Lower Bounds for Sorting Networks

Nabil Kahale [1,2] *Tom Leighton* [3] *Yuan Ma* [1,4]
C. Greg Plaxton [1,5] *Torsten Suel* [1,6] *Endre Szemerédi* [7]

Abstract

We establish a lower bound of $(1.12 - o(1))\, n \log n$ on the size of any n-input sorting network; this is the first lower bound that improves upon the trivial information-theoretic bound by more than a lower order term. We then extend the lower bound to comparator networks that approximately sort a certain fraction of all input permutations. We also prove a lower bound of $(c - o(1)) \log n$, where $c \approx 3.27$, on the depth of any sorting network; the best previous result of approximately $(2.41 - o(1)) \log n$ was established by Yao in 1980. Our result for size is based on a new technique that lower bounds the number of "0–1 collisions" in the network; we provide strong evidence that the technique will lead to even better lower bounds.

[1] Part of this work was done while the author was at DIMACS.

[2] XEROX Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304. Partially supported by the NSF under grant CCR-9404113. Email: `kahale@parc.xerox.com`.

[3] Department of Mathematics and Laboratory for Computer Science, MIT, Cambridge, MA 02139. Supported by ARPA Contracts N00014-91-J-1698 and N00014-92-J-1799. Email: `ftl@math.mit.edu`.

[4] Department of Computer Science, Stanford University, Stanford, CA 94305. Supported by an NSF Mathematical Sciences Postdoctoral Research Fellowship. Part of this work was done while the author was at MIT and supported by ARPA Contracts N00014-91-J-1698 and N00014-92-J-1799. Email: `yuan@cs.stanford.edu`.

[5] Department of Computer Science, University of Texas at Austin, Austin, TX 78712. Supported by the Texas Advanced Research Program under Grant No. ARP-93-003658-461. Email: `plaxton@cs.utexas.edu`.

[6] NEC Research Institute, 4 Independence Way, Princeton, NJ 08540. This work was done while the author was at the University of Texas at Austin, and was supported by the Texas Advanced Research Program under Grant No. ARP-93-003658-461. Email: `torsten@research.nj.nec.com`.

[7] Department of Computer Science, Rutgers University, Piscataway, NJ 08855. Supported by ARPA under contract DABT-63-93-C-0064. Part of this work was done at the University of Paderborn, Germany. Email: `szemered@cs.rutgers.edu`.

1 Introduction

The study of sorting networks has received considerable attention over the past several decades. In addition to providing a simple and elegant framework for many parallel and sequential sorting algorithms, sorting networks have also proved to be useful in several other applications, such as circuit switching and packet routing [10, 12, 19].

A *comparator network* is commonly defined as a leveled acyclic circuit of *comparators*, each having two input wires and two output wires. One of the output wires is labeled as the *max-output*, and receives the larger of the two input values; the other output is called the *min-output*, and receives the smaller value. We say that an n-input comparator network is a *sorting network* if it produces the same output sequence on all $n!$ permutations of $[n]$. (We use $[n]$ to denote the set $\{0, \ldots, n-1\}$.) The *depth* of a network is defined as the number of levels, and the *size* of a network is defined as the number of comparators in the network.

Thus, sorting networks can be seen as a simple model for oblivious (i.e., non-adaptive) sorting algorithms, where the depth and size of the network correspond to the parallel running time and the amount of hardware, respectively, that are needed to implement the algorithm. A fundamental problem in the study of sorting networks is the construction of networks that achieve a small size and/or depth. We refer the readers to [10, 19] for excellent surveys of the history of this problem.

1.1 Previous results

There is a trivial information-theoretic lower bound of $\log(n!) \approx n \log n - n \log e$ on the number of comparisons required by any comparison-based sorting algorithm. (We use "log" for the base 2 logarithm, and "ln" for the natural logarithm.) For the class of non-oblivious sorting algorithms, this lower bound can be easily matched, within a lower order term, by simple and well-known algorithms such as heapsort, merge sort, or binary-search-based insertion sort. However, the situation is quite different in the case of sorting networks.

Two elegant sorting networks of size $O(n \log^2 n)$ and depth $O(\log^2 n)$ were described by Batcher [5] in 1968. Following the work of Batcher, no improvements in the upper bounds were obtained for a number of years, and it remained uncertain whether a size of $O(n \log n)$ or a depth of $O(\log n)$ could be achieved by any sorting network. In fact, Knuth conjectured that sorting networks of size $O(n \log n)$ do not exist (see Exercise 51 on page 243 of [10]).

This conjecture was finally disproved in 1983 by Ajtai, Komlós, and Szemerédi [1], who described a sorting network with size $O(n \log n)$ and depth $O(\log n)$, commonly referred to as the "AKS network". However, even after several improvements in the construction of the network [2, 18], the constants hidden by the big-Oh notation remain impractically large.

Overall, the construction of sorting networks with small size and/or depth seems to be fundamentally more difficult than the design of fast non-oblivious sorting algorithms. A natural question to ask is whether there exists a lower bound for sorting networks that is significantly larger than the information-theoretic bound, thus formally establishing this seemingly apparent difference in complexity between sorting networks and algorithms. However, despite a number of attempts, not much progress has been made on this question so far.

In particular, Van Voorhis showed a simple lower bound of $n \log n$ in [22], and a more difficult lower bound of $n \log n + (n/2) \log \log n + \Omega(n)$ in [23]. While this is the best result known so far, it gives only a lower order improvement over the trivial information-theoretic bound of $n \log n - n \log e$. In related work, a number of bounds have been established for the size of comparator networks for merging and selection (see [3, 8, 9, 11, 16, 20, 21, 24, 25], and see Theorem F on page 230 of [10] for a result due to Floyd).

For the depth of sorting networks, Yao [24] has shown a lower bound of approximately $(2.41 - o(1)) \log n$. This should be compared with the $(2 - o(1)) \log n$ trivial information-theoretic bound. (Note that each level of a comparator network contains at most $n/2$ comparators.) In fact, Yao's lower bound was established for the depth of certain selection networks, but it also gives the best result currently known for the case of sorting. For small values of n, some computer-aided approaches for finding upper and lower bounds have been developed [6, 17].

