A Partitioning Approach to the Design of Selection Networks

BENJAMIN W. WAH, MEMBER, IEEE, AND KUO-LIANG CHEN

Abstract **—The (m, n) selection problem is defined as the selection of the m smallest numbers in any order from a set of n numbers ($m \leq n$). In this paper, we have proposed a class of design procedures for selection networks based on partitioning. Conditions are defined so that the optimal design can be found in polynomial time. The resulting selection network has $O(\lceil \log_2 n \rceil \cdot \lceil \log_2 m \rceil)$ time complexity and $O(n \cdot \lceil \log_2^2 m \rceil)$ hardware complexity for all values of m. As a comparison, networks previously known can optimize either hardware or delay but not both simultaneously, and perform worse than pure sorting when m approaches $n/2$.**

Index Terms **—Bitonic merging network, comparison-exchange module, odd–even sorting network, partitioning, scheduling, selection.**

I. INTRODUCTION

WITH the advent of parallel processing, the scheduling of tasks is very important. Suppose there are m processors, and n tasks, each of which is characterized by a value (e.g., priority, initiation time), are to be scheduled. The problem is to select the m tasks with the minimum or maximum values. A special example is seen in the evaluation of parallel branch-and-bound algorithms under best-first search [7]. In this case, there is a large list of subproblems to be expanded, and each subproblem is characterized by a lower-bound value. The scheduler selects m subproblems with the minimum lower bounds to be expanded.

In a centralized approach, selection can be performed in time complexity $O(n)$ for n tasks [3]. However, the major bottleneck lies in the centralized collection and distribution of tasks, or in the routing of tasks to processors. This motivates us to study an interconnection network that can perform selection and task routing in parallel. Our study is based on the parallel selection of numbers, and is applicable to more complex entities.

The *(m, n) selection problem* entails the transformation of a set of n numbers into another set in which none of the first m numbers exceeds any of the last $n-m$ numbers. The algorithm for performing this is sometimes called an (m, n) algorithm [4].

Manuscript received August 16, 1982; revised August 10, 1983. This work was supported by National Science Foundation Grants ECS 81-05968 and ECS 80-16580.

B. W. Wah is with the School of Electrical Engineering, Purdue University, West Lafayette, IN 47907.

K.-L. Chen is with the Department of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui, China. He was on leave at the School of Electrical Engineering, Purdue University, West Lafayette, IN 47907 at the time of this work.

This problem is closely related to sorting except that the m numbers selected do not have to appear in any order. Parallel sorting has been investigated extensively. Batcher pioneered parallel sorting by designing a network model using 2-input, 2-output comparison-exchange modules that can compare two numbers on their inputs and switch the smaller (larger) number to the upper (lower) outputs [2]. The hardware and time complexities for sorting n numbers are $O(n \log_2^2 n)$ and $O(\log_2^2 n)$, respectively. Preparata proposed the fastest parallel sorting algorithm on an adaptive model that required $(c'/\alpha) \log_2 n + o(\log_2 n)$ time complexity and $n^{1+\alpha}$ processors for $0 < \alpha \leq 1$ assuming that memory conflicts are not allowed [5]. When memory conflicts are allowed, Preparata showed that sorting can be achieved in time $C \log_2 n + o(\log_2 n)$ with $n \log_2 n$ processors. Preparata's results are based on the use of general purpose processors and form the lower bound for parallel sorting. Other parallel sorting algorithms have also been developed. However, the set obtained by sorting is over-constrained because it is arranged in a sorted order. By relaxing this constraint, we expect the complexities of selection to be smaller than that of sorting.

Parallel networks for selection have been little studied. Alekseyev derived the lower $((n - m)\lceil \log_2(m + 1)\rceil)$ and upper $((r - 1)(2\hat{S}(m) + m))$ bounds on the number of comparators for selecting m numbers out of $n = r \cdot m$ numbers where $\hat{S}(m)$ is the minimum number of comparators needed in a sorting network for m numbers [1]. An effective procedure is based on the selection of m numbers out of $2m$ numbers. By first sorting $\langle x_1, \cdots, x_m \rangle$ and $\langle x_{m+1}, \cdots, x_{2m} \rangle$, then comparing and interchanging $x_1 : x_{2m}, x_2 : x_{2m-1}, \cdots, x_m : x_{m+1}$, half of the numbers are eliminated. To select m numbers out of $n = r \cdot m$ numbers, the procedure is repeated $r - 1$ times (eliminating m numbers each time) [4]. For this algorithm, the time complexity is worse than that of parallel sorting when m approaches $n/2$.

Yao has studied the bounds on selection networks [8]. He found that for $m < \sqrt{n}$, the upper and lower bounds on the number of comparators is $O(n\lceil \log_2(m + 1)\rceil)$. The upper bound on delay is $O(\log_2 n + \lfloor \log_2 m \rfloor \log_2 \log_2 n)$ for fixed $m \geq 2$ and large n. He developed network constructions to prove these bounds. However, there are two main drawbacks. First, the networks proposed can optimize either hardware or delay, but not both simultaneously. The reduction of one parameter results in a performance that is worse than pure sorting for the other parameter. Second, the bounds apply when m is small as compared to n ($m < \sqrt{n}$). When m approaches $n/2$, the proposed networks have worse performance than Batcher's sorting networks.

Recognizing the lack of a practical selection network, our objective is to develop a parallel selection algorithm that has *better hardware and delay complexities than parallel sorting for all values of $m \le n$*. In general, this can be implemented by either Batcher's network model or Preparata's adaptive model. In this paper, we present a *partitioning approach* to the design of selection networks based on Batcher's model. As a result, the complexities are measured in terms of the number of comparison-exchange modules and the corresponding delay. Our algorithm chooses the optimal number of partitions and the optimal size of each partition in *polynomial time*. It must be pointed out that our solution is optimal with respect to the partitioning approach and does not necessarily result in the optimal selection network (which is still an open problem).

The symbols used in describing the complexities of merging and sorting are

C the number of comparison-exchange modules, and

D the delay in units in which the delay through one module is one;

with subscripts

OE odd–even merge approach, and

BIT bitonic merge approach;

and superscripts

M merging, and

T sorting.

For example, a symbol $C^T_{\mathrm{OE}}(n)$ represents the hardware complexity of sorting n numbers using odd–even merge. A symbol $D^M_{\mathrm{OE}}(m,n)$ represents the delay of O–E merging two lists of sizes m and n. Some related results on merging and sorting are shown in Table I.

The proposed selection algorithm is based on odd–even sorting and bitonic merging [2]. The selection algorithm is discussed in Section II. Bounds on the optimal number of partitions with minimum hardware and delay are derived in Sections III and IV. The performance of our algorithm is compared to Alekseyev's and Yao's selection algorithms in Section V.

II. Design of Selection Networks Using Partitioning

The selection-network design procedure using partitioning is denoted as $SEL(m,n)\,(m \le \lfloor n/2 \rfloor)$. If $m > \lfloor n/2 \rfloor$, the procedure becomes selecting the $n-m$ greatest elements $(SEL(n-m,n))$.

The procedure consists of partitioning the numbers into multiple subsets, O–E sorting the subsets individually, and bitonically merging the subsets.

