Periodic Sorting Using Minimum Delay, Recursively Constructed Merging Networks

Edward A. Bender
Center for Communications Research
4320 Westerra Court
San Diego, CA 92121, USA
ed@ccrwest.org

S. Gill Williamson
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0114, USA
gwilliamson@ucsd.edu

Submitted: December 6, 1996
Submitted in revised form August 25, 1997
Accepted: December 9, 1997

Abstract

Let α and β be a partition of $\{1, \ldots, n\}$ into two blocks. A merging network is a network of comparators which allows as input arbitrary real numbers and has the property that, whenever the input sequence $x_1, x_2, \ldots, x_n$ is such that the subsequence in the positions α and the subsequence in the positions β are each sorted, the output sequence will be sorted. We study the class of "recursively constructed" merging networks and characterize those with delay $\lceil \log_2 n \rceil$ (the best possible delay for all merging networks). When n is a power of 2, we show that at least $3^{n/2-1}$ of these nets are log-periodic sorters; that is, they sort any input sequence after $\log_2 n$ passes through the net. (Two of these have appeared previously in the literature.)

1991 AMS Classification Number. Primary: 68P10

1. Introduction

This paper is divided into two main parts in two ways. First, Sections 1–5 contain the concepts and results and Sections 6–10 contain the proofs. Second, one part of the paper deals with merging and the other with sorting. The concepts mentioned in the present section will be made precise in the next section.

In software terms, a merging network is a program with no branching, looping, or arithmetic other than the replacement of a pair of values (x, y) with $c(x, y) = \big(\min(x, y), \max(x, y)\big)$, called a comparator. In hardware terms, a merging network is a branch-free and feedback-free circuit whose only logic units are comparators. Given two "interleaved" sorted sequences, a merging net sorts the entire sequence. A sorting net is like a merging net except that the input is an arbitrary sequence and the output is sorted. Since comparators may operate in parallel when there is no overlap of inputs, a considerable amount of parallelism is possible. If a comparator takes one time unit, the delay of a net is its running time when the most efficient parallelism is used.

The problem of designing n-input merging and sorting nets having minimum delay or a minimum number of comparators has been studied by many authors. Knuth [8, Sec. 5.3.4] discussed the history and results concerning sorting and merging nets up to 1973. Aigner [1, Thm. 3.3] showed that the best merging nets have delay $\lceil \log_2 n \rceil$ provided neither of the sequences being merged is empty. It has recently been shown by Miltersen, Paterson, and Tarui [9] that a network for merging an m-long sequence and an n-long one requires $\frac{1}{2}(m+n)\log_2 m + O(n)$ comparators provided $n \geq m$ and $m \to \infty$.

A simple information-theoretic argument shows that at least $\log_2(n!)$ comparators are needed to sort n items. By Stirling's formula, it follows that the number of comparators is at least $n \log n + O(n)$. Since at most $\lfloor n/2 \rfloor$ comparators can be executed simultaneously, the delay of such a sorting net must be at least $\log n + O(1)$. Until Ajtai, Komlos, and Szemeredi [2] showed that there are networks for sorting n items having delay on order of $\log n$ and using on order of $n \log n$ comparators, researchers were unable to approach such bounds. Since all known families of nets of this type are quite complicated and have very large factors multiplying both $\log n$ and $n \log n$, it is natural to ask for simpler networks. Some families have been found with delay times $(\log n + O(1))^2$. Of particular interest are two found by Dowd, Perl, Rudolph, and Saks [5] and Canfield and Williamson [3] that consist of $\lceil \log_2 n \rceil$ repetitions of a merging network. Such a net is called an $\lceil \log_2 n \rceil$-pass periodic sorter. Kammeyer, Belew, and Williamson [7] conjectured two additional such families based on empirical studies. A pictorial representation of these nets for $n = 16$ is given at the end of the next section at the end.

In the present paper, we study a natural class of n-input merging nets which we call recursive, focusing on those with minimum delay. We characterize the structure of these nets and show that they achieve the best possible delay, namely $\lceil \log_2 n \rceil$. When n is a power of two and the two sorted input sequences have length $n/2$, we show that

(a) the least number of comparators needed is $(n/2)\log_2(n/2) + 1$, which achieves

the asymptotic best possible bound of Miltersen et al, and

(b) at least $3^{n/2-1}$ of the nets sort after $\log_2 n$ passes, thereby including the known and conjectured results of the previous paragraph.

2. Definitions

Unfortunately, a variety of concepts are required. Those needed for stating our main results are collected in this section.

- **regular expression notation**: Let S be either a set of sequences or a single sequence. In the latter case, we identify S with $\{S\}$. If T is defined similarly, then ST is the set of concatenations of pairs of sequences, one from S and one from T. In particular, S^k is the set of sequences formed by concatenating k elements of S with repetition allowed. Also, S^+ is the union of S^k over all $k > 0$, and S^* is S^+ with the empty sequence adjoined. When it will not lead to confusion we sometimes abuse notation by letting a set of sequences stand for some element of the set.
- **(adjacent) comparator**: A *comparator* is a function $c : \mathbb{R}^2 \to \mathbb{R}^2$ with

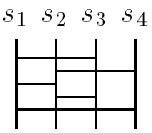
$$c(x, y) = \Big(\min(x, y), \max(x, y)\Big).$$

  If $u \in \mathbb{R}^n$ and $1 \le i < j \le n$, then $v = c_{i:j}(u) \in \mathbb{R}^n$ is given by $v_k = u_k$ if $k \ne i, j$ and $(v_i, v_j) = c(u_i, u_j)$. We also call $c_{i:j}$ a comparator. If $j = i + 1$, it is an *adjacent comparator*.
- **network**: A *network of comparators*, or simply a *net*, is a sequence of comparators $c_{i_1:j_1}, c_{i_2:j_2}, \cdots, c_{i_k:j_k}$. A function $f : \mathbb{R}^n \to \mathbb{R}^n$ given by

$$f(s) = c_{i_k:j_k}\big(\cdots c_{i_1:j_1}(s)\cdots\big)$$

  is associated with the net. In other words, the function f is a composition of the comparators $c_{i:j}$. We call s the *input* and $f(s)$ the *output* of the net.

Here is a pictorial representation of the net $c_{1:3}, c_{2:4}, c_{1:2}, c_{2:3}, c_{1:4}$ for $\mathbb{R}^4$. The inputs are shown being "fed in" at the top and comparators are represented by horizontal bars. Outputs emerge at the bottom.



(1)

The inputs and outputs of a net may be indexed by a set σ other than $\{1, 2, \ldots, n\}$. To keep track of the set, we refer to a net for σ.

- **layer**: A sequence of comparators may be executed in parallel if and only if it contains no repeated subscripts. We call such a collection of comparators a *layer*. In (1), $c_{1:3}$ and $c_{2:4}$ can be executed in parallel, but $c_{1:2}$ and $c_{2:3}$ cannot.

- **delay**: The *delay* of net is the minimum number of layers needed to represent the net. The delay of (1) is 3 and the 3 layers are $\{c_{1:3}, c_{2:4}\}$, $\{c_{1:2}\}$, and $\{c_{2:3}, c_{1:4}\}$.
- **(trivial) partition** $\vdash$: If σ is a set of integers, $|\sigma|$ denotes its size and $\sigma \vdash \{\alpha, \beta\}$ is a *partition* of σ into two, possibly empty, subsets. If either α or β is empty, the partition is *trivial*.
- **induced partitions**: If $\sigma \vdash \{\alpha, \beta\}$ and $\sigma \vdash \{\sigma^{(1)}, \sigma^{(2)}\}$, let $\alpha^{(i)} = \alpha \cap \sigma^{(i)}$ and $\beta^{(i)} = \beta \cap \sigma^{(i)}$. The *induced partitions* are $\alpha \vdash \{\alpha^{(1)}, \alpha^{(2)}\}$, $\beta \vdash \{\beta^{(1)}, \beta^{(2)}\}$, $\sigma^{(1)} \vdash \{\alpha^{(1)}, \beta^{(1)}\}$, and $\sigma^{(2)} \vdash \{\alpha^{(2)}, \beta^{(2)}\}$.
- **(bi)alternating**: Suppose $\delta \vdash \{\epsilon, \zeta\}$ is such that, when the elements of δ are listed in order, ϵ contains every other element of δ (and hence so does ζ). We say that the partition $\delta \vdash \{\epsilon, \zeta\}$ is *alternating*. We call $\sigma \vdash \{\sigma^{(1)}, \sigma^{(2)}\}$ *bialternating* (with respect to α and β) if the induced partitions of α and β are each alternating.

