A Super-Logarithmic Lower Bound for Hypercubic Sorting Networks

C. Greg Plaxton[*] *Torsten Suel*[†]

Department of Computer Science
University of Texas at Austin

Abstract

Hypercubic sorting networks are a class of comparator networks whose structure maps efficiently to the hypercube and any of its bounded degree variants. Recently, n-input hypercubic sorting networks with depth $2^{O(\sqrt{\lg \lg n})} \lg n$ have been discovered. These networks are the only known sorting networks of depth $o(\lg^2 n)$ that are not based on expanders, and their existence raises the question of whether a depth of $O(\lg n)$ can be achieved by any hypercubic sorting network. In this paper, we resolve this question by establishing an $\Omega\left(\frac{\lg n \lg \lg n}{\lg \lg \lg n}\right)$ lower bound on the depth of any n-input hypercubic sorting network. Our lower bound can be extended to certain restricted classes of non-oblivious sorting algorithms on hypercubic machines.

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 networks proposed by Batcher [2]. While the AKS network is the only known sorting network with $O(\lg n)$ depth, it also suffers from two significant shortcomings. First, the multiplicative constant hidden by the O-notation is impractically large. Through a series of improvements [3, 14], this constant has been reduced to below 2000, but remains impractical. Second, the structure of the network is highly irregular, and does not seem to map efficiently to any of the common interconnection schemes. For example, Cypher [6] has shown that any emulation of the AKS network on the cube-connected cycles requires $\Omega(\lg^2 n)$ time. (A sorting algorithm emulates the AKS network if it performs the same sequence of comparisons on any input.)

In contrast, the networks proposed by Batcher have a relatively simple structure and a small associated constant, and can be efficiently implemented on many common interconnection schemes, including meshes and hypercubic networks. This makes them

[*]Email: `plaxton@cs.utexas.edu`.
 Supported by NSF Research Initiation Award CCR–9111591, and the Texas Advanced Research Program under Grant Nos. 003658–480 and 003658–461.

[†]Email: `torsten@cs.utexas.edu`.
 Supported by the Texas Advanced Research Program under Grant Nos. 003658–480 and 003658–461, and by a Schlumberger Graduate Fellowship.

the networks of choice in many practical applications, even though they have depth $\Theta(\lg^2 n)$ and are thus asymptotically inferior to AKS. This situation has motivated a number of attempts to construct $O(\lg n)$-depth sorting networks with simpler, more regular topologies, and/or a considerably smaller constant. Three classes of networks that have received particular attention are Shellsort networks [5, 10, 16, 19], periodic sorting networks [8, 9], and hypercubic sorting networks [13, 15].

In this paper, we focus on the class of *hypercubic* sorting networks, a notion that is formalized below. We establish a lower bound of $\Omega\left(\frac{\lg n \lg\lg n}{\lg\lg\lg n}\right)$ for the depth of any sorting network in this class. In fact, our lower bound argument can be extended to apply to certain restricted classes of non-oblivious sorting algorithms on hypercubic networks and multi-dimensional meshes. Before elaborating any further on these results, we will briefly describe the comparator network model, and define several classes of hypercubic networks.

1.1 Hypercubic Sorting Networks

A comparator network is most commonly 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 consider the following alternative model.

In this model, a comparator network on n registers is determined by a sequence of pairs $(\Pi_i, \vec{x}_i)$, $0 \le i < \ell$, 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 receives as input a permutation of $\{0, \ldots, n-1\}$ that is initially stored in the registers, and then operates on the input in ℓ consecutive steps. In step i, $0 \le i < \ell$, 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 $2k+1$. In 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 $2k+1$. In 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.

The *shuffle permutation* π_{sh} on $n = 2^d$ inputs may be defined as follows. If $j_{d-1} \cdots j_0$ denotes the binary representation of some integer j, $0 \le j < n$, then $\pi_{sh}(j)$ has binary representation $j_{d-2} \cdots j_0 j_{d-1}$. A sorting network is called *hypercubic* if $\Pi_i = \pi_{sh}$ or $\Pi_i = \pi_{sh}^{-1}$ holds for all i. A natural subclass of the hypercubic networks can be obtained by requiring $\Pi_i = \pi_{sh}$ for all i; we say that a network satisfying this condition is *shuffle-based*. Similarly, if $\Pi_i = \pi_{sh}^{-1}$ for all i, then the network is *unshuffle-based*.

