A Lower Bound for Sorting Networks Based on the Shuffle Permutation

C. Greg Plaxton[*] *Torsten Suel*[†]

Department of Computer Science
University of Texas at Austin

Abstract

We prove an $\Omega(\lg^2 n/\lg\lg n)$ lower bound for the depth of sorting networks based on the shuffle permutation. This settles an open question posed by Knuth, up to a $\Theta(\lg\lg n)$ factor. The proof technique employed in the lower bound argument may be of separate interest.

1 Introduction

A variety of different classes of sorting networks have been described in the literature. Of particular interest here are the so-called AKS network [1] discovered by Ajtai, Komlós and Szemerédi, and the sorting network proposed by Batcher [2]. The AKS network is the only known sorting network with $O(\lg n)$ depth. However, the topology of the network is highly irregular, and the multiplicative constant hidden by the O-notation is impractically large [1, 8]. On the other hand, the network proposed by Batcher has a relatively simple interconnection structure and a small constant. This makes it the network of choice in many practical applications, although the network has depth $\Theta(\lg^2 n)$ and is thus asymptotically worse than AKS.

This situation has motivated attempts to construct $O(\lg n)$-depth sorting networks with simpler, more regular topologies, and/or a considerably smaller constant. For sorting networks based on Shellsort with monotonically decreasing increments, this question was answered in the negative by Cypher [3], who shows an $\Omega(\lg^2 n/\lg\lg n)$ lower bound for this class of networks. Another class of particular interest are small-depth sorting networks based on hypercubic networks (e.g., the hypercube, butterfly, cube-connected cycles, and shuffle-exchange). In this context, Cypher [4] has shown that the AKS network cannot be emulated on the cube-connected-cycles in $o(\lg^2 n)$ time.

This paper focuses on sorting networks based on the shuffle permutation, a notion that is formalized below. We will establish a lower bound of $\Omega(\lg^2 n/\lg\lg n)$ for any sorting network in this class. Before elaborating any further on this result, we will briefly describe the comparator network model and define the class of shuffle-based sorting networks.

Most commonly, a comparator network is defined as an acyclic circuit of comparator elements, each having two input wires and two output wires. One of the output wires is labeled as the *max-output*, which receives the larger of the two input values; the other output is called the *min-output*, and receives the smaller value. We will use this model of a comparator network throughout most of the paper, but will also briefly look at the following alternative model.

In this definition, a comparator network on n registers is determined by a sequence of pairs $(\Pi_i, \vec{x}_i)$, $0 \le i < d$, where Π_i is a permutation of $\{0,\ldots,n-1\}$ and $\vec{x}_i$ is a vector of length $\lfloor n/2 \rfloor$ over $\{+,-,0,1\}$. The network gets as input a permutation of $\{0,\ldots,n-1\}$ that is initially stored in the registers, and then operates on the input in d consec-

[*]Supported by NSF Research Initiation Award CCR–9111591, and Texas Advanced Research Program (TARP) Award #003658480.

[†]Supported by an MCD Fellowship of the University of Texas at Austin.

utive steps. In step i, $0 \le i < d$, the register contents are permuted according to Π_i, and then the operation stored in the kth component of $\vec{x}_i$ is applied to registers $2k$ and $2k+1$. For a "+" operation, the values stored in the two registers are compared, and the smaller of the values is stored in register $2k$, the larger one in $2k+1$. For a "−" operation, the values are stored in the opposite order. A "0" means that no operation takes place on the corresponding pair of registers. A "1" operation simply exchanges the values of the two registers. A comparator network is called a sorting network if it maps every possible input permutation to the same output permutation.

It is well known that the two models of comparator networks described above are equivalent. While the first model often appears more intuitive, we can use the second one to define some interesting special classes of networks by restricting the possible choices for the permutations Π_i.

For $n = 2^d$ where d is a nonnegative integer, the *shuffle permutation* π may be defined as follows. If $i_{d-1} \cdots i_0$ denotes the binary representation of some integer i, $0 \le i < 2^d$, then the shuffle of i has binary representation $i_{d-2} \cdots i_0 i_{d-1}$. We say that a network is based on the shuffle permutation if $\Pi_i = \pi$ holds for all i.

In this paper, we show an $\Omega(\lg^2 n / \lg\lg n)$ lower bound for the class of networks based on the shuffle permutation, thereby answering a question posed (as an open problem) by Knuth in Exercise 5.3.4.47 of [5], up to a $\Theta(\lg\lg n)$ factor. The best previously known lower bound for this class was the trivial $\Omega(\lg n)$ bound, while the best known upper bound continues to be given by Batcher's $\Theta(\lg^2 n)$-depth sorting network. In fact, our lower bound holds for a slighly more general class, the so-called delta networks (see, for example, [6]). Interestingly, the result cannot be extended to the case where both the shuffle permutation π and the unshuffle operation π^{-1} are allowed in the network. For that case, the "randomized" construction of Leighton and Plaxton [7] has been shown to imply the existence of sorting networks (in the usual sense) with nearly logarithmic depth [9].

Thus, one way of viewing the lower bound of this paper is that it establishes a non-trivial separation between the power of "ascend-descend" machines (e.g., the shuffle-exchange when both shuffling and unshuffling are permitted) and strict "ascend" machines (e.g., shuffle only). Of course, it must be emphasized that our lower bound for sorting on the directed shuffle-exchange graph only applies to sorting algorithms that correspond to shuffle-based sorting networks. On the other hand, this class of algorithms cannot be easily dismissed as being overly restrictive. After all, the primary motivation for considering hypercubic networks in the context of parallel computation is that they admit elegant and efficient strict ascend algorithms for a wide variety of basic operations (e.g., broadcast, parallel prefix, FFT).