For example listing the elements of σ in order and using subscripts to denote elements of $\alpha^{(i)}$ and $\beta^{(i)}$,

$$\sigma = \alpha_1^{(1)}, \beta_1^{(2)}, \beta_1^{(1)}, \beta_2^{(2)}, \alpha_1^{(2)}, \alpha_2^{(1)}, \alpha_2^{(2)}, \beta_2^{(1)} \tag{2}$$

is bialternating, but

$$\sigma = \alpha_1^{(1)}, \alpha_1^{(2)}, \beta_1^{(1)}, \alpha_2^{(2)}, \beta_1^{(2)}, \alpha_2^{(1)}, \beta_2^{(2)}, \beta_2^{(1)} \tag{3}$$

is not. If $\sigma = \{1, \ldots, 8\}$, the induced partitions of α and β are $\alpha \vdash \big\{\{1,6\},\ \{5,7\}\big\}$ and $\beta \vdash \big\{\{2,4\},\ \{3,8\}\big\}$ for (2), and $\alpha \vdash \big\{\{1,6\},\ \{2,4\}\big\}$ and $\beta \vdash \big\{\{3,8\},\ \{5,7\}\big\}$ for (3).

- **merger**: A *merging net*, or *merger*, for $\sigma \vdash \{\alpha, \beta\}$ is a net such that, if the subsequence of the input indexed by α is sorted and likewise for β, then the output is sorted.
- **recursive merger; correction subnet**: A merging net for $\sigma \vdash \{\alpha, \beta\}$ is *recursive* if either $\{\alpha, \beta\}$ is trivial (and so no comparators are needed) or there is a partition $\sigma \vdash \{\sigma^{(1)}, \sigma^{(2)}\}$ such that the net consists of recursive merging nets for $\sigma^{(1)}$ and $\sigma^{(2)}$ followed by an arbitrary comparator net, and the partition satisfies

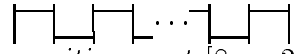
  (a) $\sigma \vdash \{\sigma^{(1)}, \sigma^{(2)}\} \neq \{\alpha, \beta\}$ is nontrivial;

  (b) if $|\alpha| \geq 2$ and $|\beta| \geq 2$, then the induced partitions $\alpha \vdash \{\alpha^{(1)}, \alpha^{(2)}\}$ and $\beta \vdash \{\beta^{(1)}, \beta^{(2)}\}$ are both nontrivial.

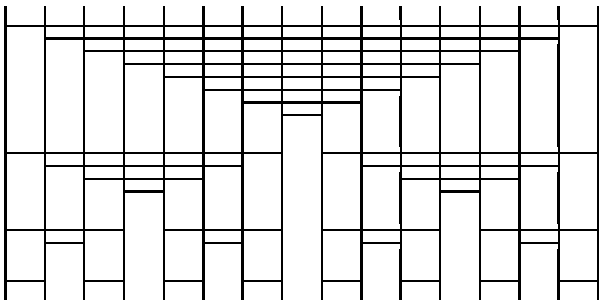
  The arbitrary comparator net following $\sigma^{(1)}$ and $\sigma^{(2)}$ is called the *correction subnet*.

Condition (a) can be restated as: At least one of the induced partitions $\alpha \vdash \{\alpha^{(1)}, \alpha^{(2)}\}$ and $\beta \vdash \{\beta^{(1)}, \beta^{(2)}\}$ is nontrivial, which is weaker than (b). Condition (a) prevents the trivial cases in which all the work is done before the correction subnet ($\{\sigma^{(1)}, \sigma^{(2)}\}$ trivial) or in the correction subnet ($\{\sigma^{(1)}, \sigma^{(2)}\} = \{\alpha, \beta\}$)

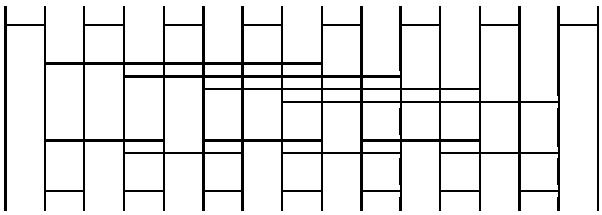
whereas (b) requires that, when possible, the work must also be divided between the merging nets for $\sigma^{(1)}$ and $\sigma^{(2)}$ as well.

- **periodic sorter**: A net is called a *k-pass periodic* (or *sequential*) *sorter* if passing a sequence through k copies of the net always produces sorted output. For example, the $2n$-input, delay-2 net  is an $(n/2)$-pass periodic sorter, called the *odd-even transposition sort* [8, p. 241].

Here is the Dowd, Perl, Rudolph, and Saks 4-pass periodic sorter for $n = 16$:



Here is the Canfield and Williamson 4-pass periodic sorter for $n = 16$:



3. Theorems About Recursive Merging Networks

Our main theorem on the structure of recursive merging nets is:

Theorem 3.1. *Suppose that* $\sigma = \{1, \ldots, n\} \vdash \{\alpha, \beta\}$ *with* $|\alpha| \geq 2$ *and* $|\beta| \geq 2$. *The following are true:*

(a) A minimum delay, recursive merging net for σ *has delay* $d = \lceil \log_2 n \rceil$.

(b) If there is a minimum delay recursive merging net associated with the partition $\sigma \vdash \{\sigma^{(1)}, \sigma^{(2)}\}$, *then this partition is bialternating and*

$$\left\lceil \max\bigl(\log_2 |\sigma^{(1)}|, \log_2 |\sigma^{(2)}|\bigr) \right\rceil < \left\lceil \log_2 |\sigma| \right\rceil. \tag{4}$$

(c) The following construction gives precisely the minimum delay recursive merging nets associated with a bialternating partition $\sigma \vdash \{\sigma^{(1)}, \sigma^{(2)}\}$.

(i) Suppose the minimum elements of α *and* β *both belong to* $\sigma^{(1)}$ *or both belong to* $\sigma^{(2)}$. *Then the correction subnet consists of* $c_{2k:2k+1}$ *for* $1 \leq k < |\sigma|/2$ *and, optionally when* $|\sigma|$ *is even,* $c_{1:|\sigma|}$.

(ii) Suppose one of the minimum elements of α and β belongs to $\sigma^{(1)}$ and the other to $\sigma^{(2)}$,. Then the correction subnet consists of $c_{2k-1:2k}$ for $1 \le k \le |\sigma|/2$.

(iii) The recursive merging nets for $\sigma^{(1)}$ and $\sigma^{(2)}$ each have delay at most $d \Leftrightarrow 1$.

Note that by (i) and (ii), the correction subnet has delay 1.

The condition in (4) rarely fails. Bialternating guarantees that $|\alpha^{(1)}|$ and $|\alpha^{(2)}|$ differ by at most one and likewise for $|\beta^{(1)}|$ and $|\beta^{(2)}|$. Hence $|\sigma^{(1)}|$ and $|\sigma^{(2)}|$ differ by at most 2, which can lead to failure of (4) when n is nearly a power of 2. Switching $\beta^{(1)}$ and $\beta^{(2)}$ if necessary, we can insure that $|\sigma^{(1)}|$ and $|\sigma^{(2)}|$ differ by at most 1 and, consequently, that (4) holds.

Theorem 3.1 enables us to say quite a bit about the structure of recursive merging networks with minimum delay: As long as $|\alpha| \ge 2$ and $|\beta| \ge 2$ there are only two ways to partition the sequence σ for recursion and the correction subnet is practically determined. In this case, if

$$\lceil \log_2 |\sigma^{(i)}| \rceil = \lceil \log_2 |\sigma| \rceil \Leftrightarrow 1, \tag{5}$$

then the net for $\sigma^{(i)}$ is also minimum delay. From Theorem 3.1(b), there are only two ways in which (5) can fail to hold for $i = 1, 2$:

(a) $|\sigma| = 2^k + 1$ for some k, in which case the values of $|\sigma^{(i)}|$ are $2^{k-1} + 1$ and 2^{k-1}.

(b) $|\sigma| = 2^k + 2$ for some k, in which case the values of $|\sigma^{(i)}|$ can be $2^{k-1} + 2$ and 2^{k-1}.

The following corollary avoids these problems by limiting $|\sigma|$ to powers of 2.