Selection Network Design Procedure $SEL(m,n)$ $m \le \lfloor n/2 \rfloor$

```
[   1) Choose $P$, the set of possible number of partitions
       (Section IV);
    2) FOR $p \in P$ DO
    3)     [ Partition the set of numbers $S$ $(|S| = n)$ into $p$
             subsets, $S_1, S_2, \cdots S_p$, such that $|S_i| \ge m$, $1 \le i \le p$ (Section III);
    4)       Sort $S_i$, $1 \le i \le p$, using O–E sort;
             Select the first $m$ elements from each sorted
             list (it can be guaranteed that the $m$ elements
             to be selected must be included in the $m$
             smallest elements of each subset);
             $p' = p$;
    5)       WHILE $p' > 1$ DO
    6)           [ Pair and juxtapose (one ascending
                   and one descending) the $p'$ sorted
                   lists into $\lfloor p'/2 \rfloor$ bitonic sequences
                   (Section IV-A)
    7)             Perform pairwise comparison on
                   each bitonic sequence to form
                   two different bitonic sequences,
                   MIN$_i$ and MAX$_i$, $1 \le i \le \lfloor p'/2 \rfloor$;
                   Eliminate MAX$_i$, $1 \le i \le \lfloor p'/2 \rfloor$
                   (Property on bitonic sequences—
                   Table I);
    8)             IF $\lceil p'/2 \rceil > 1$ THEN
                       Sort MIN$_i$, $1 \le i \le \lfloor p'/2 \rfloor$,
                       using bitonic merging;
    9)             $p' = \lceil p'/2 \rceil$
                 ]
           ]
   10) Choose $p \in P$ that result in the minimum number
       of comparators
]
```

The hardware for performing pairwise comparison and bitonic merging (steps 7 and 8) constitute a *merging module*.

TABLE I
Some Related Results in Merging and Sorting Networks

$C^M_{\mathrm{OE}}(n_1,n_2) = \begin{cases} n_1 n_2 \\ C^M_{\mathrm{OE}}\left(\left\lceil \frac{n_1}{2} \right\rceil, \left\lceil \frac{n_2}{2} \right\rceil\right) + C^M_{\mathrm{OE}}\left(\left\lfloor \frac{n_1}{2} \right\rfloor, \left\lfloor \frac{n_2}{2} \right\rfloor\right) + \left\lfloor (n_1+n_2-1)/2 \right\rfloor \end{cases}$	if $n_1 n_2 \le 1$; if $n_1 n_2 > 1$	(1)
$D^M_{\mathrm{OE}}(n_1,n_2) = 1 + \left\lceil \log_2 \max(n_1,n_2) \right\rceil$	for $n_1 n_2 \ge 1$	(2)
$C^M_{\mathrm{BIT}}(n) = C^M_{\mathrm{BIT}}\left(\left\lceil \frac{n}{2} \right\rceil\right) + C^M_{\mathrm{BIT}}\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \left\lfloor \frac{n}{2} \right\rfloor$	$n \ge 2$	(3)
$C^M_{\mathrm{BIT}}(2^t) = t\,2^{t-1}$	$n = 2^t \ge 1$	(4)
$D^M_{\mathrm{BIT}}(n) = \left\lceil \log_2 n \right\rceil$	$n \ge 2$	(5)
$C^T_{\mathrm{OE}}(n) = C^T_{\mathrm{OE}}\left(\left\lceil \frac{n}{2} \right\rceil\right) + C^T_{\mathrm{OE}}\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + C^M_{\mathrm{OE}}\left(\left\lceil \frac{n}{2} \right\rceil, \left\lfloor \frac{n}{2} \right\rfloor\right)$	$n \ge 2$	(6)
$C^T_{\mathrm{OE}}(2^t) = 2^{t-2}(t^2 - t + 4) - 1$	$n = 2^t \ge 1$	(7)
$D^T_{\mathrm{OE}}(n) = \binom{\lceil \log_2 n \rceil + 1}{2} = \left(\left\lceil \log_2 n \right\rceil\right)\left(\left\lceil \log_2 n \right\rceil + 1\right)/2$	$n \ge 2$	(8)
$C^T_{\mathrm{BIT}}(n) = C^T_{\mathrm{BIT}}\left(\left\lceil \frac{n}{2} \right\rceil\right) + C^T_{\mathrm{BIT}}\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + C^M_{\mathrm{BIT}}(n)$	$n \ge 2$	(9)
$D^T_{\mathrm{BIT}}(n) = \left(\left\lceil \log_2 n \right\rceil\right)\left(\left\lceil \log_2 n \right\rceil + 1\right)/2$		(10)
$C^T_{\mathrm{BIT}}(n) \ge C^T_{\mathrm{OE}}(n) \ge C^M_{\mathrm{BIT}}(n) \ge C^M_{\mathrm{OE}}\left(\left\lceil \frac{n}{2} \right\rceil, \left\lfloor \frac{n}{2} \right\rfloor\right) \ge C^M_{\mathrm{BIT}}\left(\left\lceil \frac{n}{2} \right\rceil\right) + \left\lfloor \frac{n}{2} \right\rfloor$	$n \ge 2$	(11)

Property on bitonic sequence:
For a bitonic sequence of $2n$ numbers, $a_1, a_2, \ldots, a_{2n}$, compute for $i = 1, 2, \ldots, n$: $b_i = \min(a_i, a_{i+n})$; $c_i = \max(a_i, a_i + n)$. The two sequences: $\mathrm{MIN} = b_1, b_2, \ldots, b_n$ and $\mathrm{MAX} = c_1, c_2, \ldots, c_n$ are both bitonic, and $b_i \le c_j$ for all $1 \le i, j \le n$.

The above procedure clearly selects the m smallest elements in an unsorted order.

It should be mentioned that our proposed design procedure is similar to Alekseyev's method [1] with a major difference in step (8) where Alekseyev proposed to sort each list using O–E sort. But since the list is already bitonic, it could be sorted more efficiently using pairwise comparison and bitonic merging (11). Further, m is restricted in Alekseyev's method to be a factor of n, and the set S is partitioned into n/m subsets of m elements each. We have lifted this restriction in our procedure. We will analyze in the next two sections the optimal number of partitions and the number of elements in each partition to result in the minimum hardware or delay.

III. Optimal Sizes of Partitions For a Given Number of Partitions

In this section, we present efficient procedures to obtain *p-optimality,* the partitioning of n numbers into p partitions with at least m elements in each partition so that the total number of comparators for O–E sorting each partition is minimum [step 3 of Procedure $SEL(m, n)$]. Comparators for the merging modules will not be accounted for until Section IV. The range on the size of each partition is determined and this substantially reduces the number of enumerations. In Section III-A, results for two partitions are shown. This is extended to the general case in Section III-B.

A. 2-Partition Case

Consider first the case which sets no requirement on the minimum number of elements in each partition. Without loss of generality, m is set to 1. The following procedure establishes 2-optimal partitioning for $m = 1$ and $n \geq 2$.

Optimal Partitioning Procedure PART(1, n, 2), $n \geq 2$

[1) Compute the smallest integral power of 2 that is larger than $n/2$ (n_U), the largest integral power of 2 that is smaller than $n/2$ (n_L) and the midpoint (n_M):

$$n_L = 2^{\lfloor \log_2 \lfloor n/2 \rfloor \rfloor} = 2^{\lfloor \log_2 n/2 \rfloor};$$

$$n_U = 2^{\lceil \log_2 \lceil (n+1)/2 \rceil \rceil} = 2^{\lceil \log_2 (n+1)/2 \rceil}; \qquad n_M = \frac{n_L + n_U}{2};$$

2) Let (n_1', n_2') be the desired partitions, $n_1' \leq n_2', n_1' + n_2' = n$. Set one of the partitions to have a size of n_L, n_M or n_U, and the size of the other partition is adjusted accordingly.

$$\text{If } n_L \leq \left\lfloor \frac{n}{2} \right\rfloor < \frac{n_L + n_M}{2},$$