The remainder of the paper is organized as follows. Section 2 contains an informal overview of the lower bound argument. In preparation for the actual proof, Section 3 states a number of useful definitions and basic lemmas. Section 4 contains the proof of the lower bound. Possible extensions of the lower bound are discussed in Section 5. Finally, Section 6 lists some open questions for future research.

2 An Overview of the Proof

A simple observation concerning comparator networks is that a sorting network has to make a comparison between all pairs of adjacent values in every input, i.e., any pair of values $\{m, m+1\}$ must appear on the input wires of some comparator element. Thus, one might attempt to prove a lower bound by maintaining a large set $\{m, \ldots, m+l\}$ of values such that no two elements of the set have been compared up to a given level of the network, for some appropriately chosen input. However, if we apply this idea to a network with "multidelta" topology (i.e., several delta networks concatenated together), starting with the set of all values, and, whenever two elements of the set get compared, removing one of them from the set, then we can lose up to half of the elements in any given level. So using this simple approach, we could only show a trivial lower bound of $\Omega(\lg n)$ for the depth of a sorting network.

The key idea to overcome this problem is to try to maintain a collection of $k \gg 1$ sets of "incomparable" elements (in our case, $k = \Theta(\lg^3 n)$). Because of the specific topology of our networks, in every level we recursively have two different collections of sets coming from two disjoint subnetworks. We will then show the existence of a partial matching between these two collections of sets such that, if we recombine the sets according to the matching and remove one element from every pair of elements that get compared, we only lose a small fraction of our elements in any level of the network.

With this idea, we can show that after passing through an entire delta network, the total number of elements in our collection of sets is still large, and the number of sets in our collection has not increased significantly. We can then iterate this result over consecutive delta networks, where after every delta network we pick the largest of the sets in our collection, partition it into k sets, and proceed with this new collection of sets.

To formalize this proof idea, we introduce the notion of an input pattern representing a class of similar inputs. We construct a class of inputs with the desired properties (existence of large sets of mutually uncompared, adjacent values) by stepwise "refinement" of a given input pattern in every level of the network.

3 Definitions

In the following, unless explicitely stated otherwise, the set of *input wires* of a comparator network will be denoted by W. An *input* to a comparator network is a total mapping from W to a set V of possible *input values*. We will restrict our attention to inputs π that are permutations of $\{0,\ldots,n-1\}$, i.e., $|W| = n$, $V = \{0,\ldots,n-1\}$, and π is 1-1. The set of all 1-1 functions from a set A to a set B will be denoted by $(A \mapsto B)$, and so the set of all inputs of a given comparator network may be written as $(W \mapsto V)$. Furthermore, for a function f on a set A and a subset B of A, let $f_{|_B}$ denote the functional restriction of f to B. For two functions f_0 and f_1 on disjoint sets A_0 and A_1, we write $f_0 \oplus f_1$ for the *union* of f_0 and f_1:

$$(f_0 \oplus f_1)(x) \stackrel{\text{def}}{=} \begin{cases} f_0(x) & \text{for all } x \text{ in } A_0, \text{ and} \\ f_1(x) & \text{for all } x \text{ in } A_1. \end{cases}$$

3.1 Input Patterns and Refinement

In the following definitions, we will introduce the notions of *input patterns* and *input pattern refinement*, which will be fundamental to understanding our proof technique. An input pattern specifies a set of inputs with certain common properties. Input pattern refinement is the process of imposing additional conditions on such a set of inputs.

Definition 3.1 Let P be a set and $<_P$ be a total ordering on P.

(a) An *input pattern* is a total mapping from W to P.

(b) Let p_0, p_1 be two input patterns. We say that p_0 can be refined to p_1 (written $p_0 \supset_W p_1$) if $(p_0(w) <_P p_0(w')) \Rightarrow (p_1(w) <_P p_1(w'))$ holds for all w and w' in W.

(c) Let p be an input pattern and π be an input. We say that p can be refined to π (written $p \supset_W \pi$) if $(p(w) <_P p(w')) \Rightarrow (\pi(w) < \pi(w'))$ holds for all w and w' in W.

The set P will be referred to as the *pattern alphabet*, and the elements of P are called *pattern symbols*. Throughout this paper, pattern symbols will be denoted by script letters.

Example 3.1 Let $P \stackrel{\text{def}}{=} \{\mathcal{S},\mathcal{M},\mathcal{L}\}$, $\mathcal{S} <_{\mathcal{P}} \mathcal{M} <_{\mathcal{P}} \mathcal{L}$, and $W \stackrel{\text{def}}{=} \{w_0,\ldots,w_6\}$. Then the input pattern p assigning $\mathcal{L}$ to w_0 and w_1 and $\mathcal{M}$ to all other wires can be refined to all inputs that assign the two largest values (in this case, $\{5,6\}$) to w_0 and w_1. We could also refine p to other input patterns, for example to a pattern p' such that $\mathcal{L}$ is assigned to w_0 and w_1, $\mathcal{S}$ is assigned to w_2 and $\mathcal{M}$ is assigned to all other wires. The new pattern p' can itself be refined to all inputs that assign the largest values to w_0 and w_1, and the smallest value (0 in this example) to w_2.

The relation $\supset_W$ defined above is a partial ordering on the set of input patterns. Note that the set V of input values can be regarded as a special case of a pattern alphabet with the ordering of the natural numbers. Every pattern can be refined to some input, and we could assume that the pattern alphabet P is always a subset of V. The pattern-to-pattern refinement in Part (b) of Definition 3.1 would then become a special case of the pattern-to-input refinement in Part (c). However, in the following we will not restrict our choice of P to subsets of V. We will see that this gives us more power of expression and, thus, simplifies the proof of our lower bound.