Corollary 3.1.1. *Suppose $|\sigma| = n = 2^k$. Fix a partition $\sigma \vdash \{\alpha, \beta\}$ with $|\alpha| = |\beta| = n/2$. Given a sequence $s_1, s_2, \ldots$, call a maximal subsequence whose successive indices differ by t a distance-t subsequence. The following are true.*

(a) The minimum delay of a recursive merging net for σ is $k = \log_2 n$.

(b) Every minimum delay recursive merging net for σ has the following structure:

- *$\Leftrightarrow$ The top j layers together form 2^{k-j} disjoint minimum delay recursive merging nets, each of which has 2^{j-1} inputs from α and 2^{j-1} inputs from β. Furthermore, the 2^{j-1} elements from α (resp. β) are a distance-2^{k-j} subsequence of the sequence α (resp. β).*
- *$\Leftrightarrow$ The first layer consists of $n/2$ comparators, each comparing an α and a β.*
- *$\Leftrightarrow$ For $j > 1$, the jth layer of the net consists of 2^{k-j} parallel correction subnets as described in Theorem 3.1(c).*

(c) The number of such nets is $3^{n/2-1}$. If we disallow the optional comparators (the $c_{1:|\sigma|}$ of Theorem 3.1(i)), then the number of nets is $2^{n/2-1}$.

(d) The minimum number of comparators in a minimum delay recursive merging net for σ is $\frac{(k-1)n}{2} + 1$ and there is exactly one such net.

Two aspects of the corollary may be misleading. First, (d) does not claim that all $\frac{(k-1)n}{2} + 1$ comparators are essential. All comparators are essential *when they*

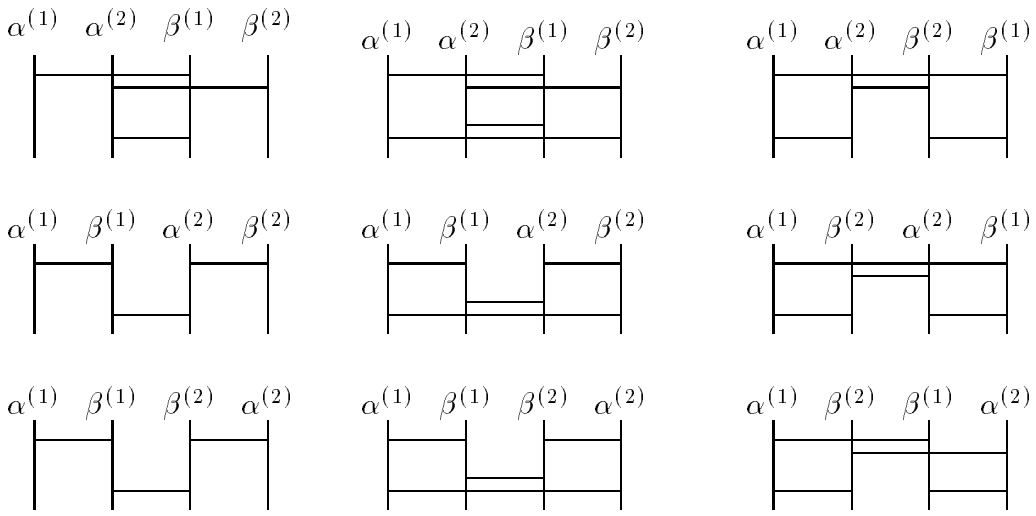
are first introduced in correction subnets as part of the the recursive construction. Later comparators may make some earlier comparators redundant. We believe that this does not happen, but have been unable to prove it.

Second, it appears to be a simple matter to count all delay-k recursive merging nets with $|\alpha| = |\beta| = 2^{k-1}$, but this depends on what is meant by "all" nets. If a merging net is defined to include both the partition and the comparators, then enumeration is trivial: There are $\frac{1}{2}\binom{n}{n/2}$ ways to partition σ into two equally long subseqences. (The factor of one-half occurs because we do not distinguish between α and β.) If a merging net is defined to be *only* the arrangement of comparators, then we are unable to count them. Problems arise because the same net can be obtained for more than one partition of σ. These partitions can be created by selectively switching $\alpha^{(2)}$ and $\beta^{(2)}$ at various levels in the recursion. Such a switch changes $\{\alpha, \beta\}$ but it does not change the net if

(a) the result is still bialternating and

(b) at all levels of the recursive construction, the use of (i) or (ii) in Theorem 3.1 is unchanged.

To illustrate the problem (a) causes, note our switching changes (2) to (3). We clarify (b) by considering nets for $\{1,2,3,4\}$ with $|\alpha| = |\beta| = 2$. Then $\{\alpha, \beta\}$ is one of $\big\{\{1,2\},\{3,4\}\big\}$, $\big\{\{1,3\},\{2,4\}\big\}$, and $\big\{\{1,4\},\{2,3\}\big\}$. We may assume $1 \in \alpha$. Here are the nine minimum delay recursive nets.



The rows in this array correspond to partitions $\{\alpha, \beta\}$ of $\sigma = \{1,2,3,4\}$, the first two columns correspond to cases in which Theorem 3.1(i) applies, and the last column to the cases in which Theorem 3.1(ii) applies. The input position labels indicate the set to which the position belongs. Refer to a net in this array as $N_{i,j}$ in the usual matrix fashion. Interchanging $\alpha^{(2)}$ and $\beta^{(2)}$ creates the following interchanges of identical nets $N_{1,3} \leftrightarrow N_{2,3}$, $N_{2,1} \leftrightarrow N_{3,1}$, and $N_{2,2} \leftrightarrow N_{3,2}$, and condition (b) holds. In contrast, the interchange $\alpha^{(2)} \leftrightarrow \beta^{(2)}$ interchanges the partition associated with $N_{3,3}$ and the partition associated with $N_{1,1}$ and $N_{1,2}$, condition (b) fails, and so the correction subnets change.

In Corollary 3.1.1, the only unused positions in a layer are those associated with the nonuse of the optional comparators in Theorem 3.1(i). In the next section, we present examples which show that connecting some of these positions can destroy a net's merging capability. The following result shows that some connections do not.

Theorem 3.2. *Let $|\alpha| = |\beta|$ be a power of two and consider the recursive merging nets described in Corollary 3.1.1. For the jth layer, let λ_j be the set of left ends of missing optional comparators and let ρ_j be the missing right ends. Adding comparators of the form $c_{l:r}$ with $l \in \lambda_j$ and $r \in \rho_j$ does not destroy the merging capability of the net. In fact, such comparators have no effect because their inputs are always already in order.*

Suppose $|\sigma| = n = 2^k$ and $\sigma \vdash \{\alpha, \beta\}$. At the jth layer up from the output, there are 2^{j-1} interleaved merging nets. If t of them could have optional comparators, $|\lambda| = |\rho| = t$ and so the number of ways to construct a set of i comparators $c_{l:r}$ is $\binom{t}{i}^2 i!$. It follows that the theorem leads to

$$\prod_{j=1}^{k-1}\left(\sum_{t=0}^{2^{j-1}}\binom{2^{j-1}}{t}\sum_{i=0}^{t}\binom{t}{i}^2 i!\right)$$

sorting nets for each partition $\sigma \vdash \{\alpha, \beta\}$. Messy asymptotic estimates show that this behaves like $(n/2)^{n/2}$, which is considerably larger than the $3^{n/2-1}$ of Corollary 3.1.1(c).

4. Theorems About Periodic Sorting Networks

We now turn our attention to periodic sorting nets built from recursive merging nets. For simplicity, we limit our attention to $|\alpha| = |\beta| = n/2$ a power of 2, say $n = 2^k$. Corollary 3.1.1 describes how all such recursive merging nets are constructed. In this case, two types of periodic k-pass sorters are known. We show that they are members of a larger family.

Since a network is a periodic sorter after a sufficiently large number of passes if and only if it contains all adjacent comparators [5, Thm. 1], two natural questions are:

(a) Which recursive merging nets contain all adjacent comparators?

(b) Of those nets in (a), which are k-pass periodic sorters?

The first question has a relatively simple answer, but the other appears more difficult. The fact that (a) is not sufficient will be illustrated by an example. We do not have a theorem similar to Theorem 3.2 for periodic sorting nets. However, Theorem 3.2 can be used to add comparators to sorting nets which are based on a series of merges, as is the case for the Batcher sort [8, Sec. 5.3.4].

Theorem 4.1. *If $|\sigma| = n = 2^k$ and $\sigma \vdash \{\alpha, \beta\}$ is alternating, then every recursive merging net for σ is a k-pass periodic sorter.*

By Corollary 3.1.1(c), there are $3^{n/2-1}$ such nets. We conjecture the nets in Theorem 3.2 are k-pass periodic sorters whenever $\sigma \vdash \{\alpha, \beta\}$ is alternating. This has been verified up to $|\sigma| = 16$ by computer.