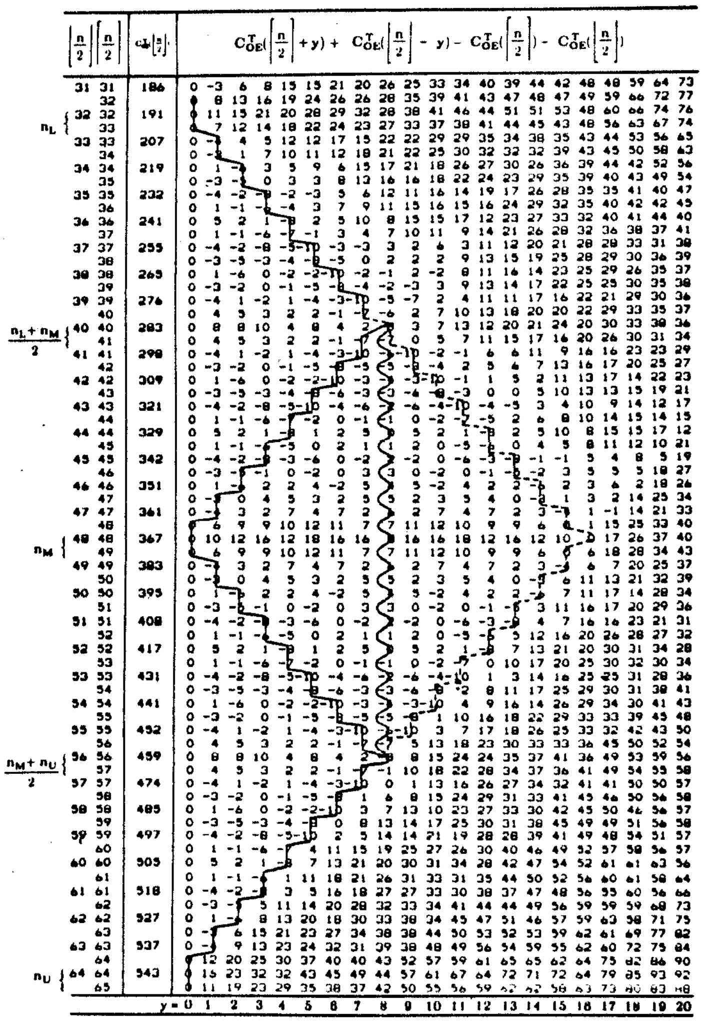
$$\text{then set} \quad n_1' = n_L, n_2' = n - n_L;$$

$$\text{If } \frac{n_L + n_M}{2} \leq \left\lfloor \frac{n}{2} \right\rfloor < \frac{n_M + n_U}{2},$$

$$\text{then set} \quad \begin{array}{l} n_1' = \min(n - n_M, n_M) \\ n_2' = \max(n - n_M, n_M) \end{array},$$

$$\text{If } \frac{n_M + n_U}{2} \leq \left\lfloor \frac{n}{2} \right\rfloor < n_U,$$

$$\text{then set} \quad n_1' = n - n_U, n_2' = n_U$$



	$\left\lfloor \frac{n}{2} \right\rfloor$	$\left\lceil \frac{n}{2} \right\rceil$	$c^T_{\cdot}\left\lfloor \frac{n}{2} \right\rfloor$	$C^T_{OE}(\left\lceil \frac{n}{2} \right\rceil + y) + C^T_{OE}(\left\lfloor \frac{n}{2} \right\rfloor - y) - C^T_{OE}(\left\lceil \frac{n}{2} \right\rceil) - C^T_{OE}(\left\lfloor \frac{n}{2} \right\rfloor)$																				
	31	31	186	0	-3	6	8	15	15	21	20	26	25	33	34	40	39	44	42	48	48	59	64	73
		32		0	8	13	16	19	24	26	26	28	35	39	41	43	47	48	47	49	59	66	72	77
n_L	32	32	191	0	11	15	21	20	28	29	32	28	38	41	46	44	51	51	53	48	60	66	74	76
		33		0	7	12	14	18	22	24	23	27	33	37	38	41	44	45	43	48	56	63	67	74
	33	33	207	0	-3	4	5	12	12	17	15	22	22	29	29	35	34	38	35	43	44	53	56	65
		34		0	-2	1	7	10	11	12	18	21	22	25	30	32	32	32	39	43	45	50	58	63
	34	34	219	0	1	[illegible]	3	5	9	6	15	17	21	18	26	27	30	26	36	39	44	42	52	56
		35		0	-3	[illegible]	0	3	3	8	13	16	16	18	22	24	23	29	35	39	40	43	49	54
	35	35	232	0	-4	-2	-8	-2	-3	5	6	12	11	16	14	19	17	26	28	35	35	41	40	47
		36		0	1	-1	-6	-4	3	7	9	11	15	16	15	16	24	29	32	35	40	42	42	45
	36	36	241	0	5	2	1	-8	2	5	10	8	15	15	17	12	23	27	33	32	40	41	44	40
		37		0	1	-1	-6	-7	-1	3	4	7	10	11	9	14	21	26	28	32	36	38	37	41
	37	37	255	0	-4	-2	-8	-5	-10	-3	-3	3	2	6	3	11	12	20	21	28	28	33	31	38
		38		0	-3	-5	-3	-4	-8	-5	0	2	2	2	9	13	15	19	25	28	29	30	36	39
	38	38	265	0	1	-6	0	-2	-2	-10	-2	-1	2	-2	8	11	16	14	23	25	29	26	35	37
		39		0	-3	-2	0	-1	-5	-8	-4	-2	-3	3	9	13	14	17	22	25	25	30	35	38
	39	39	276	0	-4	1	-2	1	-4	-3	-10	-5	-7	2	4	11	11	17	16	22	21	29	30	36
		40		0	4	5	3	2	2	-1	-7	-6	2	7	10	13	18	20	20	22	29	33	35	37
$\frac{n_L + n_M}{2}$	40	40	283	0	8	8	10	4	8	4	2	-8	3	7	13	12	20	21	24	20	30	33	38	36
		41		0	4	5	3	2	2	-1	-7	[illegible]	0	5	7	11	15	17	16	20	26	30	31	34
	41	41	298	0	-4	1	-2	1	-4	-3	-10	[illegible]	-10	-2	-1	6	6	11	9	16	16	23	23	29
		42		0	-3	-2	0	-1	-5	-8	-5	-5	-8	-4	2	5	6	7	13	16	17	20	25	27
	42	42	309	0	1	-6	0	-2	-2	-10	-3	[illegible]	-3	-10	-1	1	5	2	11	13	17	14	22	23
		43		0	-3	-5	-3	-4	-8	-6	-3	-3	-6	-8	-3	0	0	5	10	13	13	15	19	21
	43	43	321	0	-4	-2	-8	-5	-10	-4	-6	[illegible]	-6	-4	-10	-4	-5	3	4	10	9	14	12	17
		44		0	1	-1	-6	-7	-2	0	-1	-1	0	-2	-7	-5	2	6	8	10	14	15	14	15
	44	44	329	0	5	2	1	-8	1	2	5	[illegible]	5	2	1	-8	2	5	10	8	15	15	17	12
		45		0	1	-1	-6	-5	0	2	1	1	2	0	-5	-6	0	4	5	8	11	12	10	21
	45	45	342	0	-4	-2	-8	-3	-6	0	-2	[illegible]	-2	0	-6	-3	-8	-1	-1	5	4	8	5	19
		46		0	-3	-5	-1	0	-2	0	3	3	0	-2	0	-1	-6	-2	3	5	5	5	18	27
	46	46	351	0	1	[illegible]	2	2	4	-2	5	[illegible]	5	-2	4	2	2	-6	2	3	6	2	18	26
		47		0	[illegible]	0	4	5	3	2	5	5	2	3	5	4	0	-2	1	3	2	14	25	34
	47	47	361	0	[illegible]	3	2	7	4	7	2	[illegible]	2	7	4	7	2	3	[illegible]	1	-1	14	21	33
		48		0	6	9	9	10	12	11	7	7	11	12	10	9	9	6	[illegible]	1	15	25	33	40
n_M	48	48	367	0	10	12	16	12	18	16	16	[illegible]	16	16	18	12	16	12	10	[illegible]	17	26	37	40
		49		0	6	9	9	10	12	11	7	7	11	12	10	9	9	6	[illegible]	6	18	28	34	43
	49	49	383	0	[illegible]	3	2	7	4	7	2	[illegible]	2	7	4	7	2	3	[illegible]	6	7	20	25	37
		50		0	[illegible]	0	4	5	3	2	5	5	2	3	5	4	0	-5	6	11	13	21	32	39
	50	50	395	0	1	-6	2	2	4	-2	5	[illegible]	5	-2	4	2	2	-6	7	11	17	14	28	34
		51		0	-3	-6	-1	0	-2	0	3	3	0	-2	0	-1	-6	3	11	16	17	20	29	36
	51	51	408	0	-4	-2	-8	-3	-6	0	-2	[illegible]	-2	0	-6	-3	-8	4	7	16	16	23	21	31
		52		0	1	-1	[illegible]	-5	0	2	1	1	2	0	-5	-6	5	12	16	20	26	28	27	32
	52	52	417	0	5	2	1	-8	1	2	5	[illegible]	5	2	1	-8	7	13	21	20	30	31	34	28
		53		0	1	-1	-6	-7	-2	0	-1	-1	0	-2	-7	0	10	17	20	25	30	32	30	34
	53	53	431	0	-4	-2	-8	-5	-10	-4	-6	-2	-6	-4	-10	1	3	14	16	25	25	31	28	36
		54		0	-3	-5	-3	-4	-8	-6	-3	-3	-6	-8	2	8	11	17	25	29	30	31	38	41
	54	54	441	0	1	-6	0	-2	-2	-10	-3	[illegible]	-3	-10	4	9	16	14	26	29	34	30	41	43
		55		0	-3	-2	0	-1	-5	-8	-5	-5	-8	1	10	16	18	22	29	33	33	39	45	48
	55	55	452	0	-4	1	-2	1	-4	-3	-10	[illegible]	-10	3	7	17	18	26	25	33	32	42	43	50
		56		0	4	5	3	2	2	-1	-7	-7	5	13	18	23	30	33	33	36	45	50	52	54
$\frac{n_M + n_U}{2}$	56	56	459	0	8	8	10	4	8	4	2	-8	8	15	24	24	35	37	41	36	49	53	59	56
		57		0	4	5	3	2	2	-1	-7	-1	10	18	22	28	34	37	36	41	49	54	55	58
	57	57	474	0	-4	1	-2	1	-4	-3	-10	0	1	13	16	26	27	34	32	41	41	50	50	57
		58		0	-3	-2	0	-1	-5	-8	1	6	8	15	24	29	31	33	41	45	46	50	56	58
	58	58	485	0	1	-6	0	-2	-2	-10	3	7	13	10	23	27	33	30	42	45	50	46	56	57
		59		0	-3	-5	-3	-4	-8	0	8	13	14	17	25	30	31	38	45	49	49	51	56	58
	59	59	497	0	-4	-2	-8	-5	-10	2	5	14	14	21	19	28	28	39	41	49	48	54	51	57
		60		0	1	-1	-6	-7	4	11	15	19	25	27	26	30	40	46	49	52	57	58	56	57
	60	60	505	0	5	2	1	-8	7	13	21	20	30	31	34	28	42	47	54	52	61	61	63	56
		61		0	1	-1	-6	1	11	18	21	26	31	33	31	35	44	50	52	56	60	61	58	64
	61	61	518	0	-4	-2	-8	3	5	16	18	27	27	33	30	38	37	47	48	56	55	60	56	66
		62		0	-3	-5	5	11	14	20	28	32	33	34	41	44	44	49	56	59	59	59	68	73
	62	62	527	0	1	-6	8	13	20	18	30	33	38	34	45	47	51	46	57	59	63	58	71	75
		63		0	-3	6	15	21	23	27	34	38	38	44	50	53	52	53	59	62	61	69	77	82
	63	63	537	0	[illegible]	9	13	23	24	32	31	39	38	48	49	56	54	59	55	62	60	72	75	84
		64		0	12	20	25	30	37	40	40	43	52	57	59	61	65	65	62	64	75	82	86	90
n_U	64	64	543	0	15	23	32	32	43	45	49	44	57	61	67	64	72	71	72	64	79	85	93	92
		65		0	11	19	23	29	35	38	37	42	50	55	56	59	62	62	58	63	73	80	83	88
			y =	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Fig. 1. The number of comparators required for O–E sorting using 2 partitions normalized with respect to 2 partitions of sizes $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil$, $64 \leq n \leq 127$ (solid and dotted lines indicate combinations with the minimum number and comparators).