Finally, significantly better upper bounds are known for comparator networks that sort all but a small fraction of the $n!$ input permutations. In particular, Leighton and Plaxton [15] have described a comparator network of size $c_1 n \lg n$ and depth $c_2 \log n$ that sort all but a superpolynomially small fraction of the $n!$ input permutations, where $c_1 \approx 3.22$ and $c_2 \approx 7.44$. Selection and merging networks that work for most input permutations have been studied in [7, 14].

1.2 Contributions of this paper

The main contributions of this paper are as follows:

1. We prove a lower bound of $(1.12 - o(1))\, n \log n$ for the size of any n-input sorting network. This is the first lower bound for size that improves upon the information-theoretic bound by more than a lower order term, and it demonstrates that significantly more comparisons are required for sorting in the network model than in the more general comparison-based algorithm model.

2. We extend essentially the same lower bound to the size of comparator networks that approximately sort a certain fraction of all input permutations.

3. We prove a lower bound of $(c - o(1)) \log n$ for the depth of any sorting network, where $c \approx 3.27$. This improves upon a lower bound of approximately $(2.41 - o(1)) \log n$ established by Yao [24] in 1980.

Our results for size are based on a new lower bound technique described in Corollary 2.1. Informally speaking, this technique reduces the problem of showing a lower bound on the size of a sorting network to that of showing a lower bound on the number of "0–1 collisions" (that is, the number of comparators that perform a comparison between a "0" and a "1" under some 0–1 input sequence).

The number of "0–1 collisions" is then lower-bounded by a potential function argument. The strength of the resulting lower bound depends on the choice of a good potential function. In this paper, we formally establish a lower bound of $(1.12 - o(1))\, n \log n$ for the size of any sorting network. We also provide strong evidence for the existence of better potential functions, and thus better lower bounds. In particular, we propose a potential function that we conjecture will lead to a $(c - o(1))\, n \log n$ lower bound with $c \approx 1.266$; this appears to be the best constant achievable under our potential function argument.

The remainder of this paper is organized as follows. In Section 2, we describe the reduction to the number of "0–1 collisions". Section 3 contains the potential function argument, and Section 4 describes an extension to the size of comparator networks that sort a certain fraction of all input permutations. In Section 5, we establish the lower bound for depth. Finally, Section 6 contains some concluding remarks.

2 A Size Lower Bound Based on Counting "Collisions"

For each $n \geq 1$, let $S(n)$ denote the minimum size of any n-input sorting network. In this section, we describe how to lower bound $S(n)$ by another quantity $C(n)$, which corresponds to the minimum number of "collisions" in any n-input sorting network. The main result of this section is Corollary 2.1, which will be used in the next section to establish a lower bound on $S(n)$.

In a certain sense, the technique in this section can be viewed as a highly nontrivial variant of an old technique introduced by Floyd [10, Theorem F on page 230] to lower bound the size of merging networks. (See [13] for another more sophisticated application of this technique.) Floyd's technique roughly proceeds as follows. Let $M(n)$ denote the minimum size of any n-input comparator network for merging two sorted lists of length $n/2$. Consider an arbitrary n-input merging network $\mathcal{M}$ of size $M(n)$. As argued in [10, Theorem F on page 230], we may assume without loss of generality that $\mathcal{M}$ is given in such a form that an input sequence to $\mathcal{M}$ may have up to $n/4$ items to be moved from $\mathcal{M}_0$ (the upper half of $\mathcal{M}$) to $\mathcal{M}_1$ (the lower half of $\mathcal{M}$), and vice versa. Thus, $\mathcal{M}$ needs to have at least $n/4$ comparators connecting $\mathcal{M}_0$ and $\mathcal{M}_1$. In addition, it is easy to argue that both $\mathcal{M}_0$ and $\mathcal{M}_1$ must be merging networks with $n/2$ inputs each. Thus, we have

$$M(n) \geq 2M(n/2) + n/4, \tag{1}$$

which solves to give $M(n) \geq (n/4)\log n$.

It would be nice if we could adapt Floyd's argument directly to obtain a lower bound for sorting networks of the form $(c - o(1))n \log n$ for some fixed constant $c > 1$. Unfortunately, however, direct application of Floyd's argument only yields a trivial lower bound of $(n/2)\log n$ since an input sequence to a sorting network may have up to $n/2$ items to be moved from the upper half to the lower half, and vice versa. Moreover, the triviality of such a lower bound is not due to any underestimate of the number of such comparators, since there do exist sorting networks (e.g., Batcher's bitonic sorting network [5]) that have exactly $n/2$ comparators connecting the upper and lower halves. Hence, Floyd's type of argument may not seem helpful at all in the context of proving nontrivial lower bounds for sorting networks.

The novelty of our technique is to partition a sorting network into two subnetworks in a "dynamic" fashion so that the number of comparators connecting the two subnetworks is at least $(c - o(1))n$ for some fixed $c > 1$. That is, the partition is not fixed in advance, but may depend on the structure of the sorting network.

2.1 Partitioning a sorting network relative to a given 0–1 input vector

Throughout this subsection, fix $n \geq 1$, let $\mathcal{N}$ denote a given n-input comparator network, and let ϕ denote a given 0–1 vector of length n. Let k denote the number of 0's in ϕ. Input vector ϕ induces a partition of the comparators of $\mathcal{N}$ into three classes: (i) the 0–0 comparators, i.e., those comparators for which both input wires contain a 0, (ii) the 1–1 comparators, i.e., those comparators for which both input wires contain a 1, and (iii) the 0–1 comparators, i.e., those comparators for which one input wire contains a 0 and the other input wire contains a 1. Let $a(\mathcal{N}, \phi)$ (resp., $b(\mathcal{N}, \phi)$, $c(\mathcal{N}, \phi)$) denote the number of 0–0 (resp., 1–1, 0–1) comparators in comparator network $\mathcal{N}$ on input vector ϕ. We also refer to each 0–1 comparator as a *collision*. Thus, $c(\mathcal{N}, \phi)$ is the number of collisions in comparator network $\mathcal{N}$ on input vector ϕ.