We usually think of an input pattern p as a description of the set of inputs that p can be refined to. This set will be denoted by $p[V] \stackrel{\text{def}}{=} \{\pi \mid \pi \text{ is an input such that } p \supset_W \pi\}$. When we refine a pattern p_0 to p_1 then we are imposing additional constraints on this set of inputs. Formally, we have $(p_0 \supset_W p_1) \Leftrightarrow (p_0[V] \supseteq p_1[V])$.

Alternatively, the reader may also view an input pattern p as a shorthand for a logical predicate on the set W that holds for exactly the inputs in $p[V]$.

Definition 3.2 Let p and q be input patterns on W and let U be a subset of W.

(a) The input pattern $p_{|_U}$ on U is the *restriction* of p to U.

(b) We say that p can be U-refined to q (written $p \supset_U q$) if $p \supset_W q$ and $p(w) = q(w)$ holds for all w in $W \setminus U$.

Definition 3.3 Let U_0 and U_1 be disjoint subsets of W, let p_0 be an input pattern on U_0, and let p_1 be an input pattern on U_1. Then $q = p_0 \oplus p_1$ is the input pattern on $U_0 \cup U_1$ such that $q_{|U_0} = p_0$ and $q_{|U_1} = p_1$.

If for two patterns p_0 and p_1 both $p_0 \supset_W p_1$ and $p_1 \supset_W p_0$ hold, then we say that p_0 and p_1 are *equivalent*. In this case, we have $p_0[V] = p_1[V]$, and the refinement steps from p_0 to p_1 and vice versa can be achieved by simply renaming the pattern symbols in a way that preserves the ordering $<_P$. Hence, we call this special case of a refinement step an *order-preserving renaming*.

Example 3.2 Let $W \stackrel{\text{def}}{=} \{w_0, \ldots, w_{n-1}\}$ and $P \stackrel{\text{def}}{=} \{\mathcal{P}_i \mid i \geq 0\}$ with $\mathcal{P}_i <_P \mathcal{P}_{i+1}$ for all $i \geq 0$. Then any input pattern p is equivalent to the input pattern p_k, $k \geq 0$ obtained from p by substituting all pattern symbols $\mathcal{P}_i$ in p by $\mathcal{P}_{i+k}$, for all i.

3.2 Comparator Networks

We will now formalize our notion of a comparator network, and explain how its domain of operation can be extended from the set of inputs to the set of input patterns.

A network is a mapping from a set of possible inputs to a set of possible outputs. In particular, a comparator network Λ on input wires W and output wires W' is a mapping from $(W \mapsto V)$ to $(W' \mapsto V)$ such that every input $\pi : W \mapsto V$ is mapped to an output $\pi' : W' \mapsto V$ that is a permutation of π.

Let Λ_0^*, Λ_1^* be two sets of n-input comparator networks. Then $\Lambda_0^* \otimes \Lambda_1^*$, the *serial composition* of Λ_0^* and Λ_1^*, denotes the set of all networks Λ that can be obtained by connecting the output wires of a network from Λ_0^* to the input wires of a network from Λ_1^*. We place no restrictions on this mapping from output wires to input wires, except that it be one-to-one. As it happens, we will often make use of the serial composition operator in the context of singleton sets Λ_0^* and Λ_1^*. In such a case, we may write, for example, $\Lambda_0 \otimes \Lambda_1$ (where Λ_0, Λ_1 are networks) rather than $\{\Lambda_0\} \otimes \{\Lambda_1\}$.

Given two comparator networks Λ_0 and Λ_1 on disjoint sets of input and output wires, we obtain the *parallel composition* of Λ_0 and Λ_1 as the union of the two networks, written $\Lambda_0 \oplus \Lambda_1$. The set of input (output) wires of $\Lambda_0 \oplus \Lambda_1$ is the union of the sets of input (output) wires of Λ_0 and Λ_1.

Definition 3.4 A 2^l-input comparator network Δ is called an l-level *delta network* if

- $l = 0$ and Δ contains no comparator elements, or
- $l > 0$ and Δ is an element of $(\Delta_0 \oplus \Delta_1) \otimes \Gamma_l$, where
  - (i) Δ_0 and Δ_1 are $(l-1)$-level delta networks, and
  - (ii) Γ_l consists of one level with at most 2^{l-1} comparator elements,

  and such that every comparator in Γ_l takes one input from an output wire of Δ_0 and the other input from an output wire of Δ_1.

Note that we do not require the ith level to have exactly 2^{i-1} comparator elements. This corresponds to allowing the delta network to contain "0" (do nothing) and "1" (exchange) circuit elements, as introduced in the "register model" of a comparator network.

We will call a network a (k, l)-multidelta network if it consists of k consecutive l-level delta networks, or, formally, if Δ belongs to $\Delta_0 \otimes \cdots \otimes \Delta_{k-1}$ where every Δ_i is an l-level delta network. It should be pointed out that this allows an arbitrary fixed permutation between any two consecutive delta networks, due to the definition of serial composition. It has been shown that any permutation can be routed by two consecutive forward butterfly networks [10]. Thus, eliminating the permutations between the delta networks would only increase the depth of the circuit by at most a constant factor.

A comparator network Λ was defined as a mapping from the set of inputs to the set of outputs. The following definition extends Λ to a function from the set of input patterns to the set of output patterns (an output pattern is a mapping from the set of output wires to the set of pattern symbols).

Definition 3.5 Given a comparator network Λ, an input pattern p_0, and an output pattern p_1, we define

$$\Lambda(p_0) = p_1 \Leftrightarrow \Lambda(p_0[V]) = p_1[V].$$

Note that this definition characterizes the behavior of a comparator network on an input pattern in the way that we would expect: If two pattern symbols $\mathcal{P}_0$ and $\mathcal{P}_1$ arrive on the input wires of a comparator gate, then the symbol that is larger according to the ordering $<_P$ will appear on the max-output of the gate, and the smaller one will appear on the min-output.