The procedure has $O(1)$ time complexity. It is illustrated in Fig. 1 with $n_L = 32$, $n_U = 64$, and $n_M = 48$ for $64 \leq n < 128$. An entry in the table represents the number of comparators required for O–E sorting two partitions of sizes $\lfloor n/2 \rfloor - y$ and $\lceil n/2 \rceil + y$ normalized with respect to that for sorting two partitions of sizes $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil$. The minimum value(s) in each row is, thus, the normalized optimal number of comparators required. The minima are joined together using solid and dotted lines. Procedure PART$(1, n, 2)$ obtains the locus on the solid line for $n_L \leq x < (n_L + n_M)/2$ or $(n_M + n_U)/2 \leq x < n_U$, and on the dotted line for $(n_L + n_M)/2 \leq x < (n_M + n_U)/2$.

As an example, for $n = 87$, $\lfloor n/2 \rfloor = 43$, $\lceil n/2 \rceil = 44$. Scanning along the corresponding row in Fig. 1, the minimum values occur at $y = 4$ and $y = 11$ with $C^T_{OE}(43) + C^T_{OE}(44) - 7 = 643$ comparators. It corresponds to setting $n_1' = 43 - y$ and $n_2' = 44 + y$. That is, $(n_1', n_2') = (39, 48)$ or $(32, 55)$. The proposed partitioning procedure finds the partitions $(39, 48)$.

Theorem 1: Procedure $PART(1, n, 2)$, $n \geq 2$, results in 2-optimality. The proof of Theorem 1 is long and is shown in the Appendix.

Theorem 1 shows the optimality of the partitioning procedure. However, the procedure is developed with respect to $m = 1$. If this restriction is lifted, the procedure has to be modified because $PART(1, n, 2)$ may not result in feasible partitions. The modified optimal partitioning procedure, **PART(m, n, 2)**, $n \geq 2m$, partitions the set as $PART(1, n, 2)$. However, when the size of the smaller partition using $PART(1, n, 2)$, n_1', is less than m, a limited enumerative search is performed to find n_1' and $n - n_1'$ that satisfies $\min_{m \leq n_1' \leq \lfloor n/2 \rfloor}\{C_{\mathrm{OE}}^T(n_1') + C_{\mathrm{OE}}^T(n - n_1')\}$. The procedure has a time complexity of $O(1)$ when the optimal sizes of the partitions found by $PART(1, n, 2)$ are greater than or equal to m. Otherwise, enumeration is necessary, and the time complexity is $O(n/2 - m)$.

B. General Case with $p \geq 2$ Partitions

In this section, we present conditions and bounds that result in p-optimal partitioning. By using these bounds, efficient design procedures can be developed.

Theorem 2: The necessary condition for p-optimality is that the partitions are pairwise optimal.

Proof: Assume that the partitions are p-optimal, and there are two partitions that are not pairwise optimal. These two partitions can be rearranged to result in a smaller number of comparators which contradicts the assumption of p-optimality. ∎

Theorem 3: In a p-optimal partitioning with $\lfloor n/p \rfloor \geq m$, the size of each partition is between $\max(m, 2^{\lfloor \log_2(n/p) \rfloor})$ and $2^{\lceil \log_2(n/p) \rceil}$.

Proof: The condition $\lfloor n/p \rfloor \geq m$ is used to ensure the possibility of all p partitions having sizes greater than or equal to m.