The construction of a recursive merging net can be described by a simple tree constructed as follows:

$\Leftrightarrow$ At each vertex, a bialternating partition of the lines is constructed by placing the leftmost α line in $\alpha^{(1)}$ and the leftmost β line in $\beta^{(i)}$, according to the label i at the vertex.

$\Leftrightarrow$ The left son corresponds to $\sigma^{(1)}$ and the right to $\sigma^{(2)}$.

The periodic sorting nets given by Dowd, Perl, Rudolph, and Saks [5] correspond to trees having all vertices labeled 2. The nets given by Canfield and Williamson [3] correspond to trees having all vertices labeled 1 and having none of the optional comparators of Theorem 3.1(i). In both cases, $\sigma \vdash \{\alpha, \beta\}$ is alternating, and so these nets are included in Theorem 4.1. Kammeyer, Belew, and Williamson [7] discovered the two nets for $n = 16$ in which the root and its sons have one value and the remaining vertices have the other value. Based on these, they conjectured a general pattern which is included in the families in Corollary 3.1.1.

Theorem 4.2. *Call a partition* $\tau \vdash \{\tau^{(1)}, \tau^{(2)}\}$

$\Leftrightarrow$ *type 1 if listing* τ *in order gives an element of* $\left(t^{(1)}t^{(1)}t^{(2)}t^{(2)}\right)^*$, *or*

$\Leftrightarrow$ *type 2 if listing* τ *in order gives an element of* $\left(t^{(1)}t^{(2)}t^{(2)}t^{(1)}\right)^*$,

where $t^{(i)}$ *stands for any element of* $\tau^{(i)}$. *Other partitions have no type. Suppose* $|\alpha| = |\beta|$, *a power of 2. We will "mark" certain vertices of the tree described above and only consider the type of a marked vertex. The partition* $\sigma \vdash \{\sigma^{(1)}, \sigma^{2)}\}$ *of the input lines* σ *associated with a vertex determines its type, if any. Thus, each vertex will be labeled 1 or 2, and perhaps be marked and typed. The tree for a recursive merger has all adjacent comparators if and only if:*

(a) Every marked vertex has type equal to its label.

(b) The root is marked.

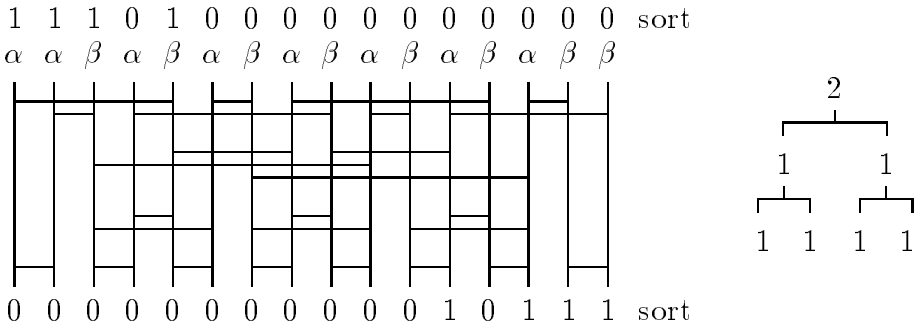
(c) For every marked non-leaf vertex:

(i) if its left son has the same label, it is also marked;

(ii) if its right son is labeled 1, it is also marked.

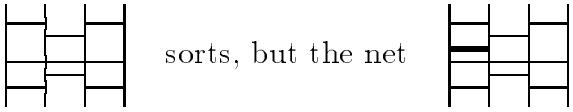
The following merging net has all adjacent comparators but is not a 4-pass sorter. The lines are labeled according to whether they are in α or β. An input sequence that causes failure and the corresponding output sequence are given. The tree associated with the net's construction is shown on the right. The root, its right

son, and the two rightmost leaves are marked.

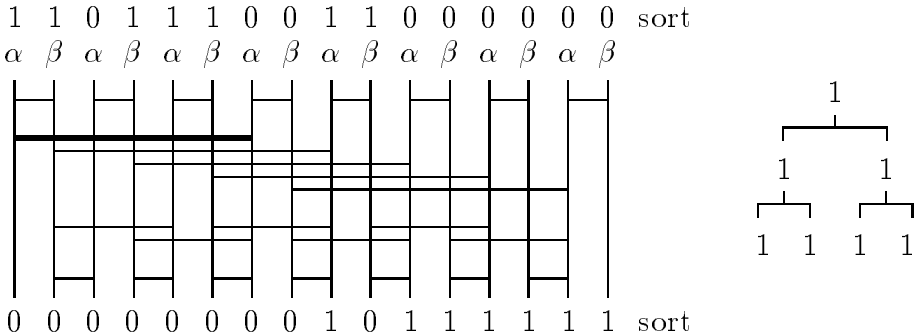


5. Adding Comparators Can Ruin a Net

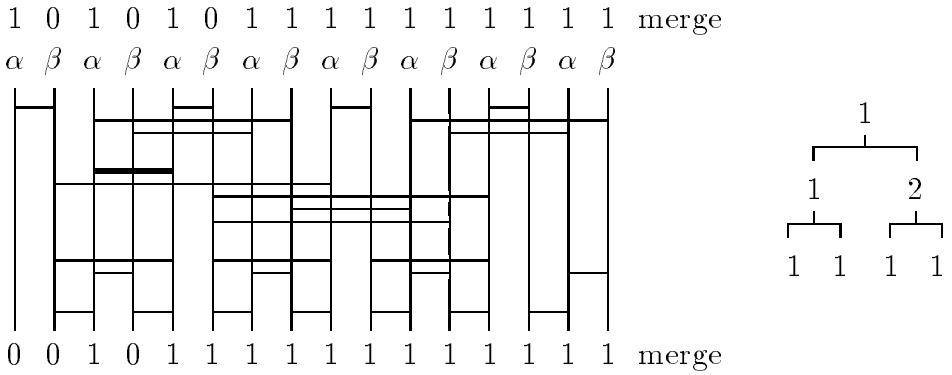
It is natural to suppose that adding comparators to a sorting or merging net will not destroy the ability of the net to sort or merge. It was shown by de Bruijn [4] that, when a sorting net contains only adjacent comparators, adding extra comparators does not destroy the ability of the net to sort. On the other hand, this is not true when nonadjacent comparators are allowed The net



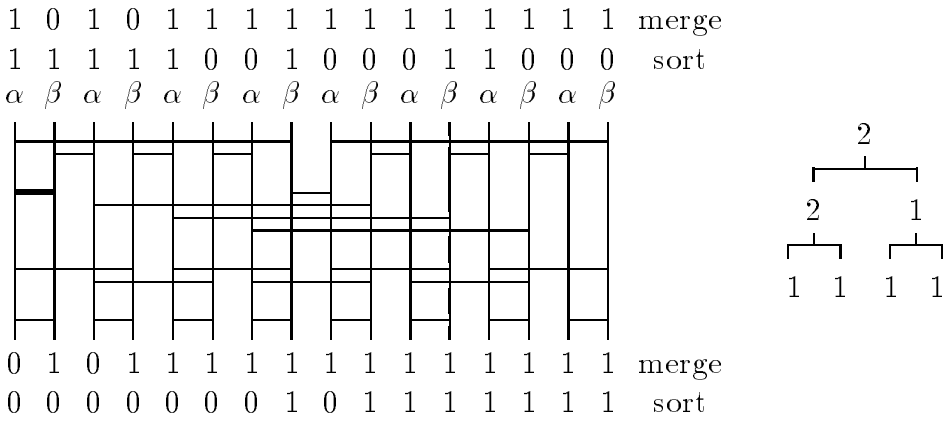
which has $c_{1:2}$ added, fails to sort the sequence $1, 1, 0, 0$. This simple counterexample is not recursively constructed and the added comparator increases the delay. There are some 16-input recursive merging nets that are 4-pass periodic sorters such that the addition of a comparator destroys either the merging, the 4-pass periodic sorting, or both. Examples of this were found by a computer, which was also used to verify sorting and merging capabilities. In the following three examples, the added comparator is indicated in bold face. In all cases, Theorems 3.1 and 4.1 insure that the original nets merge and are 4-pass periodic sorters. The following net is *not* a 4-pass periodic sorter, but is a merger.