Lemma 2.1 *If $\mathcal{N}$ is a sorting network, then $a(\mathcal{N}, \phi) \geq S(k)$ and $b(\mathcal{N}, \phi) \geq S(n - k)$.*

Proof: We only prove that $a(\mathcal{N}, \phi) \geq S(k)$; an entirely symmetric argument can be used to establish the second inequality.

From sorting network $\mathcal{N}$ and 0–1 input vector ϕ, we construct a k-input comparator network $\mathcal{N}_0$ as follows: (i) remove all wires from $\mathcal{N}$ that receive a 1 under input ϕ, (ii) remove all 1–1 comparators from $\mathcal{N}$ (which are now isolated), and (iii) for each 0–1 comparator x in $\mathcal{N}$, remove x and connect the only remaining input of x directly to the lone remaining output of x. (It is straightforward to prove that $\mathcal{N}_0$ is indeed a k-input comparator network.) Note that there is a one-to-one correspondence between the set of comparators in $\mathcal{N}_0$ and the set of 0–0 comparators in $\mathcal{N}$. Hence, $\mathcal{N}_0$ is of size $a(\mathcal{N}, \phi)$.

Furthermore, the behavior of $\mathcal{N}_0$ on a given permutation of $[k]$ mimics that of the corresponding subnetwork of $\mathcal{N}$ when the same input permutation of $[k]$ is applied to the k "0-inputs" (i.e., input wires that receive a 0 under input ϕ) of $\mathcal{N}$, and the input value

k is passed to each of the remaining $n-k$ "1-inputs". (By "mimics" we mean that each comparator in $\mathcal{N}_0$ receives the same pair of input values as the corresponding comparator in $\mathcal{N}$. The preceding claim is straightforward to prove by induction on the number of levels in the network.) Since all such inputs are sorted by $\mathcal{N}$ (which is a sorting network), each of the $k!$ possible permutations of $[k]$ is mapped to the same output permutation by $\mathcal{N}_0$, i.e., $\mathcal{N}_0$ is a sorting network. We conclude that $a(\mathcal{N},\phi) \geq S(k)$, as required. ■

2.2 A lower bound recurrence for $S(n)$

Let $\mathcal{S}_n$ denote the set of all n-input sorting networks. Let

$$C(n) \;=\; \min_{\mathcal{N}\in\mathcal{S}_n} \max_{\phi\in\{0,1\}^n} c(\mathcal{N},\phi).$$

We now show how a lower bound on $C(n)$ can be used to obtain a lower bound on $S(n)$.

Lemma 2.2 $S(n) \geq \min_{0\leq i\leq n}(S(i)+S(n-i))+C(n)$.

Proof: Let $\mathcal{N}$ denote an n-input sorting network of size exactly $S(n)$. By the definition of $C(n)$, there exists a 0–1 input vector ϕ such that $c(\mathcal{N},\phi) \geq C(n)$. Let k denote the number of 0's in ϕ. By Lemma 2.1,

$$\begin{aligned} S(n) &= a(\mathcal{N},\phi)+b(\mathcal{N},\phi)+c(\mathcal{N},\phi) \\ &\geq S(k)+S(n-k)+C(n) \\ &\geq \min_{0\leq i\leq n}(S(i)+S(n-i))+C(n). \end{aligned}$$

■

Corollary 2.1 *If $C(n) \geq (c-o(1))n$, then $S(n) \geq (c-o(1))n\log n$.*

Proof: The proof is a straightforward induction on n, using Lemma 2.2 and the convexity of the function $x\log x$. ■

3 Lower Bounds for $C(n)$

This section is devoted to proving the following theorem.

Theorem 1 $C(n) \geq (1.12-o(1))n$.

By Corollary 2.1, we immediately obtain the first main result of the paper.

Theorem 2 $S(n) \geq (1.12-o(1))n\log n$.

3.1 The expected number of collisions in a sorting network

Direct analysis of the function $C(n)$ appears to be difficult. Our lower bound is obtained by considering instead $\overline{C}(n)$, which corresponds to the minimum expected number of collisions in any n-input sorting network when the input is a *random 0–1 vector*, that is, the input is a 0–1 vector drawn uniformly at random from $\{0,1\}^n$. Formally, we have

$$\overline{C}(n) \;=\; \min_{\mathcal{N}\in\mathcal{S}_n}\left(\sum_{\phi\in\{0,1\}^n} c(\mathcal{N},\phi)\right)/2^n\ .$$

By a straightforward averaging argument, we have

$$C(n) \;\geq\; \overline{C}(n). \tag{2}$$

As a result, any lower bound for $\overline{C}(n)$ is also a lower bound for $C(n)$. In Subsection 3.4, we prove that $\overline{C}(n) \geq (1.12-o(1))n$.

3.2 The class of "good" potential functions

We employ a potential function argument to obtain a lower bound on $\overline{C}(n)$. The choice of potential function is critical for obtaining the highest possible constant factor in our lower bound. In fact, we have identified an infinite sequence of potential functions, defined in Subsection 3.4, which we believe can be used to establish successively higher lower bounds on $\overline{C}(n)$. Our lower bound argument requires that the potential function be "good", as defined below:

Definition 3.1 *Let $\Phi(p)$ be a continuous real-valued function defined over the interval $[0,1]$. We say that $\Phi(p)$ is a* good *potential function if and only if the following conditions hold:*

(a) $\Phi(0)=\Phi(1)=0$,

(b) for $0\leq p\leq 1$, $0\leq q\leq 1$, and $pq\leq r\leq \min\{p,q\}$, we have

$$\Phi(p)+\Phi(q)-\Phi(r)-\Phi(p+q-r) \leq p+q-2r.$$

While we conjecture that every function in the infinite sequence alluded to above, and defined in Subsection 3.4, is a good potential function (see Subsection 3.5 for further discussion of this conjecture), we have only proved that the first two functions in the sequence are good. As a result, there is a significant gap between the best lower bound that we have been able to establish on $\overline{C}(n)$, namely $(1.12-o(1))n$, and the lower bound that would be established if we could prove that every potential function in the sequence is good, namely $(\gamma^*-o(1))n$, where $\gamma^*\approx 1.266$.