Definition 3.6 We say that two input wires w_0 and w_1 collide in a network Λ under an input π if the input values $\pi(w_0)$ and $\pi(w_1)$ are compared in Λ when π is given as input.

Note that according to the above definition, two values that meet in a noncomparator element, that is, a "0" (do nothing) or "1" (exchange) element, are not regarded as being compared. In the rest of the paper, we do not have to distinguish between the different circuit elements any more, since the entire lower bound argument is based on the notion of collision introduced above and extended in the following.

Given a network Λ and an input π we can always determine whether two input values are compared or not. This is not the case for input patterns, since an input pattern can contain several occurences of the same pattern symbol. This motivates the following definition of collision for input patterns:

Definition 3.7 Let Λ be a comparator network, p be an input pattern for Λ, and w_0 and w_1 be two input wires of Λ.

(a) We say that w_0 and w_1 *collide* in Λ under p if they collide in Λ under all inputs π with $p \supset_W \pi$.

(b) We say that w_0 and w_1 *can collide* in Λ under p if there exists an input π with $p \supset_W \pi$ such that w_0 and w_1 collide in Λ under π.

(c) We say that w_0 and w_1 *cannot collide* in Λ under p if there is no input π with $p \supset_W \pi$ such that w_0 and w_1 collide in Λ under π.

(d) A set $U \subset W$ is called *noncolliding* in Λ under p if any two wires in U cannot collide in Λ under p.

Obviously, if two wires collide (resp., cannot collide) in Λ under p, then they also collide (resp., cannot collide) in Λ under any refinement p' of p. Similarly, if a set U is noncolliding in Λ under p, then it is also noncolliding in Λ under p'. The property *can collide* is not preserved under arbitrary refinement.

In the following we restrict our attention to a fixed pattern alphabet P which will be used throughout the lower bound argument:

$$P \stackrel{\text{def}}{=} \{\mathcal{S}_i, \mathcal{X}_{i,j}, \mathcal{M}_i, \mathcal{L}_i \mid i, j \geq 0\}.$$

The ordering $<_P$ on P is defined by

$$\begin{aligned} \mathcal{S}_i &<_P \mathcal{S}_{i+1}, \\ \mathcal{S}_i &<_P \mathcal{X}_{0,0}, \\ \mathcal{X}_{i,j} &<_P \mathcal{X}_{i,j+1}, \\ \mathcal{X}_{i,j} &<_P \mathcal{M}_i, \\ \mathcal{M}_i &<_P \mathcal{X}_{i+1,0}, \\ \mathcal{M}_i &<_P \mathcal{L}_j, \text{ and} \\ \mathcal{L}_{i+1} &<_P \mathcal{L}_i, \end{aligned}$$

for all nonnegative integers i, j.

Finally, for a pattern p and a pattern symbol $\mathcal{P}$ we define the $[\mathcal{P}]$-set of p as the set $\{w \in W \mid p(w) = \mathcal{P}\}$. We can now formally express the idea of our lower bound argument: To prove that a network Λ is not a sorting network, we will show the existence of a pattern p such that its $[\mathcal{M}_0]$-set is noncolliding in Λ under p and contains at least two elements. We can then easily infer that Λ does not sort all inputs in $p[V]$. The pattern p will be constructed using stepwise refinement, starting with a pattern containing only the symbol $\mathcal{M}_0$.

3.3 Basic Lemmas

The following lemmas will be used in our lower bound argument. Their proofs are fairly straightforward and we will only sketch some of the proof ideas. Readers who are familiar with comparator networks, and who have followed the definitions and explanations up to this point, should be able to quickly convince themselves of the validity of these lemmas.

Lemma 3.1 Let p be an input pattern on W such that only the pattern symbols $\mathcal{S}_0$, $\mathcal{M}_0$, and $\mathcal{L}_0$ appear in p. Let W_0 and W_1 be disjoint subsets of W with $W = W_0 \cup W_1$ and let A be the $[\mathcal{M}_0]$-set of p. Let q_0 and q_1 be input patterns on W_0 and W_1, respectively, with $q_0(w)$, $q_1(w)$ not in $\{\mathcal{S}_0, \mathcal{L}_0\}$ for all w in A. Then from $p_{|W_0} \supset_{A \cap W_0} q_0$ and $p_{|W_1} \supset_{A \cap W_1} q_1$, we can infer $p \supset_A q_0 \oplus q_1$.

This lemma ensures that, given an input pattern p for a network $\Lambda = \Lambda_0 \oplus \Lambda_1$, we get a refinement of p if we separately refine the input patterns $p_{|W_0}$ for Λ_0 and $p_{|W_0}$ for Λ_1 in the way described, where W_0 and W_1 are the sets of input wires of Λ_0 and Λ_1, respectively.

Lemma 3.2 Let Δ be a d-level delta network, p be an input pattern for Δ, and $\mathcal{P}_0$ and $\mathcal{P}_1$ be pattern symbols in P. Let A_0 be the $[\mathcal{P}_0]$-set of p, and let A_1 be the $[\mathcal{P}_1]$-set of p. If A_0 and A_1 are each noncolliding in the first $d-1$ levels of Δ under p, then any two wires w_0 in A_0 and w_1 in A_1 either collide in level d under p, or they cannot collide in that level.

To prove the correctness of this lemma, note that an input value on a wire w in A_0 or A_1 will follow the same path through the first $d-1$ levels of the network under all inputs π with $p \supset_W \pi$. Hence, we can identify the locations of all input values in A_0 and A_1 in level d. This is also the underlying idea in the next lemma, which shows a 1-1 correspondence between the $\mathcal{M}_i$ on the input wires and those on the output wires of a network, provided that the $[\mathcal{M}_i]$-set is noncolliding.