The primary motivation for the definition of these two classes of networks is given by the fact that they can be efficiently implemented on any of the hypercubic interconnection networks (i.e., the hypercube, butterfly, cube-connected cycles, or shuffle-exchange). More precisely, the structure of the hypercubic sorting networks corresponds exactly to the class of *normal* algorithms on the hypercube, while the structures of the shuffle-based and unshuffle-based networks correspond to the classes of *descend* and *ascend* algorithms, respectively (see [11] for a definition of these classes). Most of the important algorithms that have been proposed for the hypercube are normal (e.g., Fast Fourier Transform, parallel prefix, bitonic merging and sorting). In fact, it can be argued that the primary motivation for the definition of the bounded-degree variants of the hypercube (i.e., the butterfly, cube-connected cycles, and shuffle-exchange) has been the capability of these networks to efficiently implement the class of normal algorithms.

The study of sorting networks based on the shuffle permutation was proposed by Knuth [10, Exercise 5.3.4.47]. The best upper bound for this class is given by Batcher's bitonic sort [2], with a depth of $O(\lg^2 n)$. A lower bound of $\Omega(\lg^2 n/\lg\lg n)$ was recently established by Plaxton and Suel [17]. However, this lower bound does not extend to arbitrary hypercubic networks.

The class of hypercubic sorting networks was defined by Leighton and Plaxton [13, 15], who show the existence of a family of hypercubic sorting networks with depth $2^{O(\sqrt{\lg\lg n})}\lg n$. The construction of these networks is based on a "probabilistic" sorting network described in [12], which sorts all but a superpolynomially small fraction of the possible input permutations. We point out that the depth of the above hypercubic networks is $o(\lg^{1+\epsilon} n)$, for all $\epsilon > 0$, and that they represent the only known sorting networks of depth $o(\lg^2 n)$ that are not based on expanders. Naturally, this raises the question of whether a depth of $O(\lg n)$ can be achieved by any hypercubic sorting network.

1.2 Overview of this Paper

In this paper, we resolve this question by showing a lower bound of $\Omega\left(\frac{\lg n \lg\lg n}{\lg\lg\lg n}\right)$ on the depth of any hypercubic sorting network. Our lower bound also extends to certain restricted classes of non-oblivious sorting algorithms on hypercubic machines and multi-dimensional meshes. However, our lower bound argument does not allow the copying of elements by the algorithm. Thus, the *Sharesort* sorting algorithm of Cypher and Plaxton [7], which achieves a running time of $O(\lg n \lg\lg n)$ (with preprocessing) on any of the hypercubic machines, is not subject to our lower bound. Nonetheless, we believe that our present results are already interesting in their own right, and that they may constitute an important step towards more general lower bounds for sorting on hypercubic machines. Due to space constraints, some of the proofs have been omitted from this abstract. A more detailed description of our results can be found in [18].

The remainder of the paper is organized as follows. Section 2 describes some of the basic ideas underlying our lower bound argument. Section 3 establishes a lower bound for a restricted class of hypercubic networks. Section 4 then shows our general lower bound. Some possible extensions and implications of our results are discussed in Section 5. Finally, Section 6 lists some open questions for future research.

2 Overview of the Proof

In this section, we give a very informal description of the most important ideas in the proof of our lower bound. To do so, we will first review the lower bound argument for shuffle-based networks given in [17], and explain why this relatively simple argument does not extend to the more general class of hypercubic sorting networks. We will then describe the new proof ideas that are needed in order to get a lower bound for arbitrary hypercubic sorting networks.