For $\lfloor \log_2(n/p) \rfloor < \lceil \log_2(n/p) \rceil$, consider first that $m \leq 2^{\lfloor \log_2(n/p) \rfloor}$. We assert that in an optimal partitioning the size of any partition must lie in the range $2^{\lfloor \log_2(n/p) \rfloor}$ and $2^{\lceil \log_2(n/p) \rceil}$. Suppose this is false, then there exists two partitions S_1, S_2, with sizes s_1, s_2, such that $s_1 < 2^{\lfloor \log_2(n/p) \rfloor}$, and $s_2 > 2^{\lfloor \log_2(n/p) \rfloor}$, or $s_1 < 2^{\lceil \log_2(n/p) \rceil}$ and $s_2 > 2^{\lceil \log_2(n/p) \rceil}$. From Theorem 1, S_1 and S_2 can be rearranged to result in a smaller total number of comparators. This contradicts the assumption that the partitions are p-optimal. For the case $m > 2^{\lfloor \log_2(n/p) \rfloor}$, the lower limit is set to m. Therefore, the lower bound on the partition size is $\max(m, 2^{\lfloor \log_2(n/p) \rfloor})$.

For the special case when $\lfloor \log_2(n/p) \rfloor = \lceil \log_2(n/p) \rceil$, according to Theorem 1, the size of each partition is set to n/p to result in p-optimally. ∎

Theorems 2 and 3 lead to a procedure for finding p-optimal partitions. The procedure is a reduced form of enumeration. Assuming that there are P partitions, it is written in a recursive form and assigns one partition in each call.

Procedure OPART(e, p, min, max):

```
[ /* Procedure to find optimal allocation of $e$ elements into partitions $p$ thru $P$. The size of each of these partitions is between min and max. */
  (1) /* Check if limits are exceeded:
The partitions are assigned in increasing sizes due to symmetry. To assign $e$ elements into $P - p + 1$ partitions, the minimum partition size is $\lfloor e/(P - p + 1) \rfloor$ and must be greater than min. Likewise, the lower bound on the maximum partition size is $\lceil e/(P - p + 1) \rceil$ and must be less than max. If these limits are exceeded, the procedure returns and chooses a different assignment for partition $p - 1$ */
  (2) /* Check for condition of last partition:
If $p$ equals $P$, this is the last partition, and all the $e$ elements are assigned to this partition. If this assignment is pairwise optimal with assignments in partition 1 through $p - 1$ (Theorem 2), then the total number of comparators required for O–E sorting the partitions is computed. This number is compared to the minimum obtained previously and is taken to be the minimum if smaller. The procedure then returns to set a different assignment for partition $p - 1$. */
  (3) /* Assign elements to the current partition and call OPART again:
Choose $i$ elements ranging from min to $\lfloor e/(P - p + 1) \rfloor$ to be assigned to partition $p$. If this assignment is pairwise optimal with assignments in partitions 1 through $p - 1$, then the assignment is valid and procedure OPART is called to assign partition $p + 1$ ($OPART(e - i, p + 1, i, \max)$). */
]
```

The procedure is called initially with $OPART(n, 1, \max(m, 2^{\lfloor \log_2(n/p) \rfloor}), 2^{\lceil \log_2(n/p) \rceil})$. The complexity of the procedure is difficult to evaluate. However, it is exponential with respect to $\lfloor n/p \rfloor$.

A simulation of *OPART* shows that values of the maximum and minimum feasible solutions deviate by about 1.5 percent (Fig. 2). Thus, a heuristic procedure that generates any feasible solution is usually acceptable. Theorem 2 is useful in generating a feasible solution. The n elements are distributed uniformly into P partitions. The partitions are examined and rearranged in pairs to achieve 2-optimality. This is repeated until no further improvement is possible. The heuristic, **HPART**, is not shown here and is called with similar arguments as *OPART*. It runs very efficiently and has complexity $O(p^2)$. Performance results of the optimal and heuristic procedures will be shown in Section V.

IV. Optimal Number of Partitions

In this section, we develop bounds on the number of partitions that result in the minimum number of comparators or delay. These bounds are useful in reducing the complexity of designing an optimally partitioned selection network [step 1 of Procedure $SEL(m, n)$].

A. Bounds on the Number of Partitions with Minimum Comparators

The results in Section III are related to a fixed number of partitions and does not include the hardware for the merging modules. In order to find the optimal number of partitions, the hardware for merging must also be considered.

For a given number of partitions p the partitions are merged in pairs until a single partition is obtained. The number of merging modules needed is the same as the number of nonterminals in a binary tree with degree of 2 and p terminals. Let x be the total number of terminals and nonterminals

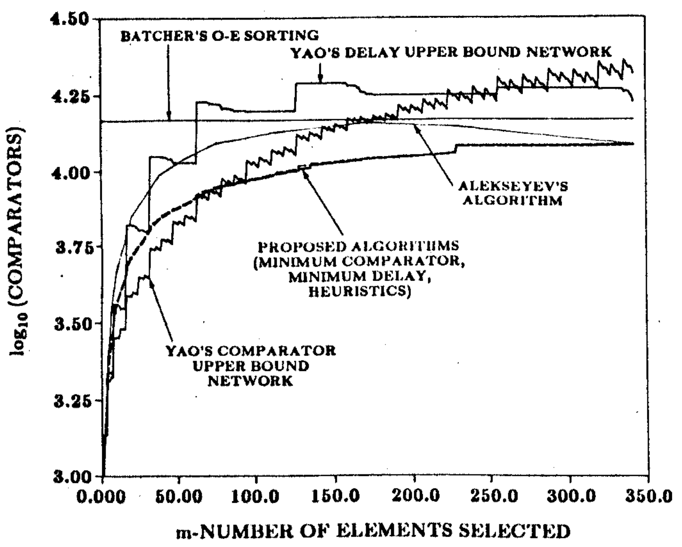


Fig. 2. Hardware complexities of different parallel selection algorithms ($n = 684$).

in the tree. There are $x - 1$ incoming edges and $(x - p)2$ outgoing edges. Solving this, we get $x = 2p - 1$. The number of nonterminals (merging modules) equals $p - 1$. The number of levels of nonterminal nodes is $\lceil \log_2 p \rceil$.

The following theorem defines the bounds on the number of partitions.

Theorem 4: The optimal number of partitions for minimizing hardware P^C_{OPT} has a lower bound $P^C_L = \lfloor n/2^{\lceil \log_2 m \rceil} \rfloor$ and an upper bound $P^C_U = \lfloor n/m \rfloor$.

Proof: The maximum number of partitions P^C_U must be $\lfloor n/m \rfloor$. Otherwise, there will be at least one partition whose size is less than m.

To prove the lower bound, suppose there are P^C_L partitions, and each partition is forced to have $2^{\lceil \log_2 m \rceil}$ elements with the exception of possibly one partition which has $2^{\lceil \log_2 m \rceil} \leq [n - 2^{\lceil \log_2 m \rceil}(P^C_L - 1)] < 2^{\lceil \log_2 m \rceil + 1}$ elements. We show that it is impossible to improve on the total number of comparators by decreasing the number of partitions.

We first have to prove that this distribution is P^C_L-optimal. The proof is shown by induction on the number of partitions. Given this fact, a partition is now eliminated, and the elements contained in it are distributed among the remaining $(P^C_L - 1)$ partitions until the allocation is $(P^C_L - 1)$-optimal. We assert that it is sufficient to combine two partitions of size $2^{\lceil \log_2 m \rceil}$ each into one partition in order to achieve $(P^C_L - 1)$-optimality. The assertion is also proved by induction on the number of partitions and using Theorem 1.