Lemma 3.3 Let Λ be a comparator network in $\Lambda_0 \otimes \Lambda_1$, let i be a nonnegative integer, and let p be an input pattern for Λ_0 such that its $[\mathcal{M}_i]$-set A is noncolliding in Λ_0 under p. Let $q \stackrel{\text{def}}{=} \Lambda_0(p)$ be an input pattern for Λ_1 and let B be the $[\mathcal{M}_i]$-set of q. Then for every q' with $q \supset_B q'$ there exists a p' with $p \supset_A p'$ such that $q' = \Lambda_0(p')$. Furthermore, if the $[\mathcal{M}_i]$-set of q' is noncolliding in Λ_1 under q', then the $[\mathcal{M}_i]$-set of p' is noncolliding in Λ under p'.

To verify the validity of the final lemma, note that the paths taken by the $\mathcal{M}_i$-symbols through a network are not changed if we rename the rest of the symbols in the way described in the lemma.

Lemma 3.4 Let Λ be a comparator network, let p be an input pattern for Λ, and let A be the $[\mathcal{M}_i]$-set of p. Let $\rho_i(p)$ be the input pattern obtained from p by changing all pattern symbols $\mathcal{P}$ with $\mathcal{P} <_P \mathcal{M}_i$ to $\mathcal{S}_0$ and all pattern symbols $\mathcal{P}$ with $\mathcal{M}_i <_P \mathcal{P}$ to $\mathcal{L}_0$. If A is noncolliding in Λ under p, then A is also noncolliding in Λ under $\rho_i(p)$.

4 The Lower Bound

This section contains the proof of the lower bound for sorting on multidelta networks. The argument is divided into three major steps: First, a lemma will be established that shows the existence of a pattern with a "large" $[\mathcal{M}_0]$-set that is noncolliding in a single delta network. This is the main part of our proof, and also the one that employs the novel proof ideas. A subsequent theorem iterates the result of the lemma, thus showing how to maintain a "large" noncolliding $[\mathcal{M}_0]$-set over several stages of a multidelta network. Finally, a corollary establishes the lower bound.

Lemma 4.1 Let Δ be an l-level delta network, $l \geq 0$, and let p be an input pattern for Δ such that only the pattern symbols $\mathcal{S}_0$, $\mathcal{L}_0$, and $\mathcal{M}_0$ occur in p. Let A be the $[\mathcal{M}_0]$-set of p, and let k be a positive integer. Then there exists an input pattern q with $p \supset_A q$ and $t(l) \stackrel{\text{def}}{=} k^3 + lk^2$ sets $M_0, \ldots, M_{t(l)-1}$ of input wires such that the following properties hold:

(1) Every M_i is the $[\mathcal{M}_i]$-set of q,

(2) Every M_i is noncolliding in Δ under q,

(3) $B \subset A$, and

(4) $|B| \geq |A| - \frac{l \cdot |A|}{k^2}$,

where $B \stackrel{\text{def}}{=} \bigcup_{0 \leq i < t(l)} M_i$.

Proof: We will prove the lemma by induction over l, the number of levels in the delta network.

Base Case: $l = 0$

We define the sets $M_0, \ldots, M_{t(0)-1}$ by setting M_0 to A and all $M_i, 1 \leq i < t(0)$, to the empty set. If we set $q = p$, then Properties (1) to (4) are satisfied. In particular, Property (2) is satisfied since a 0-level delta network does not contain any comparators, and hence every set is noncolliding in the network under every pattern.

Induction Step: $l \to l+1$

An $(l+1)$-level delta network Δ consists of two l-level delta networks Δ_0 and Δ_1, and an $(l+1)$th level Γ_{l+1} satisfying the conditions of Definition 3.4. The input wires W of Δ can be partitioned into the sets W_0 and W_1 of input wires of Δ_0 and Δ_1, respectively. Let $p_0 \stackrel{\text{def}}{=} p_{|W_0}$ and $p_1 \stackrel{\text{def}}{=} p_{|W_1}$. Then $A_0 \stackrel{\text{def}}{=} A \cap W_0$ is the $[\mathcal{M}_0]$-set of p_0 and $A_1 \stackrel{\text{def}}{=} A \cap W_1$ is the $[\mathcal{M}_0]$-set of p_1.

Applying the induction hypothesis to Δ_0, p_0, and A_0 we can infer the existence of an input pattern q_0 with $p_0 \supset_{A_0} q_0$, and of $t(l)$ disjoint sets $M_{0,i}$, $0 \leq i < t(l)$, such that

- every $M_{0,i}$ is the $[\mathcal{M}_i]$-set of q_0,
- every $M_{0,i}$ is noncolliding in Δ_0 under q_0,
- $B_0 \subset A_0$, and
- $|B_0| \geq |A_0| - \frac{l \cdot |A_0|}{k^2}$,

where $B_0 \stackrel{\text{def}}{=} \bigcup_{0 \leq i < t(l)} M_{0,i}$.

Correspondingly, for Δ_1, p_1, and A_1 we get an input pattern q_1, disjoint sets $M_{1,i}$, $0 \leq i < t(l)$, and a set B_1, with the same properties.

We will now construct the sets M_i, $0 \leq i < t(l+1)$, by combining the sets $M_{0,i}$ of Δ_0 with the sets $M_{1,j}$ of Δ_1, according to some partial matching to be determined in the following.

Note that, due to the topology of a delta network, no element of a set $M_{0,i}$ can collide in Δ with any element of a set $M_{1,j}$ before level $l+1$. Also, because of Lemma 3.2, any two elements w_0 in $M_{0,i}$ and w_1 in $M_{1,j}$ either collide in level $(l+1)$ of Δ under $q_0 \oplus q_1$, or they cannot collide in that level.