2.1 A Naive Proof Idea

A simple observation concerning comparator networks is that a sorting network must perform a comparison on every pair of adjacent values in every input, that is, every pair of values $\{m, m+1\}$ must appear on the input wires of some comparator element. (We assume the inputs to be permutations of $\{0,\ldots,n-1\}$.) Thus, one might attempt to prove a lower bound of ℓ for the depth of a class of comparator networks by showing, for all networks in the class, the existence of an input permutation π, and of a set of adjacent values $\{m,\ldots,m+i\}$ in π, such that no two elements of the set are compared up to level ℓ of the network. In the following, we will call such a set an *incomparable set*. If we apply this proof idea to a hypercubic network, starting out with the set of all values as our incomparable set, and, whenever two elements of the set get compared, removing one of them from the set, then we might lose up to half of the elements in any given level. So using this simple approach, we could only show the trivial lower bound of $\Omega(\lg n)$ for the depth of a sorting network.

2.2 The Proof for Shuffle-Based Sorting Networks

The key idea to overcome this problem is to modify the proof technique in a way that allows us to exploit the structural properties of the particular class of networks that we are studying. To explain this idea, we first consider the proof of the lower bound for shuffle-based sorting networks in [17]; the case of the unshuffle-based networks is symmetric. Note that a shuffle-based network can be seen as a concatenation of a number of butterfly networks of depth $\lg n$ each. Thus, if we can show that the size of our incomparable set decreases by at most a polylogarithmic factor in each butterfly, then at least $\Omega(\lg n/\lg\lg n)$ consecutive butterflies are needed in order to bring the size of the incomparable set down to 1; this directly implies the $\Omega(\lg^2 n/\lg\lg n)$ lower bound for shuffle-based sorting networks of [17].

The following recursive definition of a butterfly is crucial for understanding our proof technique: A butterfly with 2^d inputs and depth d consists of two parallel 2^{d-1}-input butterflies of depth $d-1$, followed by a final level of up to 2^{d-1} comparators. Every comparator in the final level takes one input from the outputs of each of the two 2^{d-1}-input subnetworks. Finally, a 1-input butterfly is just a wire. This "tournament-like" structure leads to the following important property of a butterfly: An observer of a 2^d-input butterfly tournament who sees the outcomes of all comparisons in the two 2^{d-1}-input subnetworks, but not the outcomes of the final level of comparisons, will not be able to say anything about the relative ordering of any two items taken from different subnetworks. In other words, the observer will not be able to say anything about the relative strength of the two "subtournaments" before the final stage. This "disjointness property" of the subnetworks plays a crucial role in the lower bound argument of [17].

Instead of maintaining only a single incomparable set, we now maintain a collection of incomparable sets in each recursive subnetwork. More precisely, after entering a new butterfly of depth $\lg n$, we partition our current incomparable set into $n\lg^3 n$ disjoint incomparable sets, most of which are empty, with $\lg^3 n$ sets entering on each wire. (Recall that a single wire is a 1-input butterfly.)

Due to the recursive structure of a butterfly, in every level we recursively have two different collections of $\Theta(\lg^3 n)$ incomparable sets coming from two disjoint subnetworks. In [17], it is shown that there exists a partial matching between these two collections of sets such that, if we combine the sets according to the matching and remove one element from every pair of elements from the same set that gets compared, we obtain a new collection of incomparable sets while losing only a very small fraction of our elements. The number of sets in this new collection is only slightly larger than the number of sets in either of the two previous collections. The aforementioned "disjointness property" of the two subnetworks is needed at this point to make sure that the new sets in the collection each contain adjacent elements.

If we repeat this process over all $\lg n$ levels of the butterfly, then we end up with a single collection of $\Theta(\lg^3 n)$ incomparable sets. The total number of elements in the sets is only a constant factor smaller than it was when we entered the butterfly. If we pick the largest of the $\Theta(\lg^3 n)$ sets as our new incomparable set, then we only lose a polylogarithmic factor in the size of the set.

To formalize this proof idea, the notion of an *input pattern* representing a class of similar inputs was introduced in [17]. A class of inputs with the desired property (existence of a large incomparable set) was then constructed by stepwise *refinement* of a given input pattern in every level of the network.

2.3 Hypercubic Sorting Networks

The above argument does not work for arbitrary hypercubic networks, as they do not satisfy the "disjointness property" of the two subnetworks used in the argument. In this paper, we overcome this obstacle, and derive a super-logarithmic lower bound for arbitrary hypercubic sorting networks. To do so, we introduce the class of hypercubic networks with "bounded overlap".