The following net is a 4-pass periodic sorter, but is *not* a merger.



The following net is neither a 4-pass periodic sorter nor a merger. The upper inputs and outputs are for the merge and the lower are for the sort.



6. Proof of Theorems 3.1 and 3.2 and Corollary 3.1.1

Let $\sigma \vdash \{\alpha, \beta\}$ be nontrivial. We will prove the following two theorems in Sections 7 and 8.

Theorem 6.1. *If $\sigma \vdash \{\sigma^{(1)}, \sigma^{(2)}\}$ is bialternating, $|\alpha| \geq 2$, and $|\beta| \geq 2$, then Theorem 3.1(i) and (ii) describe the minimum delay correction subnets.*

Theorem 6.2. *If $\sigma \vdash \{\sigma^{(1)}, \sigma^{(2)}\}$ is not bialternating and the induced partitions of α and β are not trivial, then the correction subnet has delay greater than one.*

Proof (of Theorem 3.1): Without loss of generality, we may assume that $|\sigma^{(1)}| \geq |\sigma^{(2)}|$. Since being recursive requires that $\sigma \vdash \{\sigma^{(1)}, \sigma^{(2)}\} \neq \{\alpha, \beta\}$ be nontrivial, it is easily seen that the correction subnet cannot be empty. Since Aigner [1 Thm. 3.3] has shown that the delay of any n-input merging net for two nonempty sequences is at least $\lceil \log_2 n \rceil$, it follows the delay for σ is at least $\lceil \log_2 |\sigma^{(1)}| \rceil + 1$. Theorem 3.1(a) and (c) now follow easily by using induction on n and Theorem 6.1, provided (4) holds. The discussion immediately following the statement of Theorem 3.1 shows

that it is always possible to satisfy (4). Theorem 3.1(b) now follows easily from $d = \lceil \log_2 n \rceil$ and Theorem 6.2. □

Proof (of Corollary 3.1.1): The delay in (a) follows immediately from the theorem. Whenever $|\alpha|$ (resp. $|\beta|$) is even, the bialternating requirement guarantees that $|\alpha^{(1)}| = |\alpha^{(2)}| = |\alpha|/2$ (resp. $|\beta^{(1)}| = |\beta^{(2)}| = |\beta|/2$). The recursive structure in (b) now follows immediately from the theorem.

The distinction between $\alpha^{(1)}$ and $\alpha^{(2)}$ is irrelevant, so we may as well suppose that $\alpha^{(1)}$ contains the first element of α. Given $\alpha^{(1)}$, the distinction between $\beta^{(1)}$ and $\beta^{(2)}$ is relevant, because Theorem 3.1(b) shows that this leads to different correction subnets and hence to different merging nets. This fact guarantees that the nets counted in this proof are distinct.

Let N_n be the number of minimum delay recursive merging nets for σ. When $n \geq 4$, the theorem tells us that we have minimum delay merging nets on $\sigma^{(1)}$ and $\sigma^{(2)}$ and that $|\alpha^{(i)}| = |\beta^{(i)}| = n/4$. It also tells us that there are three choices for the correction subnet. Hence $N_n = N_{n/2} \times N_{n/2} \times 3$ for $n \geq 4$. Note that $N_2 = 1$, since the net is a single comparator. It is easily verified that $N_n = 3^{n/2-1}$ is the solution to this recursion. If we disallow $c_{1:|\sigma|}$, "3" is replaced by "2" in the previous argument.

The minimum number of comparators, C_k, for the $|\alpha| = |\beta| = 2^{k-1}$ case is easily found. The theorem tells us that there is a correction subnet with $2^{k-1} \Leftrightarrow 1$ comparators but not with less. Hence $C_k = C_{k-1} + C_{k-1} + 2^{k-1} \Leftrightarrow 1$. Since $C_1 = 1$, one easily verifies that $C_k = (k \Leftrightarrow 1)2^{k-1} + 1$. To achieve this minimum, case (ii) in Theorem 3.1 can never occur. Hence $\sigma \vdash \{\alpha, \beta\}$ determines $\{\sigma^{(1)}, \sigma^{(2)}\}$. □

Proof (of Theorem 3.2): It suffices to prove the last statement in the theorem for a single comparator $c_{l:r}$ added to the jth layer. Relabeling α and β if necessary, we can assume that $l \in \alpha$. After the jth layer, Corollary 3.1.1(b) guarantees that position l will contain the minimum of some set μ' of input positions associated with one of the 2^{k-j} recursive merging nets of Corollary 3.1.1(b). Likewise, position r will contain the maximum of some set μ'' of such a set of input positions. Since the inputs α are sorted, it suffices to show that some element of $\alpha' = \mu' \cap \alpha$ is to the left of an element of $\alpha'' = \mu'' \cap \alpha$. (We may suppose $\alpha' \neq \alpha''$.) This follows from two observations:

(a) Since there are no optional comparators in the first layer, $|\alpha'| \geq 2$ and $|\alpha''| \geq 2$.

(b) If α' and α'' are the positions in α associated with two merging nets on the jth layer, then $\{\alpha', \alpha''\}$ is an alternating partition. We now prove this.

Let $k > j$ be the layer at which the inputs in positions μ' and μ'' first enter the same correction subnet. It suffices to look at the first k layers and induct on $k \Leftrightarrow j$. For $k \Leftrightarrow j = 1$, the result follows from Theorem 3.1(b).

We now suppose the result is true for $k \Leftrightarrow j = m$ and prove it for $k \Leftrightarrow j = m + 1$. In level $j + 1$, α' is associated with some correction subnet $\hat{\mu}'$. Let $\hat{\alpha}'$ be the subset of α associated with that correction subnet and define $\hat{\alpha}''$ similarly associated with a correction subnet $\hat{\mu}''$. By induction, $\{\hat{\alpha}', \hat{\alpha}''\}$ is alternating. By Theorem 3.1(b), $\hat{\alpha}' \vdash \{\alpha', \overline{\alpha}'\}$ and $\hat{\alpha}'' \vdash \{\alpha'', \overline{\alpha}''\}$ are alternating partitions. The result follows. □

7. Proof of Theorem 6.1

Note that Theorem 6.1 is not recursive—it deals only with the correction subnet after both $\sigma^{(1)}$ and $\sigma^{(2)}$ are sorted. Since (i) and (ii) have very similar proofs, we will prove only (i). We may assume that $1 \in \sigma^{(1)}$.

We make use of the 0-1 principle, which states that a net correctly merges or sorts if and only if it does so when its inputs are restricted to $\{0,1\}$. See [8, p. 224] for a proof. If τ is a set, let τ_i be the ith smallest element in τ and let $\tau(k)$ be the set of elements of τ which do not exceed k. Note that $|\tau(\tau_k)| = k$. When we say "τ contains m zeroes", we mean "the number of zeroes in a specified input that are in positions indexed by τ equals m". Note that, for (i) in the theorem,

$$\begin{aligned} \Big(|\alpha(k)| \text{ is odd}\Big) &\Leftrightarrow \Big(\max(\alpha(k)) \in \alpha^{(1)}\Big). \\ \Big(|\beta(k)| \text{ is odd}\Big) &\Leftrightarrow \Big(\max(\beta(k)) \in \beta^{(1)}\Big). \end{aligned} \tag{6}$$

We distinguish four cases based on the parity of the number of zeroes in α and in β.

Both Parities. We may assume that α contains $2i$ zeroes and β contains $2j+1$. It follows that $\sigma^{(1)}$ contains $i+j+1$ zeroes and $\sigma^{(2)}$ contains $i+j$. Let $k = 2i+2j+1$. Since $|\alpha(k)| + |\beta(k)| = k$, exactly one of $|\alpha(k)|$ and $|\beta(k)|$ is even, say $|\alpha(k)|$. Hence $|\alpha^{(1)}(k)| = |\alpha^{(2)}(k)|$ and $|\beta^{(1)}(k)| = |\beta^{(2)}(k)| + 1$. Thus $|\sigma^{(1)}(k)| = |\sigma^{(2)}(k)| + 1$ and so

$$\begin{aligned} |\sigma^{(1)}(k)| &= i + j + 1, \quad \text{the number of zeroes in } \sigma^{(1)}; \\ |\sigma^{(2)}(k)| &= i + j, \quad \text{the number of zeroes in } \sigma^{(2)}. \end{aligned}$$

It follows that, after sorting $\sigma^{(1)}$ and $\sigma^{(2)}$ we have an element of $0^k 1^*$.