For $0 \le i, j < t(l)$, we define $C_{i,j}$ as the set of all w_0 in $M_{0,i}$ such that w_0 collides with some w_1 in $M_{1,j}$ in level $l+1$ of Δ under $q_0 \oplus q_1$.

For $0 \le i < k^2$ and $0 \le j < t(l+1)$, we define

$$M(i,j) \stackrel{\text{def}}{=} \begin{cases} M_{0,j} & 0 \le j < i, \\ Z(i,j) & i \le j < t(l), \\ M_{1,j-i} & t(l) \le j < t(l)+i, \text{ and} \\ \emptyset & t(l)+i \le j < t(l+1), \end{cases}$$

where $Z(i,j) = (M_{0,j} \setminus C_{j,j-i}) \cup M_{1,j-i}$.

By their construction, the sets $M(i,j)$ are noncolliding in Δ under $q_0 \oplus q_1$. If we let $L_i \stackrel{\text{def}}{=} \bigcup_{i \le j < t(l)} C_{j,j-i}$, $0 \le i < k^2$, then:

$$\bigcup_{0 \le j < t(l+1)} M(i,j) = (B_0 \setminus L_i) \cup B_1.$$

The $C_{i,j}$ are pairwise disjoint and contained in B_0. Thus, the L_i's are also pairwise disjoint and contained in B_0. Hence, there must be an i_0, $0 \le i_0 < k^2$, such that $|L_{i_0}| \le \frac{|B_0|}{k^2}$. This i_0 determines the partial matching between the $M_{0,i}$ and the $M_{1,j}$.

We now define our sets M_j, $0 \le j < t(l+1)$ as $M_j \stackrel{\text{def}}{=} M(i_0, j)$ and show that this choice satisfies Properties (3) and (4). We have

$$\begin{aligned} B &\stackrel{\text{def}}{=} \bigcup_{0 \le j < t(l+1)} M_j \\ &= (B_0 \setminus L_{i_0}) \cup B_1 \\ &\subset B_0 \cup B_1 \subset A_0 \cup A_1 = A. \end{aligned}$$

This establishes Property (3). Verifying Property (4) is also straightforward:

$$\begin{aligned} |B| &= |B_0| + |B_1| - |L_{i_0}| \\ &\ge |A_0| - \frac{l \cdot |A_0|}{k^2} + |A_1| - \frac{l \cdot |A_1|}{k^2} - |L_{i_0}| \\ &= (|A_0| + |A_1|)\left(1 - \frac{l}{k^2}\right) - |L_{i_0}| \\ &\ge |A| - \frac{l \cdot |A|}{k^2} - \frac{|B_0|}{k^2} \\ &\ge |A| - \frac{(l+1) \cdot |A|}{k^2}. \end{aligned}$$

To complete our proof, we have to construct a refinement q of p such that Properties (1) and (2) hold for q and the sets M_j. We do this by A_0-refining q_0 to some q_0' and A_1-refining q_1 to some q_1'. Then $p_0 \supset_{A_0} q_0'$ and $p_1 \supset_{A_1} q_1'$, and by Lemma 3.1 the pattern $q \stackrel{\text{def}}{=} q_0' \oplus q_1'$ is an A-refinement of p.

We refine q_0 to q_0' in the following steps:

1. First change all pattern symbols $\mathcal{M}_i$ and $\mathcal{X}_{i,j}$ with $i \ge t(l)$ to $\mathcal{M}_{i+k^2}$ and $\mathcal{X}_{i+k^2,j}$, respectively.
2. Then change the pattern symbols of all wires in $C_{i,i-i_0}$ with $i_0 \le i < t(l)$ to $\mathcal{X}_{i,j_0}$, where j_0 is chosen such that before this step only symbols $\mathcal{X}_{i,j}$ with $j < j_0$ appear in the pattern.

The steps for the refinement of q_1 to q_1' are:

1′. First change all pattern symbols $\mathcal{M}_i$ and $\mathcal{X}_{i,j}$ with $i \ge t(l)+i_0$ to $\mathcal{M}_{i+k^2}$ and $\mathcal{X}_{i+k^2,j}$, respectively.

2′. Then change all pattern symbols $\mathcal{M}_i$ and $\mathcal{X}_{i,j}$ with $0 \le i < t(l)$ to $\mathcal{M}_{i+i_0}$ and $\mathcal{X}_{i+i_0,j}$, respectively.

All refinement steps described above are order-preserving renamings and, thus, valid refinement steps. Steps 1 and 1′ remove all symbols $\mathcal{M}_i$ and $\mathcal{X}_{i,j}$ with $t(l) \le i < t(l+1)$ from the pattern. Then Steps 2 and 2′ can be executed to perform the matching between the sets $M_{0,i}$ and $M_{1,j}$. Note that Steps 1 and 1′ are not really necessary since we can assume that the patterns q_0 and q_1 themselves have been constructed using the above refinement steps, and that, therefore, no symbols $\mathcal{M}_i$ and $\mathcal{X}_{i,j}$ with $t(l) \le i < t(l+1)$ exist in the pattern. However, we have chosen not to make this assumption a part of our induction hypothesis.

The pattern $q = q_0' \oplus q_1'$ has been constructed such that the sets M_i are the $[\mathcal{M}_i]$-sets of q, so Property (1) is satisfied.

To see that Property (2) holds, note that $C_{i,j}$, the set of input wires of $M_{0,i}$ that collide with an input wire of $M_{1,j}$ in Γ_{l+1} under $q_0 \oplus q_1$, also contains the same colliding wires with respect to $q = q_0' \oplus q_1'$. The sets $M_{0,i}$ are noncolliding in Δ_0 under q_0' and, thus, also noncolliding in Δ under q. By analogy, the sets $M_{1,j}$ are noncolliding in Δ under q. Hence,

$$M_j = (M_{0,j} \setminus C_{j,j-i_0}) \cup M_{1,j-i_0}$$

is noncolliding in Δ under q.