3.3 The potential function argument

In this subsection, we establish the following lower bound on $\overline{C}(n)$.

Lemma 3.1 *If Φ is a good potential function, then* $\overline{C}(n) \geq (\Phi(1/2) - o(1))n$.

We begin with some definitions. A function u from $\{0,1\}^n$ to $\{0,1\}$ is an *increasing function* if

$$\begin{aligned} & u(x_0, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_{n-1}) \\ \leq \; & u(x_0, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_{n-1}), \end{aligned}$$

for $0 \leq i < n$. The definition of a *decreasing function* from $\{0,1\}^n$ to $\{0,1\}$ is analogous. We state without proof the following classical result (see, e.g., [4, Chapter 6] for a proof).

Lemma 3.2 (The FKG inequality) *Let $X_0, \ldots, X_{n-1}$ be i.i.d. unbiased 0–1 random variables. If u and v are two decreasing functions from $\{0,1\}^n$ to $\{0,1\}$, then $u(X_0, \ldots, X_{n-1})$ and $v(X_0, \ldots, X_{n-1})$ are positively correlated, i.e.,* $\mathrm{E}[uv] \geq \mathrm{E}[u]\mathrm{E}[v]$.

Lemma 3.3 *Let $\mathcal{N}$ be an n-input comparator network, and let the input to $\mathcal{N}$ be a random 0–1 vector. Then for all pairs of output indices i and j, the event that the ith output receives a 0 and the event that the jth output receives a 0 are positively correlated.*

Proof: This follows from the FKG inequality. To see this, we need to check that the value of any output is an increasing function of the input sequence. This can be shown by induction on the number of comparators, since both min and max (the functions applied at each comparator) are increasing functions. ■

Let $\mathcal{N}$ be a comparator network and let x_i (resp., y_i) denote the probability that the ith input (resp., output) receives a 0 when the input to $\mathcal{N}$ is a random 0–1 vector. For any potential function Φ, define the *input potential* of $\mathcal{N}$ as $\Phi_{in}(\mathcal{N}) = \sum_{0 \leq i < n} \Phi(x_i)$. Similarly, define the *output potential* of $\mathcal{N}$ as $\Phi_{out}(\mathcal{N}) = \sum_{0 \leq i < n} \Phi(y_i)$.

Lemma 3.4 *Let $\mathcal{N}'$ be an n-input comparator network obtained by appending a comparator x to an n-input comparator network $\mathcal{N}$, and assume that the input to $\mathcal{N}'$ is a random 0–1 vector. Let Φ be a good potential function. Then the probability of a collision occurring at x is at least $\Phi_{out}(\mathcal{N}) - \Phi_{out}(\mathcal{N}')$.*

Proof: Let p (resp., q) be the probability that the first (resp., second) input to comparator x holds 0, and let s denote the probability that both inputs hold 0. The probability that the larger output of x is 0 is s, and the probability that the smaller output of x is 0 is $p+q-s$. By Lemma 3.3, $s \geq pq$. On the other hand, it is clear that $s \leq \min(p,q)$. Hence, Property (b) of Definition 3.1 implies that

$$\begin{aligned} & \Phi_{out}(\mathcal{N}) - \Phi_{out}(\mathcal{N}') \\ = \; & \Phi(p) + \Phi(q) - \Phi(s) - \Phi(p+q-s) \\ \leq \; & p + q - 2s. \end{aligned}$$

We conclude the proof by noting that the probability that a collision occurs in comparator x is $(p-s)+(q-s) = p+q-2s$. ■

Corollary 3.1 *Let $\mathcal{N}$ be any n-input comparator network, and assume that the input to $\mathcal{N}$ is a random 0–1 vector. Let Φ be a good potential function. Then the expected number of collisions occurring in $\mathcal{N}$ is at least $\Phi_{in}(\mathcal{N}) - \Phi_{out}(\mathcal{N}) = n \cdot \Phi(1/2) - \Phi_{out}(\mathcal{N})$.*

Proof: Straightforward by Lemma 3.4 and linearity of expectation. ■

Proof of Lemma 3.1: By Corollary 3.1, it is sufficient to prove that for any sorting network $\mathcal{N}$, and any good potential function Φ, we have

$$\Phi_{out}(\mathcal{N}) = o(n). \tag{3}$$

In order to establish Equation (3), we partition the outputs of $\mathcal{N}$ into three groups: (i) the $n/2 - \sqrt{n \ln n}$ lowest ranked outputs, (ii) the $n/2 - \sqrt{n \ln n}$ highest ranked outputs, and (iii) the remaining $2\sqrt{n \ln n}$ "middle" outputs. We now bound the contribution of each group of outputs to the sum defining $\Phi_{out}(\mathcal{N})$. By a standard Chernoff bound argument [4, Theorem A.1 on page 233], each of the outputs in group (i) has an associated probability of receiving a 0 that is greater than or equal to $1 - \exp(-2 \ln n) = 1 - 1/n^2 = 1 - o(1)$. (An output in group (i) can receive a 1 only if the total number of 1's in the random 0–1 input vector exceeds $n/2$ by at least $\sqrt{n \ln n}$.)