Both Even. Let α contain $2i$ zeroes and β contain $2j$. Let $k = 2i+2j$. Note that $\sigma^{(1)}$ and $\sigma^{(2)}$ each contain $k/2$ zeroes. If $|\alpha(k)|$ and $|\beta(k)|$ are even, the reasoning is similar to the previous case. Otherwise, both $|\alpha(k)|$ and $|\beta(k)|$ are odd. Thus $|\sigma^{(1)}(k)|$ exceeds $|\sigma^{(2)}(k)|$ by 2; that is, $|\sigma^{(1)}(k)| = i+j+1$ and $|\sigma^{(2)}(k)| = i+j \Leftrightarrow 1$. By (6), $k \in \sigma^{(1)}$ and so $k+1 \in \sigma^{(2)}$. It follows that after sorting $\sigma^{(1)}$ and $\sigma^{(2)}$, we have an element of $0^{k-1}101^*$ and so $c_{k:k+1}$ is needed.

Both Odd. Let α contain $2i+1$ zeroes and β contain $2j \Leftrightarrow 1$. Let $k = 2i+2j$. Note that $\sigma^{(1)}$ contains $k+1$ zeroes and $\sigma^{(2)}$ contains $k \Leftrightarrow 1$. If $|\alpha(k)|$ and $|\beta(k)|$ are both odd, the reasoning is similar to the first case. Otherwise, both $|\alpha(k)|$ and $|\beta(k)|$ are even. Hence $|\sigma^{(1)}(k)| = |\sigma^{(2)}(k)| = k/2$ and, by (6), $k \in \sigma^{(2)}$ and $k+1 \in \sigma^{(1)}$ It follows that after sorting $\sigma^{(1)}$ and $\sigma^{(2)}$, we have an element of $0^{k-1}101^*$ and so $c_{k:k+1}$ is needed.

In both of the last two cases, k is even and hence any essential comparator must be of the form $c_{2l:2l+1}$. We now show that all of these are essential. If $|\alpha(2l)|$ and $|\beta(2l)|$ are even, let α and β each contain an odd number of zeroes for a total of $2l$. Use the "both odd" case to see that $c_{2l:2l+1}$ is essential. If $|\alpha(2l)|$ and $|\beta(2l)|$

are odd, let α and β each contain an even number of zeroes for a total of $2l$. Use the "both even" case to see that $c_{2l:2l+1}$ is essential.

We remark that for case (ii), comparators are needed in the "both parities" case and are not needed in the other two cases. □

8. Proof of Theorem 6.2

As in the proof of Theorem 6.1, we use the 0-1 principle to limit our attention to 0-1 inputs and note that Theorem 6.2 is not recursive.

To avoid excessive superscripting, we will associate a with $\alpha^{(1)}$, A with $\alpha^{(2)}$, b with $\beta^{(1)}$, and B with $\beta^{(2)}$. Furthermore, we usually drop subscripts and punctuation and abuse equality when referring to sequences so that we are dealing with words in $\mathcal{M} = \{a, b, A, B\}^*$. For example, $\sigma = Abab$ denotes the fact that $\sigma = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$, where $\sigma_i < \sigma_{i+1}$, has been partitioned into $\alpha = \{\sigma_1, \sigma_3\}$ and $\beta = \{\sigma_2, \sigma_4\}$. Furthermore, $\alpha^{(1)} = \{\sigma_3\}$, $\alpha^{(2)} = \{\sigma_2\}$, $\beta^{(1)} = \beta$, and $\beta^{(2)}$ is empty. Note that the $abAB$ notation conveys all the information needed to study one level of the recursive construction: The recursion will sort the lower case positions and will sort the upper case positions. The correction subnet will merge these two sorted sequences, using the fact that α (the sequence in $\{a, A\}^*$) and β (the sequence in $\{b, B\}^*$) were originally sorted.

- Let $\mathcal{M}_k$ be the set of all $x \in \mathcal{M}$ for which the least possible delay in the correction subnet is k.
- The notation
$$s \to x \to t \quad \text{with } x \in \mathcal{M} \text{ and } s, t \in \{0,1\}^*$$
means that, if s is sorted on α and β, then t is what we obtain *after* sorting on $\sigma^{(1)}$ and on $\sigma^{(2)}$, but *before* the correction subnet. Thus we think of x as describing a network that sorts lower case letter positions and sorts upper case letter positions, and we think of the arrows as indicating data flowing into and out of the network.

To prove Theorem 6.2, we require four lemmas.

Lemma 8.1. *If $x = x_1 \cdots x_n \in \mathcal{M}_k$, then the following are in $\mathcal{M}_k$ and conversely:*
(a) the word obtained from x by changing all α's to β's and conversely, preserving case;
(b) the word obtained from x by changing the case of all letters;
(c) the words obtained from x by allowing a and b to commute and allowing A and B to commute.
(d) the word $x_n \cdots x_1$ obtained by reversing x;

Proof: The operations in (a) and (b) do not affect the initial bimonotone requirements on the α's and on the β's, nor do they affect the monotone rearrangement of the upper case and lower case sequences. Interchanging an adjacent a,b pair leads to the same sequence after a monotone rearrangement on the lower case letters. Since

the same is true for upper case letters, (c) is proved. The reversal of all sequences together with the interchange of 0 and 1 leaves all statements valid and so (d) is proved. □

Lemma 8.2. *If $x, y, z \in \mathcal{M}$ and $xyz \in \mathcal{M}_1$, then $x, y, z \in \mathcal{M}_0 \cup \mathcal{M}_1$.*

We omit the simple proof of this lemma. The following lemma will be proved in the next section.

Lemma 8.3. *A correction subnet is minimal if removing any of the comparators destroys the merging property of the net. If $x \in \mathcal{M}_1$, then all comparators in a minimal, delay 1 correction subnet are of the form $c_{i:i+1}$; i.e., they are adjacent.*

Lemma 8.4. *The following sequences are not in $\mathcal{M}_1$:*
(a) elements of $b\{a, A\}^* Aa^* b$,
(b) elements of $A\{b, B\}^* b\{a, A\}^* b$.

Proof: For (a) we consider two functions. The first function is 0 everywhere except at the Aa^*; that is,

$$0\{0,0\}^*11^*0 \to b\{a, A\}^* Aa^* b \to 0\{0,0\}^*101^*.$$

Thus a comparator is needed between the rightmost A and the following position. The second function is

$$1\{0,0\}^*01^*1 \to b\{a, A\}^* Aa^* b \to \{0,0\}^*10^+1^*1,$$

where 0^+ corresponds to the rightmost block of A's. Thus a comparator is needed between the leftmost A in that block and the preceding position. This shows that $b\{a, A\}^* Aa^* b \notin \mathcal{M}_1$.

By (a), we can assume that (b) has the form $A\{b, B\}^* ba^* b$. By Lemmas 8.1(c) and 8.2, it suffices to show that $A\{b, B\}^* b^2 \notin \mathcal{M}_1$. Consider the function which is 1 at A and 0 elsewhere. Then

$$10^*0^2 \to A\{b, B\}^* b^2 \to 0^*10^*0^2,$$

where the final 1 is in the position of the rightmost capital letter. This sequence requires a nonadjacent comparator and so is not in $\mathcal{M}_1$ by Lemma 8.3. □

The rest of this section is devoted to proving the following strengthened version of Theorem 6.2.

Theorem 8.1. *We say the sequence x meets a set $\mathcal{S}$ k times if k elements of x lie in $\mathcal{S}$. Suppose $x \in \mathcal{M}$ meets each of the sets $\{a, A\}$, $\{b, B\}$, $\{a, b\}$, and $\{A, B\}$ at least twice (the "meet conditions") and is not bialternating, then $x \notin \mathcal{M}_1 \cup \mathcal{M}_0$.*

To see that this is stronger than Theorem 6.2, note that $x = aaBB$ satisfies the meet conditions but the partitions of α and β are both trivial.

Proof (of Theorem 8.1): We induct on $|x|$. Let $x = x_1 \cdots x_{n+1}$. By Lemma 8.1, we may assume that $x_{n+1} = b$. Let $x' = x_1 \cdots x_n$. The meet condition tells us that

$$x' \text{ contains at least two } \alpha\text{'s.} \tag{7}$$

By Lemma 8.2, and the induction hypothesis, we are done if x' satisfies the meet conditions and is not bialternating. Thus, it suffices to show that the following three situations are impossible when x is in $\mathcal{M}_1$, satisfies the meet conditions, and is not bialternating.