2

Theorem 4.1 Let Λ be a (d,l)-multidelta network with $d,l \geq 0$. Let W be the set of input wires of Λ, and let $n = |W| \geq 8$ be the number of input wires of Λ. Then there exists an input pattern p such that the following properties hold, where D is the $[\mathcal{M}_0]$-set of p:

(1) Only the symbols $\mathcal{S}_0$, $\mathcal{M}_0$, and $\mathcal{L}_0$ occur in p,

(2) D is noncolliding in Λ under p, and

(3) $|D| \geq \frac{n}{\lg^{4d} n}$.

Proof: We will prove the theorem by induction over d, the number of consecutive delta networks in Λ.

Induction Start: $d = 0$

Choose $D = W$ and p such that $p(w) = \mathcal{M}_0$ for all w in W.

Induction Step: $d \to d+1$

A $(d+1,l)$-multidelta network Λ consists of a (d,l)-multidelta network Λ_0 followed by a single delta network Δ, or, formally, $\Lambda \in \Lambda_0 \otimes \Delta$.

By the induction hypothesis there exists a pattern p' such that the following properties hold, where D' is the $[\mathcal{M}_0]$-set of p':

- Only the symbols $\mathcal{S}_0$, $\mathcal{M}_0$, and $\mathcal{L}_0$ occur in p',
- D' is noncolliding in Λ_0 under p', and
- $|D'| \geq \frac{n}{\lg^{4d} n}$.

Then the input pattern $q' \stackrel{\text{def}}{=} \Lambda_0(p')$ for Δ contains only the symbols $\mathcal{S}_0$, $\mathcal{M}_0$, and $\mathcal{L}_0$. The $[\mathcal{M}_0]$-set B' of q' has size $|B'| = |D'| \geq \frac{n}{\lg^{4d} n}$.

We can now apply Lemma 4.1 with Δ, q', and $l = k = \lg n$. By the lemma, there exists an input pattern q'' with $q' \supset_{B'} q''$ and $t(\lg n) = 2\lg^3 n$ disjoint sets $M_0, \ldots, M_{t(\lg n)-1}$ of input wires of Δ such that

- Every M_i is the $[\mathcal{M}_i]$-set of q'',
- Every M_i is noncolliding in Δ under q'',
- $B'' \subset B'$, and
- $|B''| \geq |B'| - \frac{\lg n \cdot |B'|}{\lg^2 n} \geq \frac{n}{\lg^{4d} n}(1 - \frac{1}{\lg n})$,

where $B'' \stackrel{\text{def}}{=} \bigcup_{0 \leq i < t(\lg n)} M_i$.

By averaging, there must be a set M_{i_0}, $0 \leq i_0 < 2\lg^3 n$, of size at least

$$\frac{n}{2\lg^{4d+3} n}\left(1 - \frac{1}{\lg n}\right) \geq \frac{n}{\lg^{4(d+1)} n},$$

where the last inequality follows from the fact that $\frac{1}{2}(1 - 1/\lg n) \geq 1/\lg n$ for all $n \geq 8$.

By Lemma 3.3, there exists an input pattern p'' for Λ with $p' \supset_{D'} p''$ such that $q'' = \Lambda_0(p'')$. The set M_{i_0} is noncolliding in Δ, hence the $[\mathcal{M}_{i_0}]$-set D of p'' is noncolliding in $\Lambda \in \Lambda_0 \otimes \Delta$ under p''.

Then, by Lemma 3.4, D is also noncolliding in Λ under $p''' \stackrel{\text{def}}{=} \rho_{i_0}(p'')$. By renaming in p''' all pattern symbols $\mathcal{M}_{i_0}$ to $\mathcal{M}_0$ we get a pattern p with $[\mathcal{M}_0]$-set D such that

- Only the symbols $\mathcal{S}_0$, $\mathcal{M}_0$, and $\mathcal{L}_0$ occur in p,
- D is noncolliding in Λ under p, and
- $|D| = |M_{i_0}| \geq \frac{n}{\lg^{4(d+1)} n}$.

This concludes the induction step.

2

Corollary 4.1.1 All n-input sorting networks with multidelta topology have depth $\Omega\left(\frac{\lg^2 n}{\lg\lg n}\right)$.

Proof: Let Λ be a $(d, \lg n)$-multidelta network with $d < \frac{\lg n}{4\lg\lg n}$. Then by Theorem 4.1 there exists an input pattern p and a set $D \subset W$ such that

- D is the $[\mathcal{M}_0]$-set of p,
- D is noncolliding in Λ under p, and
- $|D| \geq \frac{n}{\lg^{4d} n} > \frac{n}{\lg^{\left(\frac{\lg n}{\lg\lg n}\right)} n} = 1.$

Since $|D|$ is an integer, D must contain at least two elements, w_0 and w_1. Since $p(w_0) = p(w_1) = \mathcal{M}_0$, we can refine p to an input π such that the wires w_0 and w_1 have adjacent input values, that is, $\pi(w_0) = m$ and $\pi(w_1) = m+1$ for some integer m. Since D is noncolliding in Λ under p, the input values $\pi(w_0)$ and $\pi(w_1)$ never get compared in Λ under input π.

Let π' be the input obtained from π by exchanging the input values $\pi(w_0)$ and $\pi(w_1)$. Then the network Λ performs the same permutation on input π and on input π'. Hence, the network cannot sort both π and π'.

2

5 Extensions of the Result

This section discusses a few implications/extensions of our lower bound.