Since the potential function Φ is continuous and $\Phi(1) = 0$, the contribution of each output in group (i) is $\Phi(1 - o(1)) = o(1)$. Hence, the total contribution of the outputs in group (i) to $\Phi_{out}(\mathcal{N})$ is $o(n)$. A similar argument shows that the total contribution of the outputs in group (ii) to $\Phi_{out}(\mathcal{N})$ is $o(n)$. Finally, note that the contribution of any single output to $\Phi_{out}(\mathcal{N})$ is at most some constant independent of n. Hence, the total contribution of the outputs in group (iii) to $\Phi_{out}(\mathcal{N})$ is $O(\sqrt{n \log n}) = o(n)$. Together these arguments imply that Equation (3) holds, completing the proof of Lemma 3.1. ■

3.4 A good potential function

In this subsection, we first define an infinite sequence of potential functions g_d, $d \geq 1$. Then, we prove that g_2 is a good potential function. Finally, we use g_2 to prove the correctness of Theorem 1.

We first inductively define an infinite sequence of real functions f_d ($d \geq 1$) over the interval $[0, 1]$:

$$f_d(p) = \begin{cases} p - p^2 & \text{if } d = 1 \\ \frac{f_{d-1}(p^2)+f_{d-1}(2p-p^2)}{2} + p - p^2 & \text{if } d > 1. \end{cases}$$

In particular,

$$f_2(p) = 2p - 3p^2 + 2p^3 - p^4.$$

We next define a sequence of constants γ_d ($d \geq 1$) as

$$\begin{aligned} \gamma_d &= \frac{f_d(1/2)}{2(2f_d(1/2) - f_d(1/4) - f_d(3/4))} \\ &= \frac{f_d(1/2)}{4(f_d(1/2) - f_d(1/4))}, \end{aligned}$$

where the latter equality follows from the observation that $f_d(p) = f_d(1-p)$ for all $d \geq 1$ and $0 \leq p \leq 1$. For example,

$$\gamma_1 = 1 \text{ and } \gamma_2 = 28/25 = 1.12. \tag{4}$$

Given f_d and γ_d, potential function g_d is defined as

$$g_d(p) = \gamma_d f_d(p)/f_d(1/2). \tag{5}$$

It is straightforward to prove that g_1 is a good potential function. However a direct application of g_1 only yields a trivial lower bound of $(1 - o(1))n \log n$ for $S(n)$. To obtain a nontrivial lower bound, we need to prove that g_d is a good potential function for some $d \geq 2$. In fact, numerical experiments suggest that $g_d(1/2) \geq g_{d-1}(1/2)$ for all $d > 1$. Hence, we could hope to prove better and better lower bounds on $S(n)$ by proving that g_d is a good potential function for larger and larger values of d. In particular, we conjecture that all of the g_d's are good potential functions (see Subsection 3.5).

Lemma 3.5 *The potential function g_2 is good.*

Proof: The function g_2 is clearly continuous. Thus, it remains only to verify that Properties (a) and (b) of Definition 3.1 hold for g_2. Property (a) is immediate. Before addressing Property (b), observe that

$$\begin{aligned} f_2''(p) &= -6 + 12p - 12p^2 \\ &= -6(p^2 + (1-p)^2) \\ &\leq 0 \end{aligned}$$

for $0 \leq p \leq 1$. Thus, f_2 and g_2 are both concave.

To verify Property (b), we need to show that

$$\begin{aligned} & g_2(p) + g_2(q) - g_2(r) - g_2(p + q - r) \\ \leq \; & p + q - 2r \end{aligned} \tag{6}$$

holds for all p, q, and r such that $0 \leq p, q \leq 1$ and $pq \leq r \leq \min\{p, q\}$. Note that the difference between the right-hand side and the left-hand side of Equation (6) is a concave function in r, as its second derivative is equal to $g_2''(r) + g_2''(p+q-r) \leq 0$, since g_2 is concave. Thus, in order to show that Equation (6) holds for $pq \leq r \leq \min(p, q)$, it suffices to show that it holds for $r = pq$ and for $r = \min(p, q)$. The second case is straightforward, and so we are left to verify that the quantity

$$\begin{aligned} & p + q - 2pq - g_2(p) - g_2(q) \\ & + g_2(pq) + g_2(p + q - pq) \end{aligned} \tag{7}$$

is nonnegative. Letting $p = (u + v + 1)/2$ and $q = (u - v + 1)/2$, this quantity becomes

$$\begin{aligned} \frac{1}{50} \; & (3u^2 + 18u^4 - 20u^6 - u^8 + 69v^2 + \\ & 108u^2v^2 + 36u^4v^2 + 4u^6v^2 - 14v^4 - \\ & 12u^2v^4 - 6u^4v^4 - 4v^6 + 4u^2v^6 - v^8). \end{aligned}$$

Since $0 \leq p \leq 1$ and $0 \leq q \leq 1$, we have $|u| \leq 1$ and $|v| \leq 1$. Thus, we have $20u^6 + u^8 \leq 21u^4 \leq 3u^2 + 18u^4$ and $14v^4 + 12u^2v^4 + 6u^4v^4 + 4v^6 + v^8 \leq 37v^2$. This implies that the negative terms in the above expression are dominated by the positive terms. Hence, the above expression is nonnegative, as desired. ■

Proof of Theorem 1: By Equations (4) and (5), $g_2(1/2) = 1.12$. Hence, the correctness of the theorem follows from Equation (2) and Lemmas 3.1 and 3.5. ■

3.5 A seemingly valid and optimal potential function

We say that an n-input comparator network n^δ-approximately sorts all but an ϵ fraction of the $n!$ input permutations if there exists a fixed permutation $\pi : [n] \rightarrow [n]$ and a fixed set of n^δ output wires S such that, on at least $(1-\epsilon)n!$ of all input permutations, the rank of the input item that finishes in output i will be in the interval $[\pi(i)-n^\delta, \pi(i)+n^\delta]$ for all $i \notin S$, where δ is a constant less than 1. Because the output potential of an n-input comparator network that approximately sorts all but a polynomially small fraction of the $n!$ permutations is $o(n)$, our lower bound argument for $\overline{C}(n)$ can be adapted to show the following.