(i) x' is bialternating.
(ii) x' is not bialternating, contains b, and does not satisfy the meet conditions.
(iii) x' is not bialternating, does not contain b, and does not satisfy the meet conditions.

Case (i). Since x' is bialternating, (7) tells us that it contains both a and A. By Lemma 8.4(a), x must end in either bab or bb. By Lemma 8.1(c), we can replace bab with abb, which does not destroy the bialternating nature of x'. It follows from the fact that x' is bialternating and Lemma 8.4(b) that x ends with either $Cabb$ or $cBbb$ where c (resp. C) stands for some lower (resp. upper) case letter. Set all inputs to 0 except for the last A and the following a, if any. For $Cabb$, we have

$$0^*10^*100 \to \cdots Cabb \to 0^*1001,$$

which requires the nonadjacent comparator $c_{n-2:n}$. By Lemma 8.3, $x \notin \mathcal{M}_1$. For $cBbb$, we have

$$\left.\begin{matrix} 0^*10^*0^2 \\ 0^*10^*10^*0^2 \end{matrix}\right\} \to \cdots cBbb \to \begin{cases} 0^*100, & \text{if the rightmost } \alpha \text{ is } A; \\ 0^*101, & \text{if the rightmost } \alpha \text{ is } a; \end{cases}$$

The former requires the nonadjacent comparator $c_{n-1:n+1}$ and so is covered by Lemma 8.3. The latter requires $c_{n-1:n}$. In this case, set all inputs except c and the last bb to 0:

$$0^*1011 \to x \to 0^*1011,$$

which requires $c_{n-2:n-1}$. Since both $c_{n-2:n-1}$ and $c_{n-1:n}$ are required, the delay is at least two and so $x \notin \mathcal{M}_1$. This completes case (i).

Case (ii). There are three possible cases:

(a) x' contains A but not a.
(b) x' contains A and a.
(c) x' contains a but not A.

Suppose (a) holds. Since a is not present and x satisfies the meet conditions, it contains at least two A's. Set all A's to 1 and all β's to 0. After passing through the net x, the rightmost position will be 0 and there will be at least two 1's to the left of it. Apply Lemma 8.3.

Suppose (b) holds. Since x' does not satsify the meet conditions, it cannot contain B. By Lemma 8.4(a), it follows that no A can lie between two b's. Starting

at the rightmost A, we have $x = \cdots Aa^*ba^*b\cdots$. By Lemma 8.1(c), we can rearrange Aa^*ba^*b as Aba^*b, which is not in $\mathcal{M}_1$ by Lemma 8.4(b).

Suppose (c) holds. By the meet condition for x, x' contains a and B. If x' contained more than one a, we could apply Lemma 8.1(a) and Lemma 8.4 to conclude that $x' \notin \mathcal{M}_1$. Therefore x' contains at exactly one a. Since $x_n = b$, x meets $\{a, A\}$ only once, a contradiction.

Case (iii). Let $\pi(y)$ stand for some permutation of the letters of the word y. Since $b \notin x'$, $x = \pi(a^*A^*B^*)b$. The meet conditions on x show that this is in fact either $\pi(aa^+BB^+)b$ or $\pi(a^+A^+B^+)b$. The former cannot occur since then x' would satisfy the meet conditions. Thus it remains to consider $x = \pi(a^+A^+B^+)b$. Note that $x_1 \neq b$.

Consider the word $y = x_{n+1}\cdots x_1$. By Lemma 8.1(d) and the previous parts of the proof, especially the last two sentences in the previous paragraph, we may assume that y is one of

$$\pi(A^+b^+B^+)a, \quad \pi(a^+b^+B^+)A, \quad \pi(a^+A^+b^+)B.$$

Combining this with the conclusion of the previous paragraph, it follows that we may assume that x is one of

$$x^{(1)} = a\,\pi(A^+B^+)b, \quad x^{(2)} = A\,\pi(a^+B^+)b, \quad x^{(3)} = B\,\pi(a^+A^+)b.$$

Recall that x is not bialternating. By Lemma 8.1, we can assume that $x^{(1)}$ and $x^{(2)}$ contain at least two B's and that $x^{(3)}$ contains two adjacent a's. To eliminate $x^{(1)}$, apply Lemma 8.3 to

$$1\pi(1^+0^21^*)1 \to a\pi(A^+B^2B^*)b \to 10^21^*.$$

To eliminate $x^{(2)}$, apply Lemma 8.1(b), possibly Lemma 8.1(d), and lastly Lemma 8.4 to $\pi(a^+B^2B^*)$. To eliminate $x^{(3)}$, apply Lemmas 8.1(a) and 8.4(b). □

9. Proof of Lemma 8.3

Suppose that the correction subnet has delay 1 and contains a nonadjacent comparator. We will show that this leads to a contradiction. Among all inputs from $\{0,1\}^n$ that require a nonadjacent comparator let s be one containing the maximum number of ones. Let t be the result before the correction subnet, that is $s \to x \to t$. There are (0,1)-sequences p, q, and r with q nonempty such that $t = p1q0r$ is the result of passing s through all of the net except the correction subnet. The *leftmost* nonadjacent comparator that is needed, $c_{i:j}$, switches the 0 and 1 in $p1q0r$. Without loss of generality, we may assume that $i \in \sigma^{(1)}$ and $j \in \sigma^{(2)}$. We now prove a sequence of facts

(a) *We have* $p = 0^*$. If this were false, there would be a 1 in p that would require a nonadjacent comparator.

(b) *The last 0, if any, in a position indexed by α (resp. β) is in a position indexed by $\sigma^{(2)}$.* If this were false, we could change that 0 to a 1 without affecting t_i or t_j.

(c) *Any zeroes in t after the ith position must be indexed by $\sigma^{(2)}$.* Since $t_i = 1$ is indexed by $\sigma^{(1)}$ this follows from the fact that the positions indexed by $\sigma^{(i)}$ are sorted before the correction subnet.

(d) *We have $r = 1^*$.* Changing the last 0 in s to a 1 will not alter t_i and, if r contains a 0, will also not alter t_j by (c).

(e) *We have $q = 1^+$.* If we change the last 0 in s to a 1, it follows from (b) and (d) that t is changed to $p1q1r$. If q contained a 0, a comparator other than $c_{i:j}$ would be needed to move t_i. This contradicts the delay 1 assumption.

At this point we have shown that $t = 0^*11^+01^*$.

(f) *The first 1, if any, in a position indexed by α (resp. β) is in a position indexed by $\sigma^{(2)}$.* If not, change that 1 to a 0. Since $p = 0^*$, this changes t to $0^*01^+01^*$ and so a comparator other than $c_{i:j}$ is needed to move t_j, contradicting delay 1.

(g) *We may divide the total number of input ones between positions indexed by α and β in any manner without altering t.* By (b) and (f), increasing the number of ones by 1 in one subsequence and decreasing in by 1 in the other will not change the number of ones in $\sigma^{(1)}$ and $\sigma^{(2)}$. Hence t will be unchanged.

We are now ready to derive a contradiction. Let k be the total number of zeroes in s. Let l of the first k positions be indexed by α. From (g), we can place l zeroes in the positions indexed by α and the remaining in the positions indexed by β. Then $s = 0^k1^*$ and so it will not be rearranged by comparators. Hence $t = 0^k1^*$, contradicting the fact that $t = 0^*11^+00^*$. □

10. Proof of Theorems 4.1 and 4.2

Proof (of Theorem 4.2): Think of marking as meaning that the subnets being merged at that point must contain adjacent comparators. Consider the lines associated with a marked vertex. Since there are no comparators between $\sigma^{(1)}$ and $\sigma^{(2)}$ above the correction subnet, pairs of lines requiring comparators must belong to the same block of the partition. The agreement of label and type is precisely what is needed to guarantee that a pair of adjacent lines needing a comparator occur in the same block. This explains (a) in the theorem. Since the final correction layer contains only about half of the needed adjacent comparators, condition (b) starts the adjacent comparator requirement. Condition (c) insures that the requirement is propagated when the correction layer of a son does not contain the required adjacent comparators. □

Proof (of Theorem 4.1): The method of proof is essentially that same as that in [3] and [5].

Recall that we defined a *distance-t subsequence* of $s_1, s_2, \ldots$ to be a maximal subsequence whose successive indices differ by t. We call a sequence *t-sorted* if all its distance-t subsequences are sorted. We will show the following:

I For $0 \leq i \leq k \Leftrightarrow 2$, if α and β are 2^{i+1}-sorted, then, after passing through the first $k \Leftrightarrow i$ layers, they are 2^i-sorted.