First, we point out that the proof of the lower bound still holds if the network is allowed to be "adaptive", in the following sense: If we write the network as a sequence of pairs $(\Pi_i, \vec{x}_i)$, then the labeling $\vec{x}_i$ of the ith level with elements from $\{+, -, 0, 1\}$ can depend on the outcome of all the comparisons made in all previous levels. Note that in our lower bound argument, it was never assumed that the labeling is fixed beforehand; instead, in every level, we allowed the "adversary" to choose the labeling in an arbitrary way. Hence, the validity of the argument is not affected by allowing the construction of the network to be adaptive.

Leighton and Plaxton [7] have designed a shuffle-unshuffle comparator network (i.e., a comparator network in which the permutation between successive levels can be either shuffle or unshuffle) of depth $O(\lg n)$ that sorts all but a superpolynomially small fraction of the inputs. Their technique can be applied to construct a shuffle-based network of depth $O(\lg n \lg\lg n)$ that sorts all but a polynomially small fraction of the inputs. This implies that we cannot hope to prove a lower bound close to $\Omega(\lg^2 n/\lg\lg n)$ for the "average case" complexity of shuffle-based networks, where the average case complexity is defined in the following manner. First, determine for every possible input the depth of the first level of the network at which the input "becomes sorted" (i.e., agrees with an appropriate fixed assignment of ranks given to the nodes at that level). Then define the average case complexity as the average of this depth over all inputs. It is not difficult to see that the $O(\lg n \lg\lg n)$ network mentioned above, followed by a bitonic sorter (say), achieves an average sorting depth of $O(\lg n \lg\lg n)$.

A similar argument can be used to show that we cannot hope to extend the lower bound to the "randomized" complexity of shuffle-based sorting networks. In order to construct a randomized shuffle-unshuffle network that sorts all inputs with high probability, Leighton and Plaxton [7] use an additional "randomizing" circuit element that interchanges the input values with probability 1/2, and leaves them unchanged otherwise. This new element can be used to construct a shuffle-based randomized sorter of depth $O(\lg n \lg\lg n)$.

For the class of shuffle-based networks, our lower bound also rules out a seemingly attractive way of designing small-depth sorting networks. The 0-1 principle says that every comparator circuit that sorts the 2^n inputs in $\{0,1\}^n$ is a sorting network. One might hope to strengthen this result by showing that there exists a relatively "small" subset of the 0-1 permutations such that every network that sorts these permutations would be "nearly" a sorting network. We formalize this idea by defining a *representative set* (with respect to the class of shuffle-based networks) as a subset of $\{0,1\}^n$ such that for any network Γ that sorts these permutations, there exists a shuffle-based network Λ of depth $o(\lg^2 n/\lg\lg n)$ such that $\Gamma \otimes \Lambda$ contains a true sorting network. Assuming the existence of a representative set of polynomial size, and using Corollary 4.2.1 of [9], the existence of an $O(\lg n \lg\lg n)$-depth shuffle-based network for sorting all but a polynomially small fraction of the inputs implies the existence of a $o(\lg^2 n/\lg\lg n)$-depth shuffle-based sorting network. Since this would contradict our lower bound, we can conclude that there does not exist a representative set of polynomial size.

This argument can be strengthened somewhat. The technique of Leighton and Plaxton [7] can also be used to construct a $o(\lg^2 n/\lg\lg n)$-depth shuffle-based sorting network that sorts all but an

$$\epsilon \stackrel{\text{def}}{=} 2^{-2^{o(\lg n/\lg\lg n)}}$$

fraction of the inputs. Arguing as in the preceding paragraph, there cannot exist a representative set of size less than $1/\epsilon$.

6 Concluding Remarks

In this paper, we have shown an $\Omega(\lg^2 n/\lg\lg n)$ lower bound on the depth of sorting networks based on the shuffle permutation. It would certainly be interesting to close the $\Theta(\lg\lg n)$ gap that remains between the upper and lower bounds.

Another possible direction for future research would be to consider other classes of networks based on restricted sets of permutations. For instance, it would be interesting to prove a $\omega(\lg n)$ lower bound for the class of sorting networks based on the shuffle and unshuffle permutations. Of course, we cannot discount the possibility that depth $O(\lg n)$ may be achievable within that class. Other interesting classes could be obtained by restricting the number of different permutations that can appear in the network. We might ask, for example, whether any small-depth sorting network exists that is based on a single permutation.

Finally, one might try to apply our proof technique of maintaining and recombining a large number of incomparable sets to other problems in connection with lower bounds.

References

[1] M. Ajtai, J. Komlós, and E. Szemerédi. Sorting in $c \log n$ parallel steps. *Combinatorica*, 3:1–19, 1983.

[2] K. E. Batcher. Sorting networks and their applications. In *Proceedings of the AFIPS Spring Joint Computer Conference,* vol. 32, pages 307–314, 1968.

[3] R. E. Cypher. A lower bound on the size of Shellsort sorting networks. *SIAM J. Comput.*, 22:62–71, 1993.

[4] R. E. Cypher. Theoretical aspects of VLSI pin limitations. *SIAM J. Comput.*, 22:58–63, 1993.

[5] D. E. Knuth. *The Art of Computer Programming*, volume 3. Addison-Wesley, Reading, MA, 1973.

[6] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, and Hypercubes.* Morgan-Kaufmann, San Mateo, CA, 1991.

[7] F. T. Leighton and C. G. Plaxton. A (fairly) simple circuit that (usually) sorts. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 264–274, October 1990.

[8] M. S. Paterson. Improved sorting networks with $O(\log N)$ depth. *Algorithmica*, 5:75–92, 1990.

[9] C. G. Plaxton. A hypercubic sorting network with nearly logarithmic depth. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 405–416, May 1992.

[10] F. Soviš. On rearrangeable networks of the shuffle-exchange type. *Computers and Artificial Intelligence*, 7:359–373, 1988.