Lemma 3.6 *The expected number of collisions in an n-input comparator network that n^δ-approximately sorts all but a polynomially small fraction of the $n!$ input permutations (where δ is a constant less than 1) is at least $(1.12 - o(1))n$.*

The following result is proved in [15].

Theorem 3 (Leighton-Plaxton) *There exists a fixed permutation $\pi : [n] \to [n]$ and a fixed set of n^δ output positions S in an n-player butterfly tournament such that if the n players start in random positions, then with probability at least $1 - O(2^{-n^\epsilon})$ the rank of the player finishing in output i is in the interval $[\pi(i) - n^\delta, \pi(i) + n^\delta]$ for all $i \notin S$.*

On the other hand, a direct calculation shows that the expected number of collisions in an n-input butterfly when the input is a random 0–1 vector is $(\gamma^* - o(1))n$, where

$$\gamma^* \quad \approx \quad 1.266. \tag{8}$$

This leads us to ask whether the constant 1.12 in Lemma 3.6 can be improved. We conjecture that, indeed, Lemma 3.6 is valid if 1.12 is replaced by γ^*. We provide below evidence supporting this conjecture, although we do not yet know of a formal proof.

We observe that $f_d(p)$ is equal to the expected number of collisions in a butterfly of depth d, divided by the number of inputs, where the inputs are i.i.d. 0–1 random variables, each of which is equal to 0 with probability p. (In the following, we refer to this as the *expected number of collisions per input.*) This observation can be shown by induction on d. The case $d = 1$ follows easily. Assume now that the induction hypothesis holds for $d-1$. The expected number of collisions per input in the first level of the butterfly is $p - p^2$. The rest of the butterfly consists of two sub-butterflies of depth $d - 1$. The inputs to the first sub-butterfly are i.i.d. 0–1 random variables, each of which is equal to 0 with probability $2p - p^2$. The expected number of collisions per input in this sub-butterfly is equal to $f_{d-1}(2p - p^2)$. Similarly, the expected number of collisions per input in the second sub-butterfly is equal to $f_{d-1}(p^2)$. Thus, the expected number of collisions per input in the entire butterfly is

$$p - p^2 + \frac{f_{d-1}(p^2) + f_{d-1}(2p - p^2)}{2} = f_d(p).$$

The proof of Theorem 3, together with the above interpretation of $f_d(p)$, shows that $f_d(p)$ is uniformly bounded in d for all $p \in [0,1]$. For fixed $p \in [0,1]$, the sequence $f_d(p)$ is increasing and bounded, and so it converges to a real number $f(p)$. (It is possible to define the function g in a similar fashion, but it turns out that $g(p) = f(p)$ for all $p \in [0,1]$.) It can be shown that f is a continuous function of p. Note that f is the limit, as n goes to infinity, of the expected number of collisions per input in an n-input butterfly. It follows from the recursive definition of f_d that f satisfies the functional equation

$$f(p) = p - p^2 + \frac{f(p^2) + f(2p - p^2)}{2}.$$

We have written a computer program based on this functional equation that evaluates $f(p)$ to a high degree of precision for a large number of evenly-spaced values of p in $[0,1]$. We have also written a program to evaluate f_d for any given d. The function values generated by our programs lead us to the following conjecture.

Conjecture 1 *The functions g_d, for $d > 2$, and f are good potential functions.*

If Conjecture 1 holds, then the constant 1.12 occurring in our lower bounds can be improved to $f(1/2) = \gamma^* \approx 1.266$.

4 Lower Bounds for Networks Approximately Sorting Some Permutations

In this section, we extend our size lower bound for sorting networks to comparator networks that approximatedly sort a certain fraction of all input permutations. Throughout the section, we assume that Φ is a good potential function. For example, we can think of Φ as the function g_2 defined in Section 3.

Lemma 4.1 *Let $\mathcal{N}$ be an n-input comparator network such that for all but $\epsilon n!$ of the $n!$ input permutations, all but B of the n input items are output to a position that is within Δ of the correct position in a sorted order. Then, the expected number of collisions when a random 0–1 vector is input to $\mathcal{N}$ is at least*

$$\Phi\,(1/2)\,n - O\left(\epsilon n + 2B + 2\Delta + 2\sqrt{n \ln n} + n^{\frac{1}{3}}\right).$$

Proof Sketch: By Corollary 3.1, we need only show that

$$\Phi_{out}(\mathcal{N}) \leq O\left(\epsilon n + 2B + 2\Delta + 2\sqrt{n \ln n} + n^{\frac{1}{3}}\right).$$

This is done in a manner that is similar to the proof of Lemma 3.1. Specifically, we bound the probability that an output is incorrect, making use of the fact that most input permutations are mostly near-sorted. The full proof will appear in the final version of the paper. ■

Theorem 4 *For any $\epsilon > 0$, let $\mathcal{N}$ be an n-input comparator network such that for all but $\epsilon n!$ of the $n!$ input permutations, all but $B \leq n^\epsilon$ of the n input items are output to within $\Delta \leq n^\epsilon$ of the correct position in a sorted order. Then, $\mathcal{N}$ must have size at least*

$$\Phi(1/2)n \log n - O(\epsilon n \log n) - O(n \log \log n).$$

Proof Sketch: The proof is analogous to that of Theorem 2 except that we must use Lemma 4.1 instead of Lemma 3.1 and we need to be much more careful with the recursion. In particular, we will need to find an alternative argument to Lemma 2.2. Our approach will be to explicitly examine the behavior of the network on input sequences that have a restricted number of random 0–1 inputs as well as a specified number of $-\infty$ and $+\infty$ inputs. In this way, we will be able to obtain a better handle on the number of comparators that compare inputs in various regions.

In particular, we will manipulate the numbers and locations of $\pm\infty$ inputs to focus attention on how to derive lower bounds on the size of various subnetworks of the network. The lower bound is then derived by repeated applications of Lemma 4.1 on the subnetworks in combination with an averaging argument.