From (7), with $P^C_L \geq 3$, the increase in the number of comparators for O–E sorting due to the elimination of one partition is

$$C^T_{\mathrm{OE}}(2^{\lceil \log_2 m \rceil + 1}) - 2C^T_{\mathrm{OE}}(2^{\lceil \log_2 m \rceil}) = 2^{\lceil \log_2 m \rceil}\lceil \log_2 m \rceil + 1. \qquad (12)$$

By eliminating one partition and assuming $P^C_L \geq 3$, the comparators for one merging module is saved. The maximum saving occurs at $m = 2^{\lceil \log_2 m \rceil}$ and, from (4),

$$C^M_{BIT}(2^{\lceil \log_2 m \rceil}) + m = \lceil \log_2 m \rceil 2^{\lceil \log_2 m \rceil - 1} + m. \qquad (13)$$

Equality exists between (12) and (13) for $m \leq 2$. For $m \geq 3$, the terms in (12) are larger. This shows that eliminating one partition results in increased hardware.

The same argument follows that eliminating more than one partition is not cost-effective. The lower bound on the number of partitions is, therefore, $\lfloor n/2^{\lceil \log_2 m \rceil} \rfloor$. ∎

Although the number of merging modules can be obtained by finding the optimal number of partitions P^C_{OPT}, the overall delay of the network is affected by the way that these modules are connected [step 6 of Procedure $SEL(m, n)$]. To illustrate this, consider the selection of 127 numbers out of 428. The optimal partitions are 128, 128, and 172 with delays of 28, 28, and 36 units, respectively. The delay of a merging module for pairwise comparison of 254 elements and bitonic merging of 127 elements is 8 units. Suppose the partitions of sizes 128 and 172 are first combined through bitonic merging. This requires a delay of $8 + \max(28, 36) = 44$ units. One more unit of delay is needed due to the final pairwise comparison of 254 elements to result in a total delay of 45 units. Another configuration combines the partitions of sizes 128 and 128 together first to result in a total delay of $1 + \max(8 + \max(28, 28), 36) = 37$ units.

In order to determine the order for merging the partitions, it is observed that in an optimal partitioning, all the partitions have sizes between $2^{\lceil \log_2(m-1) \rceil}$ and $2^{\lceil \log_2 m \rceil}$ except in one case in which $p - 1$ partitions have sizes $2^{\lceil \log_2 m \rceil}$ and one partition S_p has size between $2^{\lceil \log_2 m \rceil}$ and $2^{\lceil \log_2 m \rceil + 1}$. This is shown in the proof of Theorem 4. Therefore, the sizes of partitions can differ by a maximum factor of 2. In this case, the maximum difference in delay between O–E sorting the smallest and largest partitions cannot exceed the delay of a merging module since from (5) and (8),

$$D^T_{\mathrm{OE}}(2x) - D^T_{\mathrm{OE}}(x) = D^M_{BIT}(x) + 1 = \lceil \log_2 x \rceil + 1. \qquad (14)$$

The merging modules are connected into a binary tree of $\lceil \log_2 p \rceil$ levels. The delay of the tree is $\lceil \log_2 p \rceil (\lceil \log_2 m \rceil + 1) - \lceil \log_2 m \rceil$ units. The second term is due to the fact that no bitonic merging is necessary in the last stage. If p is a power of 2, all the path lengths equal $\lceil \log_2 p \rceil$; otherwise, there exists one or more paths with length $< \lceil \log_2 p \rceil$. To minimize the overall delay, the partitions are attached to the terminals in any order except in the case in which the number of partitions is not a power of 2 and S_p, the largest partition, has a size between $2^{\lceil \log_2 m \rceil}$ and $2^{\lceil \log_2 m \rceil + 1}$ [with delay $\binom{\lceil \log_2 m \rceil + 2}{2}$—(8)]. In this case, S_p should be located as the terminal node of a shorter path from the root of the binary tree. The extra delay of S_p is offset by the reduced path delay of the tree. In summary, the delay of the selection network is

$$\lceil \log_2 p \rceil (\lceil \log_2 m \rceil + 1) - \lceil \log_2 m \rceil + \begin{cases} \binom{\lceil \log_2 m \rceil + 1}{2} & (p \text{ is not a power of 2) or} \\ & (p \text{ is a power of 2 and } |S_p| \leq 2^{\lceil \log_2 m \rceil}) \\ \binom{\lceil \log_2 m \rceil + 2}{2} & (p \text{ is a power of 2 and } |S_p| > 2^{\lceil \log_2 m \rceil}). \end{cases}$$

B. Bounds on the Number of Partitions with Minimum Delay

The optimally partitioned selection network with minimum comparators does not always result in a network with minimum delay. As an example, consider the selection of 35 out of 175 numbers, the optimal partitions with minimum comparators are 35, 35, 35, 35, 35 with a delay of 36 units and 1555 comparators. If four partitions 39, 40, 48, 48 are used, the delay is 29 units with 1568 comparators. The following theorem shows the bounds on the number of partitions for minimum-delay networks.

Theorem 5: The optimal number of partitions for minimizing delay P^D_{opt} has a lower bound

$$P^D_L = \lfloor n/2^{\lceil \log_2 m \rceil} \rfloor$$

and an upper bound

$$P^D_U = \begin{cases} \lfloor (n/m) \rfloor & 2^{\lfloor \log_2(n/m) \rfloor} < P^D_L \\ 2^{\lfloor \log_2(n/m) \rfloor} & 2^{\lfloor \log_2(n/m) \rfloor} \geq P^D_L. \end{cases}$$

Proof: The lower bound can be proved by showing that delay cannot improve with the elimination of one or more partitions. The proof is very similar to that of Theorem 4 and will not be illustrated here. To prove the upper bound, the maximum number of partitions is $\lfloor n/m \rfloor$. It is easy to show that $\lfloor n/m \rfloor \geq 2^{\lfloor \log_2(n/m) \rfloor}$. If $2^{\lfloor \log_2(n/m) \rfloor} \geq P^D_L$, then the upper bound can be reduced from $\lfloor n/m \rfloor$ to $2^{\lfloor \log_2(n/m) \rfloor}$ resulting in one less level in the tree of merging modules and no increase in the delay for O–E sorting each partition. To further reduce the delay of the tree of merging modules, the upper bound has to be halved, resulting in $2^{\lfloor \log_2(n/m) \rfloor - 1} < P^D_L$ partitions which is impossible. On the other hand, if $2^{\lfloor \log_2(n/m) \rfloor} < P^D_L$, the upper bound cannot be reduced to result in a smaller delay. P^D_U is set to $\lfloor n/m \rfloor$. ■

V. Comparison to Previous Algorithms

In this section, we present some performance results of the proposed selection algorithm and compare them to Alekseyev's and Yao's selection algorithms.

The proposed network has a delay complexity of $O(\lceil \log_2 n \rceil \cdot \lceil \log_2 m \rceil)$ and hardware complexity of $O(n \cdot \lceil \log_2^2 m \rceil)$. As compared to Alekseyev's algorithm, there is a constant improvement in hardware complexity and a speedup of $O(\lceil \log_2 n \rceil / \lceil \log_2 m \rceil)$. The proposed algorithm has guaranteed performance for hardware and delay and for all values of m. In contrast, Yao's selection algorithms have better performance when $m < \sqrt{n}$ and worse performance than pure sorting when m approaches $n/2$. Furthermore, Yao's algorithms minimize either hardware or delay complexities, but not both simultaneously. Yao's networks are aimed towards the development of bounds on performance. Although our proposed network performs worse on individual measure for $m < \sqrt{n}$, it has better overall delay-hardware characteristics and represents a practical approach to the problem. This is demonstrated in the simulation results.

In Table II, the complexity of our proposed optimal design procedure is compared to exhaustive enumeration. It is seen that exhaustive enumeration is impossible to use practically. Further, iterations for the proposed design procedure are not monotonically decreasing because they depend on the range between the lower and upper bounds of the number of partitions.

TABLE II
Comparison of Exhaustive Enumeration and the Proposed Optimal Design Algorithm for n = 100. (Iterations in the Proposed Algorithm are the Number of Recursive Calls to Procedure *OPART*.)

