*,k)}$ is initially ordered, then sorting $V_{(*,j,*)}$ leads to (69). Now by Lemma 1, if the "rows" $V_{(i,j,*)}$ are ordered and $n_{(i,j,*)}$ satisfies (69), then the "columns" $V_{(*,*,k)}$ are also ordered. Therefore, if $V_{(*,*,k)}$ is initially ordered, then sorting $V_{(*,j,*)}$ leaves $V_{(*,*,k)}$ ordered.

Q.E.D.

We are now ready to prove Theorem 3.

Theorem 3:

Let the multiset $V = \{v_1, v_2, \ldots, v_N\}$, where $N = tsd$, be considered a $t \times s \times d$ array. Then the following small f-networks together constitute an $[sg,d]$ f-network for V.

i) s $[g,d]$ f-networks for $V_{(*,j,*)}$, $1 \leq j \leq s$; followed by

ii) one $[s,d]$ f-network for V.

Proof:

According to Definition 1, the sequence of comparators represented by i) and ii) is an $[sg,d]$ f-network for V if and only if it will complete the ordering of V given that: a) the d planes $V_{(*,*,k)}$, $1 \leq k \leq d$, are ordered; and b) $n_{(*,*,k)}$ satisfies

$$n_{(*,*,d)} \leq n_{(*,*,d-1)} \leq \cdots \leq n_{(*,*,1)} \leq n_{(*,*,d)} + sg. \tag{70}$$

Let us assume that the partial ordering in V satisfies conditions a) and b). Then, since $V_{(*,*,k)}$ is ordered, each of the submultisets $V_{(*,j,k)}$, $1 \leq j \leq s$, is also ordered. In addition, the distribution of the $n_{(*,*,k)}$ 0's among the verticals $V_{(*,j,k)}$ satisfies

$$n_{(*,j,k)} = \left\lfloor (n_{(*,*,k)} + s - j) / s \right\rfloor . \tag{71}$$

We may use (70) and (71) to show that

$$n_{(*,j,d)} \leq n_{(*,j,d-1)} \leq \cdots \leq n_{(*,j,1)} \leq n_{(*,j,d)} + g. \tag{72}$$

Since the d submultisets $V_{(*,j,k)}$, $1 \leq k \leq d$, of $V_{(*,j,*)}$ are ordered, and since $n_{(*,j,k)}$ satisfies (72), a [g,d] f-network will order $V_{(*,j,*)}$.

Lemma 2 implies that the [g,d] f-networks that order the $t \times d$ planes $V_{(*,j,*)}$ leave the $t \times s$ planes $V_{(*,*,k)}$ ordered. Furthermore, once $V_{(*,j,*)}$ is ordered, the distribution of the $n_{(*,j,*)}$ 0's among the verticals $V_{(*,j,k)}$ satisfies

$$n_{(*,j,k)} = \left\lfloor (n_{(*,j,*)} + d - k) / d \right\rfloor . \tag{73}$$

Equation (73) implies that

$$n_{(*,j,d)} \leq n_{(*,j,d-1)} \leq \cdots \leq n_{(*,j,1)} \leq n_{(*,j,d)} + 1. \tag{74}$$

Summing (74) for $1 \leq j \leq s$ we obtain

$$n_{(*,*,d)} \leq n_{(*,*,d-1)} \leq \cdots \leq n_{(*,*,1)} \leq n_{(*,*,d)} + s. \tag{75}$$

Therefore, since the [g,d] f-networks for $V_{(*,j,*)}$ leave the planes $V_{(*,*,k)}$ ordered, (75) guarantees that an [s,d] f-network will complete the ordering of V.

We have seen that if the partial ordering in V satisfies conditions a) and b), then the s [g,d] f-networks in i) followed by the [s,d] f-network in ii) will complete the ordering of V. Therefore, i) and ii) together constitute an [sg,d] f-network.

Q.E.D.

Appendix B: A Lower Bound for $f_{[g,d]}(N)$

In this appendix we calculate a lower bound for $f_{[g,d]}(N)$, the number of comparators required by the most efficient [g,d] f-network for the set $V = \{v_1, v_2, \ldots, v_N\}$, where $N = td$, $t \geq g \geq 2$. Let r and s be any two integers satisfying $1 \leq r \leq t-1$, $2 \leq s \leq d$. Then, by definition, a [g,d] f-network will complete the ordering of V if the columns $V_{(*,j)}$, $1 \leq j \leq d$, are ordered and if

$$\begin{aligned} n_{(*,1)} &= r+1; \\ n_{(*,j)} &= r, \qquad 2 \leq j \leq s-1; \\ n_{(*,j)} &= r-1, \qquad s \leq j \leq d. \end{aligned} \tag{76}$$

From (76) we see that V is ordered except that the 0 at position $V_{(r+1,1)}$ should be moved to $V_{(r,s)}$. Since a [g,d] f-network will complete the ordering of V, it must include a comparator or a sequence of comparators that provide a path from $V_{(r+1,1)}$ to $V_{(r,s)}$.

Now a comparator can only move a 0 in one position of V to a position labeled by a smaller index. Therefore, a [g,d] f-network must contain either the comparator $V_{(r,s)}:V_{(r+1,1)}$ or else the comparator $V_{(r,s)}:V_{(r,j)}$, where $s < j \leq d$ and where the f-network includes a path from $V_{(r+1,1)}$ to $V_{(r,j)}$. Since r and s are arbitrary integers satisfying $1 \leq r \leq t-1$, $2 \leq s \leq d$, we have shown that

$$f_{[g,d]}(td) \geq (t-1)(d-1), \qquad g,d \geq 2, \tag{77}$$

or, using $N = td$,

$$f_{[g,d]}(N) \geq (1 - d^{-1})\, N - (d - 1), \qquad g,d \leq 2. \qquad (78)$$

For $g > 2$ and/or $d > 2$ (78) does not provide a very tight bound for $f_{[g,d]}(N)$; indeed, it is not at all the greatest lower bound known. However, (78) is sufficient to show that $a_{[g,g]} > 0$ and $\alpha_g > 0$, so that the number of comparators required by a $[g,d]$ N-sorter network grows as $N(\log_2 N)^2$.

REFERENCES

[1] D.E. Knuth [1969]: Seminumerical algorithms. The Art of Computer Programming, **2**, Addison-Wesley Publishing Company.

[2] R.W. Floyd and D.E. Knuth [1970]: The Bose-Nelson sorting problem. CS Report 70-177, Stanford University, Stanford, California; November 1970.

[3] D.C. Van Voorhis [1970]: An improved lower bound for the Bose-Nelson sorting problem. Technical Note No. 7, Digital Systems Laboratory, Stanford University, Stanford, California; February 1971. (Also Chapter 4 of this thesis.)

[4] K.E. Batcher [1968]: Sorting networks and their applications. Proc. AFIPS Spring Joint Comp. Conf. 32, 307-314.

[5] W.A. Kautz, M.C. Pease, and M.W. Green [1970]: Cellular logic-in-memory arrays. Final Report, Part 1, SRI Project 5509, Stanford Research Institute, Menlo Park, California; May 1970.

[6] R.C. Bose and R.J. Nelson [1962]: A sorting problem. J. Assoc. Comp. Mach. **9**, 282-296.

[7] D.C. Van Voorhis [1971]: Large [g,d] sorting networks. Technical Report no. 18, Digital Systems Laboratory, Stanford University, Stanford, California; August 1971. (Also Chapter 3 of this thesis.)