The full details will be presented in the final version of the paper. ■

Theorem 5 *Let $\mathcal{N}$ be an n-input comparator network that for at least $\delta n!$ of the $n!$ input permutations sorts all but n^ϵ of the input items to within n^ϵ of their correct position. Then, $\mathcal{N}$ must have size at least*

$$\Phi(1/2)n \log n - O(\epsilon n \log n) - O(n \log\log(1/\delta)) - O(n \log \log n).$$

In particular, if $\delta \geq e^{-n^\epsilon}$, then $\mathcal{N}$ must have size at least

$$\Phi(1/2)n \log n - O(\epsilon n \log n) - O(n \log \log n).$$

Proof Sketch: The proof is derived by examining the performance of $\mathcal{N}$ on various subsets of its inputs. An averaging argument is used to show that if $\mathcal{N}$ works for at least some input sequences of length n, then most of the various subnetworks of $\mathcal{N}$ work for most inputs. Then, Theorem 4 is applied to the subnetworks to obtain the lower bound. The full proof will appear in the final version of the paper. ■

5 A Lower Bound on Depth

In this section, we prove a lower bound of approximately $(3.27 - o(1)) \log n$ on the depth of any n-input sorting network. Our lower bound argument is an extension of the approach employed by Yao in [24]. More precisely, Yao established a lower bound of $(1/(2 - \log 3) - o(1)) \log n$ for the depth of certain selection networks; as a direct corollary, this also implies an identical lower bound for the depth of any sorting network. In the following, we show how to improve the lower bound in the case of sorting networks.

Theorem 6 *Any n-input sorting network has depth at least $(1/(1 - \log(a-1)) - o(1)) \log n \approx (3.27 - o(1)) \log n$, where $a = (3 + \sqrt{5})/2$.*

We point out that this lower bound only applies to networks that sort all input permutations. For networks that sort a certain fraction of all input permutations, a lower bound of $(2.24 - o(1)) \log n$ follows directly from Theorem 5. We are not aware of any previous results for this case.

Throughout this section, we assume that all comparator networks are given in Knuth's standard form [10], where the network consists of n horizontal lines numbered from 0 to $n-1$, and a set of comparators, such that every comparator outputs the larger of its two input values to the higher-numbered line. It is well known that every comparator network can be transformed into such a standard form, without increasing its size or depth.

We say that an n-input comparator network is an (n, k)-*selection network* if, on all input permutations, the network outputs the smallest k values on the first k output wires of the network, and the largest $n - k$ values on the last $n-k$ output wires. An n-input comparator network is called an (n, k, Δ)-*approximate-selection network* if, on all input permutations, the network outputs the smallest k values on the first $k + \lfloor\Delta\rfloor$ output wires, and the largest $n - k$ values on the last $n - k + \lfloor\Delta\rfloor$ output wires.

The next lemma is essentially due to Yao [24], who established a similar result for the case of

(n,t)-selection, whereas our lemma is for (n,t,t)-approximate-selection. For ease of notation, we use $T(n,t,t)$ to denote the smallest depth of any (n,t,t)-approximate-selection network.

Lemma 5.1 *For any $t \le n$,*

$$\begin{aligned} & 2t\lceil \log(t+1)\rceil & (9)\\ \ge\ & \frac{n}{2^{T(n,t,t)}} \sum_{i=0}^{\lfloor \log t\rfloor} \left(\lceil \log(t+1)\rceil - i\right)\binom{T(n,t,t)}{i}. \end{aligned}$$

Proof Sketch: The proof is almost identical to that of Theorem 4.1 in [24]. To prove Theorem 4.1 of [24], Yao assigned a weight to each segment of a wire in the network and argued that at most t output segments in an (n,t)-selection network could have weight less than or equal to $\lfloor \log t\rfloor$. (A wire of a comparator network is viewed as being partitioned into $d+1$ *segments*, where d denotes the depth of the network.) The only change needed in our proof is that an (n,t,t)-approximate-selection network may have up to $2t$ (as opposed to t) output segments with weight less than or equal to $\lfloor \log t\rfloor$. This is also why we have an extra factor of 2 on the left-hand side of Equation (9), compared with Equation (16) of [24]. ■

The next lemma is given as Exercise 25 on page 239 of Knuth's book [10] (see page 639 for a solution).

Lemma 5.2 *Let w be any wire of a comparator network. If w contains value i under input permutation π_i, and value $j > i$ under input permutation π_j, then for any k with $i \le k \le j$ there exists an input permutation π_k such that w contains value k under π_k.*

The next lemma is the key to extending Yao's depth lower bound for selection to a larger lower bound for sorting.

Lemma 5.3 *Given any sorting network $\mathcal{C}$ and any β with $0 \le \beta \le 1$, let $\mathcal{C}_\beta$ be the network obtained from $\mathcal{C}$ by removing all comparators located in the last $\lfloor \beta \log n\rfloor$ levels. Then $\mathcal{C}_\beta$ is an (n, n^β, n^β)-approximate-selection network.*

Proof: Assume for contradiction that $\mathcal{C}_\beta$ is not an (n, n^β, n^β)-approximate-selection network. Thus, there exist $i,j \in [n]$ with $|j-i| > n^\beta$, and an input permutation π such that j appears on output wire i of $\mathcal{C}_\beta$ under π. We assume $j > i$; the case $i > j$ is symmetric.

Since $\mathcal{C}_\beta$ is given in standard form, the value i will appear on output wire i under the identity permutation. By Lemma 5.2, this means that each of the $|j-i|+1 > n^\beta$ values between i and j appears on output wire i under some input permutation.

However, since every comparator has a fan-out of two, at most $2^{\lfloor \beta \log n\rfloor} \le n^\beta$ of the output wires of $\mathcal{C}$ are reachable from output wire i of $\mathcal{C}_\beta$. This implies that $\mathcal{C}$ cannot sort every input permutation, which contradicts our assumption that $\mathcal{C}$ is a sorting network. ■

Lemma 5.3 implies that the depth of any sorting network is at least $\beta \log n$ larger than the depth of an (n, n^β, n^β)-approximate-selection network.