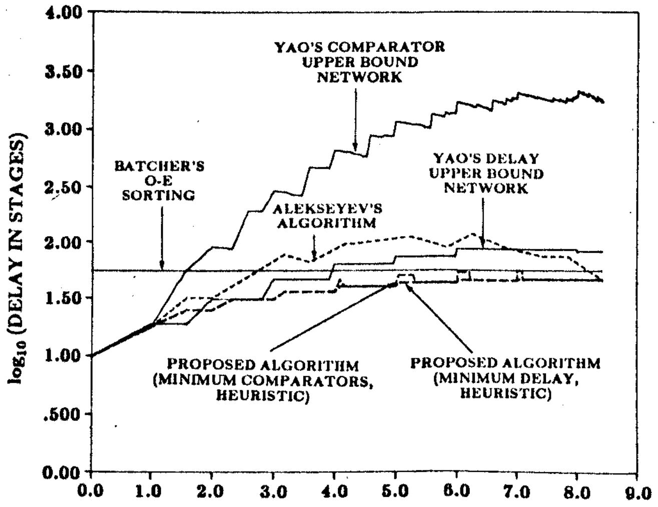
	Exhaustive Enumerations	Proposed Optimal Algorithm				
			# of partitions		Min. # of	Delay
m	Iterations	Iterations	Lowerbound	Upperbound	Comparators	in stages
2	21339417	50	50	50	196	12
3	4372211	261	25	33	243	16
4	1237284	25	25	25	313	16
6	175536	184	12	16	382	19
8	39829	19	12	12	458	19
10	12149	226	6	10	510	21
16	1301	28	6	6	599	21
20	306	43	5	5	682	22
25	93	4	3	4	735	22
50	2	2	1	2	840	22
100	1	1	1	1	1077	28

In Figs. 2 and 3, the performance of the selection networks for n = 684 and m between 1 and 342 are plotted. The value 684 is picked randomly. The curves for m between 343 and 684 are symmetric with respect to m = 342. These design procedures were implemented on a VAX 11/780. The simulations took 7.5 h for the optimal procedure using *OPART* and 0.6 h for the heuristic using *HPART*.

In Fig. 2, the number of comparators for the various methods are compared. It is seen that optimization under minimum comparators and minimum delay result in almost the same number of comparators ($\simeq$1 percent deviation on the average). The heuristic method also gives excellent results, and the error is less than 1 percent on the average. In contrast, Alekseyev's method requires significantly more comparators than our proposed method.[1] The comparators needed for Yao's hardware- and delay-upper-bound networks are also compared. It is seen that Yao's hardware-upper-bound network performs better for $3 \leq m \leq 79$ with n = 684. Yao's delay-upper-bound network has worse hardware requirement than his hardware-upper-bound network. As a comparison, the hardware for pure O–E sorting is also shown. Yao's networks are worse than O–E sorting when m approaches $n/2$.

In Fig. 3, the delays of the different networks are compared. It is seen that the delay for the minimum-delay network is monotonically increasing while the delay for the minimum-comparator network is not. This is due to the fact that the number of partitions in a minimum-comparator network can be increased beyond a power of two which incurs an additional stage of delay in the tree of bitonic merging modules. The delay curves for the heuristics are also very close to those of the optimal algorithms. As a comparison, the delays for Alekseyev's and Yao's algorithms are plotted.

[1]The performance curves of Alekseyev's algorithm are discrete because m must be a factor of n.



$\log_2$ (NUMBER OF ELEMENTS SELECTED)

Fig. 3. Delay in stages of different parallel selection algorithms (n = 684).

Alekseyev's algorithm may require less comparators than pure sorting, but at the expense of additional delays. Yao's networks have worse delays for $m \geq 4$.

In summary, our proposed network has better hardware-delay characteristics than other approaches. While maintaining hardware and delays to be always less than pure sorting (which other networks cannot), the network represents a practical approach to the parallel selection problem.

VI. Conclusion

In this paper, we have presented a practical design of selection networks. The set of n numbers are first partitioned and O–E sorted independently before they are bitonically merged together. To minimize the number of comparators or delay of the network, the number of partitions and the size of each partition must be succinctly chosen. Exhaustive enumeration is shown to be impractical.

We have proved the necessary condition for optimal partitioning and have found the range on the number of partitions in which optimal partitioning occurs. Using these results, design procedures are proposed to find the size of each partition. The selection network designed has a delay complexity of $O(\lceil \log_2 n \rceil \cdot \lceil \log_2 m \rceil)$ and hardware complexity of $O(n \cdot \lceil \log_2^2 m \rceil)$. The proposed network always maintains less hardware and delay than pure sorting. It is compared to Alekseyev's and Yao's selection networks and found to have better hardware-delay characteristics.

The partitioning approach is not limited to the network model used here. The adaptive model of Valiant and Preparata can be applied to sort and merge the lists. Using Preparata's scheme [5], a list of size m can be sorted in $O(\log m)$ time and $O(m \log m)$ hardware. With Valiant's method [6], two sorted lists of size m each can be merged together in $2 \log \log m + O(1)$ time and m processors. Of course, the basic hardware in the adaptive model is a processor. Further, there is no penalty for memory-processor alignment, and the overhead corresponding to the reassignment of sets of processors to subsequences to be merged is ignored. An (m, n) selection can, thus, be done in $O(\log(n/m) \log \log m)$ time with $n \log m$ processors. However, these results cannot be compared directly to our earlier results because different hardware units are used. An open problem at this time is to develop a parallel selection algorithm on multiprocessors.

Although we have proposed a very efficient design, our design is only optimal with respect to the partitioning approach. It is by no means the optimal design in general. The problem here is similar to the problem of designing optimal sorting networks which is still open at this time. Future research is necessary in this direction.

Appendix—Proof of Theorem 1

The proof is done by first showing

$$\text{(i)} \quad \min_{0 \leq n_1 \leq n/2} \left\{ C_{\mathrm{OE}}^T(n_1) + C_{\mathrm{OE}}^T(n - n_1) \right\} \geq C_{\mathrm{OE}}^T(n_L) + C_{\mathrm{OE}}^T(n - n_L) \qquad n_L \leq \frac{n}{2} < n_M \qquad \text{(A-1)}$$

$$\text{(ii)} \quad \min_{0 \leq n_1 \leq n/2} \left\{ C_{\mathrm{OE}}^T(n_1) + C_{\mathrm{OE}}^T(n - n_1) \right\} \geq C_{\mathrm{OE}}^T(n_U) + C_{\mathrm{OE}}^T(n - n_U) \qquad n_M \leq \frac{n}{2} < n_U. \qquad \text{(A-2)}$$

The proof is completed by showing the equivalence between points on the solid and dotted lines in the region

$$\frac{n_L + n_M}{2} \leq \frac{n}{2} < \frac{n_M + n_U}{2}$$

(refer to Fig. 1).

We show the proof of (A-1). The proof is by induction on $n/2$.

Basis: Let $n_L = 2, n_M = 3, n_U = 4$,

(i) For $2 \leq \frac{n}{2} < 3$

$$\min\{C_{\mathrm{OE}}^T(4), C_{\mathrm{OE}}^T(1) + C_{\mathrm{OE}}^T(3), C_{\mathrm{OE}}^T(2) + C_{\mathrm{OE}}^T(2)\} = C_{\mathrm{OE}}^T(2) + C_{\mathrm{OE}}^T(2) = 2.$$

(ii) For $3 \leq n/2 < 4$

$$\min\{C_{\mathrm{OE}}^T(6), C_{\mathrm{OE}}^T(1) + C_{\mathrm{OE}}^T(5), C_{\mathrm{OE}}^T(2) + C_{\mathrm{OE}}^T(4), C_{\mathrm{OE}}^T(3) + C_{\mathrm{OE}}^T(3)\} = C_{\mathrm{OE}}^T(4) + C_{\mathrm{OE}}^T(2) = 6.$$

Induction Hypothesis: Assume that (A-1) and (A-2) are true for any $(n/2) \geq 4$.