II If the α and β sequences feeding into layer j are 2^i-sorted and $j > k \Leftrightarrow i$, then so are the outputs.

We now show how to complete the proof given I and II. Suppose α and β are 2^{i+1}-sorted. By I, they are each 2^i-sorted after passing through the first $k \Leftrightarrow i$ layers. By II, this property is preserved by passing through the remaining layers. Hence, if the input α and β sequences are 2^{i+1}-sorted, the output α and β sequences are 2^i-sorted. Since $|\alpha| = |\beta| = 2^{k-1}$, both sequences are trivially 2^{k-1}-sorted and so, by the previous sentence they are 2^0-sorted after $k \Leftrightarrow 1$ passes through the net. Since a 1-sorted sequence is sorted and the net is a merging net, one more pass completes the sort. This proves the theorem subject to verifying I and II.

The following observation will prove useful later. Let $\hat{\alpha}$ and $\hat{\beta}$ be any distance-t subsequences of α and β for any t. Together, they are a subsequence $\hat{\sigma}$ of σ. Since the hypothesis of the theorem states that $\sigma \vdash \{\alpha, \beta\}$ is alternating, it follows that

$$\hat{\sigma} \vdash \{\hat{\alpha}, \hat{\beta}\} \text{ is alternating.} \tag{8}$$

We now prove I. From Corollary 3.1.1(b), the first $k \Leftrightarrow i \Leftrightarrow 1$ layers consist of 2^{i+1} merging nets whose inputs are distance-2^{i+1} subsequences of α and β. Since α and β are assumed to be 2^{i+1}-sorted, each of the 2^{i+1} merging nets has as input a sorted subsequence of α and a sorted subsequence of β, and so the output of each of these merging nets is sorted. By Corollary 3.1.1(b) again, the $(k \Leftrightarrow i)$th layer consists of 2^i correction subnets. Limit attention to one of these correction subnets. Use $\{c, C\}$ and $\{d, D\}$ to stand for the $\{a, A\}$ and $\{b, B\}$ subsequences in some order, with upper (resp. lower) cases corresponding to upper (resp. lower) cases; i.e., letters correspond to letters and cases to cases. Note that:

(a) By the preceding discussion, the inputs to the correction subnet from cd are sorted as are the inputs from CD.

(b) By Corollary 3.1.1(b), c, C, d, and D are distance-2^{i+1} subsequences of α and β.

(c) By Corollary 3.1.1(b), the cC and dD sequences are distance-2^i subsequences of α and β.

Since the meaning of c and d and the assignment of upper and lower case can each be changed, it follows from (8), (b), and (c) that the only possible choices for the interleaving of the four subseqences are

$$\text{(i) } c_1, d_1, C_1, D_1, c_2, d_2, C_2, \ldots, D_l \text{ and (ii) } c_1, D_1, C_1, d_1, c_2, D_2, C_2, \ldots, d_l,$$

where $l = 2^{k-i-2}$. Note that (i) and (ii) of Theorem 3.1(c) apply to subsequences (i) and (ii), respectively.

We treat (i) and leave (ii) to the reader since it is similar. Let a sequence letter stand for the input to the $(k \Leftrightarrow i)$th layer and a primed letter stand for the output. By (a) above, $c_i \le d_i \le c_{i+1}$ and $C_i \le D_i \le C_{i+1}$. We have $C'_i = \max(d_i, C_i)$ and $c'_{i+1} = \max(D_i, c_{i+1})$. With or without the optional comparator, $c'_1 \le c_1$. Hence

$$\begin{aligned} c'_1 \le d_1 \le \max(d_1, C_1) = C'_1, \\ C'_i = \max(d_i, C_i) \le \max(D_i, c_{i+1}) = c'_{i+1}, \\ c'_i = \max(D_{i-1}, c_i) \le \max(d_i, C_i) = C'_i. \end{aligned}$$

Thus the cC sequence is sorted. Similarly, one obtains

$$\begin{aligned} D'_l \ge C_l \ge \min(d_l, C_l) = d'_l, \\ d'_i = \min(d_i, C_i) \le \min(D_i, c_{i+1}) = D'_i, \\ D'_{i-1} = \min(D_{i-1}, c_i) \le \min(d_i, C_i) = d'_i, \end{aligned}$$

which shows that the dD sequence is sorted. Since each of these are distance-2^i subsequences, we have proved I.

We now prove II. Consider that part of layer j corresponding to a particular correction subnet.

Assume that the correction subnet is as in Theorem 3.1(ii). The layer's input consists of distance-2^{k-j} subsequences $\hat{\alpha}$ and $\hat{\beta}$ of α and β. From (8), the partition $\hat{\sigma} \vdash \{\hat{\alpha}, \hat{\beta}\}$ is alternating. From Theorem 3.1(c), either all left inputs to the comparators are α's or all inputs to the comparators are β's. Now look at one of the distance-2^i subsequences of α that is a subequence of $\hat{\alpha}$. Call it $\tilde{\alpha}$. Note that

$$\tilde{\alpha} \text{ is a distance-} 2^{i-(k-j)} \text{ subsequence of } \hat{\alpha}. \tag{9}$$

Now look at the $\hat{\beta}$ subsequence $\tilde{\beta}$ that shares comparators with $\tilde{\alpha}$. It follows from (8) and (9) that $\tilde{\beta}$ is a distance-$2^{i-(k-j)}$ subsequence of $\hat{\beta}$ and hence a distance-2^i subsequence of β. By assumption, $\tilde{\alpha}$ and $\tilde{\beta}$ are sorted, and they are of the same length. Since the sequence formed by the minima (resp. maxima) of corresponding positions of sorted sequences is also sorted, we are done.

Now assume that Theorem 3.1(i) applies and use the notation and argument of the preceding paragraph. We are done except for dealing with the first and last elements of $\hat{\sigma}$, neither of which is the input of an adjacent comparator. We may assume that the first element of $\hat{\sigma}$ is in α. In this case, the right inputs of the adjacent comparators are always $\hat{\alpha}$'s. Suppose the first element of $\hat{\alpha}$ is in $\tilde{\alpha}$. Use primes to denote the ouput of the correction layer and lack of primes to denote its input. With or without the optional comparator, $\tilde{\alpha}'_1 \le \tilde{\alpha}_2$. Since $\tilde{\alpha}'_i = \max(\tilde{\beta}_{i-1}, \tilde{\alpha}_i)$ for $i > 1$, it follows that $\tilde{\alpha}'_1 \le \tilde{\alpha}'_2$. Hence $\tilde{\alpha}'$ is sorted. A similar argument applies to the distance-2^i β subsequence containing the last element of $\hat{\sigma}$. This completes the proof of II and hence of Theorem 4.1. □

References

[1] M. Aigner, Parallel complexity of sorting problems, *J. Algorithms* **3** (1982) 79–88. V. Grinberg has given an alternate proof. See Knuth, *The Art of Computer Programming*, Volume 3 errata, solution to Ex. 46 of Sec. 5.3.4 (text page 641). See `http://www-cs-faculty.stanford.edu/~knuth/taocp.html`.

[2] M. Ajtai, J. Komlos, and E. Szemeredi, Sorting in $c \log n$ parallel steps, *Combinatorica* **3** (1983) 1–19.

[3] E. R. Canfield and S. G. Williamson, A sequential sorting network analogous to the Batcher merge, *Linear and Multilinear Algebra* **29** (1991) 43–51.

[4] N. G. de Bruijn, Sorting by means of swapping, *Discrete. Math.* **9** (1974) 333-339.

[5] M. Dowd, Y. Perl, L. Rudolph, and M. Saks, The periodic balanced sorting network, *J. Assoc. Comput. Mach.* **36** (1989) 738–757.

[6] Z. Hong [H. Zhu] and R. Sedgewick, Notes on merging networks, *Proc. 14th Annual ACM Symposium on the Theory of Computing*, ACM, New York (1982) 296–302.

[7] T. E. Kammeyer, R. K. Belew, and S. G. Williamson, Evolving compare-exchange networks using grammars, *Artificial Life* **2** (1995) 199-237.

[8] D. E. Knuth, *Sorting and Searching*, volume 3 of *The Art of Computer Programming*, Addison-Wesley, Reading, MA (1973).

[9] P. B. Miltersen, M. Paterson, and J. Tarui, The asymptotic complexity of merging networks, *J. Assoc. Comput. Mach.* **43** (1996) 147–165.