Proof Sketch of Theorem 6: At a high level, we prove the claimed lower bound as follows. Let $t = n^\alpha$, where α is a constant to be chosen later. We first apply Lemma 5.1 to obtain a depth lower bound of $(A_\alpha \alpha - o(1))\log n$ for any (n, n^α, n^α)-approximate-selection network, where A_α is a constant depending on α). Then, by Lemma 5.3, we obtain a depth lower bound of $(A_\alpha\alpha + \alpha - o(1))\log n$ for any sorting network. Finally, we choose a particular α to maximize the value of $A_\alpha \alpha + \alpha$.

For $\alpha = 1/(3(2-\log 3)) \approx 0.80$, as computed by Yao [24], Lemma 5.1 implies a lower bound of $(3\alpha - o(1))\log n \approx (2.41 - o(1))\log n$ on the depth of any (n, n^α, n^α)-approximate-selection network. This, together with Lemma 5.3, immediately implies a lower bound of $(4\alpha - o(1))\log n \approx (3.21 - o(1))\log n$ on the depth of any sorting network. Such a choice of α indeed yields the best lower bound for selection networks (or (n,t,t)-approximate-selection networks) attainable from Lemma 5.1. For sorting networks, however, we need to choose a different α so that the lower bound of the form $((A_\alpha + 1)\alpha - o(1))\log n$ is maximized. The calculations are similar to those of Yao [24], and will be included in the full paper. ■

6 Concluding Remarks

We have established new lower bounds for the size and depth of sorting networks. Furthermore, we have extended our size lower bound to obtain essentially the same bound for comparator networks that only approximately sort a small fraction of the $n!$ input permutations. The size lower bound is obtained by counting the number of "0–1 collisions" in a comparator network. In particular, we have lower-bounded the number of 0–1 collisions by a potential function argument. The particular potential function we have exploited in the paper (i.e., g_2) leads to a lower bound of $(1.12 - o(1))\, n\log n$. Using the same potential function, we can also establish a constant factor slightly larger than 1.12 by: (i) showing that the expression of Equation (7) is 0 only at the point $(p, q) = (1/2, 1/2)$, and (ii) establishing an upper bound on the drop in potential at each comparator. Moreover, we have provided strong evidence that another potential function, the function f defined in Section 3.5, will yield a lower bound of $\gamma^* n \log n$, $\gamma^* \approx 1.266$, which appears to be the best bound achievable by any potential function in the context of counting the expected number of 0–1 collisions.

Since the potential function argument is based on lower bounds for $\overline{C}(n)$, which corresponds to the expected number of 0–1 collisions, we believe that a still better constant, larger than γ^*, is also possible by proving lower bounds for $C(n)$, which corresponds to the maximum number of 0–1 collisions.

Acknowledgements

We thank Andrew Yao for stimulating discussions on this research and helpful conversations on his work in [24], and we thank Friedhelm Meyer auf der Heide and Ran Raz for helpful comments.

References

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

[2] M. Ajtai, J. Komlós, and E. Szemerédi. Halvers and expanders. In *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 686–689, October 1992.

[3] V. E. Alekseyev. Sorting algorithms with minimum memory. *Kibernetika*, 5:99–103, 1969.

[4] N. Alon and J. H. Spencer. *The Probabilistic Method*. Wiley-Interscience, New York, NY, 1991.

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

[6] W. D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D*, 42:228–234, 1990.

[7] T. Ikeda. A probabilistic selection network with butterfly networks. In *ISAAC '93, Lecture Notes in Computer Science*, volume 762, pages 267–276, 1993.

[8] M. Jimbo and A. Maruoka. Selection networks with $8n \log n$ size and $O(\log n)$ depth. In *ISAAC '92, Lecture Notes in Computer Science*, volume 650, pages 165–174, 1992.

[9] N. Kahale. Eigenvalues and expansion of regular graphs. Technical Report 93-70R, DIMACS, Rutgers University, 1993.

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

[11] E. A. Lamagna. The complexity of monotone networks for certain bilinear forms, routing problems, sorting, and merging. *IEEE Trans. Comput.*, 28:773–782, 1979.

[12] T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, and Hypercubes.* Morgan-Kaufmann, San Mateo, CA, 1992.

[13] T. Leighton and Y. Ma. Tight bounds on the size of fault-tolerant merging and sorting networks with destructive faults. In *Proceedings of the 5th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 30–41, July 1993.

[14] T. Leighton, Y. Ma, and T. Suel. On probabilistic networks for selection, merging, and sorting. Manuscript, 1994.

[15] 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 458–469, October 1990.

[16] P. B. Miltersen, M. S. Paterson, and J. Tarui. The asymptotic complexity of merging networks. In *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 236–246, October 1992.

[17] I. Parberry. A computer-assisted optimal depth lower bound for nine-input sorting networks. *Mathematical Systems Theory*, 24:101–116, 1991.

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

[19] N. Pippenger. Communication networks. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, pages 805–833. North-Holland, Amsterdam, 1990.

[20] N. Pippenger. Selection networks. *SIAM J. Comput.*, 20:878–887, 1991.

[21] N. Pippenger and L. G. Valiant. Shifting graphs and their applications. *J. Assoc. Comput. Mach.*, 23:423–432, 1976.

[22] D. C. Van Voorhis. An improved lower bound for sorting networks. *IEEE Trans. Comput.*, 21:612–613, 1972.

[23] D. C. Van Voorhis. Toward a lower bound for sorting networks. In R. E. Miller and J. W. Thatcher, editors, *The Complexity of Computer Computations*, pages 119–129. Plenum Press, New York, NY, 1972.

[24] A. C. Yao. Bounds on selection networks. *SIAM J. Comput.*, 9:566–582, 1980.

[25] A. C. Yao and F. F. Yao. Lower bounds on merging networks. *J. Assoc. Comput. Mach.*, 23:423–432, 1976.