Induction Step: Consider $\tilde{n} = 2n$. Define

$$\tilde{n}_L = 2^{\lfloor \log_2 \tilde{n} \rfloor}, \qquad \tilde{n}_U = 2^{\lceil \log_2(\tilde{n}+1) \rceil}, \qquad \tilde{n}_M = \frac{\tilde{n}_L + \tilde{n}_U}{2}.$$

It is easy to show that

$$n_L = \frac{\tilde{n}_L}{2}, \qquad n_U = \frac{\tilde{n}_U}{2}, \qquad n_M = \frac{\tilde{n}_M}{2}.$$

For $\tilde{n}_L \le \tilde{n}/2 < \tilde{n}_M$, this implies $n_L \le n/2 < n_M$. Referring to (A-1),

$$\text{LHS} = \min_{0\le\tilde{n}_1\le\tilde{n}/2}\left\{C^T_{OE}(\tilde{n}_1) + C^T_{OE}(\tilde{n}-\tilde{n}_1)\right\}.$$

Expanding using (6),

$$\begin{aligned}\text{LHS} = \min_{0\le\tilde{n}_1\le\tilde{n}/2}&\left\{C^T_{OE}\left(\left\lceil\frac{\tilde{n}_1}{2}\right\rceil\right) + C^T_{OE}\left(\left\lfloor\frac{\tilde{n}_1}{2}\right\rfloor\right)\right.\\ &+ C^M_{OE}\left(\left\lceil\frac{\tilde{n}_1}{2}\right\rceil, \left\lfloor\frac{\tilde{n}_1}{2}\right\rfloor\right) + C^T_{OE}\left(\left\lceil\frac{\tilde{n}-\tilde{n}_1}{2}\right\rceil\right)\\ &\left.+ C^T_{OE}\left(\left\lfloor\frac{\tilde{n}-\tilde{n}_1}{2}\right\rfloor\right) + C^M_{OE}\left(\left\lceil\frac{\tilde{n}-\tilde{n}_1}{2}\right\rceil, \left\lfloor\frac{\tilde{n}-\tilde{n}_1}{2}\right\rfloor\right)\right\}.\end{aligned}$$

Since $\tilde{n}$ is even ($\tilde{n} = 2n$), it can be shown that

$$\left\lceil\frac{\tilde{n}_1}{2}\right\rceil + \left\lfloor\frac{\tilde{n}-\tilde{n}_1}{2}\right\rfloor = \left\lfloor\frac{\tilde{n}_1}{2}\right\rfloor + \left\lceil\frac{\tilde{n}-\tilde{n}_1}{2}\right\rceil = \frac{\tilde{n}}{2} = n.$$

By using the induction hypothesis,

$$\begin{aligned}\min_{0\le\tilde{n}_1\le\tilde{n}/2}&\left\{C^T_{OE}\left(\left\lceil\frac{\tilde{n}_1}{2}\right\rceil\right) + C^T_{OE}\left(\left\lfloor\frac{\tilde{n}-\tilde{n}_1}{2}\right\rfloor\right)\right\}\\ &= \min_{0\le\tilde{n}_1\le\tilde{n}/2}\left\{C^T_{OE}\left(\left\lfloor\frac{\tilde{n}_1}{2}\right\rfloor\right) + C^T_{OE}\left(\left\lceil\frac{\tilde{n}-\tilde{n}_1}{2}\right\rceil\right)\right\}\\ &= C^T_{OE}(n_L) + C^T_{OE}(n-n_L).\end{aligned}$$

By using induction, it can be proved separately that

$$\begin{aligned}\min_{0\le\tilde{n}_1\le\tilde{n}/2}&\left\{C^M_{OE}\left(\left\lceil\frac{\tilde{n}_1}{2}\right\rceil, \left\lfloor\frac{\tilde{n}_1}{2}\right\rfloor\right) + C^M_{OE}\left(\left\lceil\frac{\tilde{n}-\tilde{n}_1}{2}\right\rceil, \left\lfloor\frac{\tilde{n}-\tilde{n}_1}{2}\right\rfloor\right)\right\}\\ &\ge C^M_{OE}(n_L, n_L) + C^M_{OE}(n-n_L, n-n_L).\end{aligned}$$

Since $\min(x+y) \ge \min(x) + \min(y)$, therefore, using (6),

$$\begin{aligned}\text{LHS} &\ge 2C^T_{OE}(n_L) + 2C^T_{OE}(n-n_L) + C^M_{OE}(n_L, n_L)\\ &+ C^M_{OE}(n-n_L, n-n_L) = C^T_{OE}(\tilde{n}_L) + C^T_{OE}(\tilde{n}-\tilde{n}_L) = RHS.\end{aligned}$$

To complete the proof of (A-1), the case of $\tilde{n} = 2n + 1$ must also be considered. The proof is similar. Likewise, the proof of (A-2) is similar and will not be illustrated here. By the theory of induction, (A-1) and (A-2) are proved.

To complete the proof, the equivalence between points on the solid and dotted lines in the region

$$\frac{n_L + n_M}{2} \le \frac{n}{2} \le \frac{n_M + n_U}{2}$$

(Fig. 1) must be shown. The proof is also done by induction on $n/2$. The approach is similar and will not be repeated here. ■

REFERENCES

[1] V. E. Alekseyev, "Sorting algorithms with minimum memory," *Kibern.*, vol. 5, no. 5, pp. 99–103, 1969.

[2] K. E. Batcher, "Sorting networks and their applications," in *Proc. 1968 Spring Joint Comput. Conf.*, AFIPS Press., vol. 32, 1968, pp. 307–314.

[3] M. Blum, R. Floyd, V. Pratt, R. Rivest, and R. Tarjan, "Time bounds for selection," *J. Comput. Syst. Sci.*, vol. 7, no. 4, pp. 448–461, 1972.

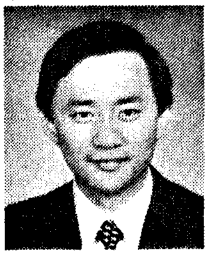
[4] D. E. Knuth, *The Art of Computer Programming, Vol. 3: Sorting and Searching*. Reading, MA: Addison-Wesley, 1973.

[5] F. P. Preparata, "New parallel-sorting schemes," *IEEE Trans. Comput.*, vol. C-27, pp. 669–673, July 1978.

[6] L. G. Valiant, "Parallelism in comparison problems," *SIAM J. Comput.*, vol. 4, no. 3, pp. 348–355, Sept. 1975.

[7] B. W. Wah and Y. W. Ma, "MANIP—A multi-computer architecture for solving combinatorial extremum search problems," *IEEE Trans. Comput.*, to be published.

[8] A. C. C. Yao, "Bounds on selection networks," *SIAM J. Comput.*, vol. 9, no. 3, pp. 566–582, Aug. 1980.



Benjamin W. Wah (S'74–M'79) received the B.S. and M.S. degrees in electrical engineering and computer science from Columbia University, New York, NY, in 1974 and 1975, and the M.S. degree in computer science and the Ph.D. degree in engineering both from the University of California, Berkeley, in 1976 and 1979, respectively.

Currently, he is an Assistant Professor in the School of Electrical Engineering, Purdue University, West Lafayette, IN. His current research interests include parallel computer architecture, distributed databases, and theory of computing.

Dr. Wah has been a Distinguished Visitor of the IEEE Computer Society since 1983.

Kuo-Liang Chen graduated with honors from Chiao-Tung University in 1961 majoring in digital computers.

Since 1961 he has been researching and teaching logical design and computer architecture at the Institute of Special-Purpose Computer, Beijing, and the University of Science and Technology of China, Hefei, Anhui, China, respectively. From 1981 to 1983, he was on leave as a visiting scientist at the Department of Computer and Information Sciences, University of Florida, Gainesville, FL, and the School of Electrical Engineering, Purdue University, West Lafayette, IN. His research interests include parallel/distributed processing, interconnection networks, and data flow computation.