Assume we are given an arbitrary hypercubic network Λ with ℓ levels $(\Pi_i, \vec{x}_i)$, $0 \leq i < \ell$, as described in the register model of a comparator network. In order to define the "span" and "overlap" of Λ, it is convenient to introduce a number of auxiliary variables. Let $a_i = 1$ if $\Pi_i = \pi_{sh}$ and $a_i = -1$ if $\Pi_i = \pi_{sh}^{-1}$, $0 \leq i < \ell$. (We remark that the value of a_0 has no impact on the definitions that follow.) Let $b_i = \sum_{1 \leq j \leq i} a_j$, $0 \leq i < \ell$. The *span* of Λ may now be defined as $|\{b_i : 0 \leq i < \ell\}|$. The *overlap* of Λ is the minimum integer $r \geq 0$ such that either: (i) $b_i \leq b_j + r$ for all $0 \leq i < j < \ell$, or (ii) $b_i \geq b_j - r$ for all $0 \leq i < j < \ell$. Note that a network has overlap 0 iff $\Pi_i = \Pi_j$ for all $1 \leq i < j < \ell$. Furthermore, the span of a network is always at least as large as its overlap, with equality occurring only in the case $\ell = 0$, where the span and overlap are both 0.

The proof of the lower bound in this paper is based on two main new ideas. First, we show in Section 3 how the lower bound argument for shuffle-based networks can be modified to handle hypercubic networks with small overlap. The overall structure of this proof is very similar to that in [17]. However, a number of subtle changes are required in order to extend the argument to networks with non-zero overlap. The modified proof is based on the observation that, informally speaking, a shuffle-based network with small overlap still satisfies some relaxed version of the "disjointness property". More precisely, we will exhibit a trade-off between the overlap of the network and the lower bound that can be shown.

Second, we show in Section 4 that any hypercubic network can be partitioned into a number of consecutive hypercubic networks such that the overlap of each network in the partition is sufficiently smaller than its depth.

3 Hypercubic Networks with Small Overlap

In this section, we show that a large incomparable set can be effectively maintained over the levels of any hypercubic network with sufficiently small overlap. The main result of this section is Lemma 3.4, which bounds the decrease in the size of the incomparable set that can occur in any 2^d-input hypercubic network with span $s \leq d$ and overlap r. This lemma is used in Section 4 to establish our lower bound for arbitrary hypercubic sorting networks.

The actual argument addressing the size of the incomparable set is contained in the proof of Lemma 3.3, and is described with respect to a more general class of networks, called (d, s, r)-hypercubic networks, which properly contains the class of 2^d-input networks with span s and overlap r. The proof of Lemma 3.3 has a very similar structure to that of Lemma 4.1 in [17], and we only describe the necessary modifications. Most of the notations used in this section are taken from [17]. For the sake of completeness, we define these notations again in the following subsections.

The remainder of this section is organized as follows. In the first subsection, we introduce the concepts of *input patterns* and *input pattern refinement*. Subsection 3.2

defines our notion of a comparator network and its action on an input pattern, and introduces the class of (d, s, r)-hypercubic networks. Subsection 3.3 lists a few basic lemmas. Finally, Subsection 3.4 contains the proof of the main lemma, and a lower bound on the depth of hypercubic sorting networks with small overlap.

In the following, unless explicitly stated otherwise, the set of *input wires* of a comparator network is denoted 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., where $|W| = n$, $V = \{0, \ldots, n-1\}$, and π is one-to-one. The set of all one-to-one 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 introduce the notions of *input patterns* and *input pattern refinement*, which are fundamental to our proof technique. Informally, an input pattern describes 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 are denoted by script letters. An input pattern p may be viewed as a description of the set of inputs to which p can be refined. This set is denoted $p[V] \stackrel{\text{def}}{=} \{\pi : \pi \text{ is an input such that } p \supset_W \pi\}$. 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 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 $W \setminus U$.

3.2 Comparator Networks

A comparator network is interpreted as a mapping from a set of possible inputs to a set of possible outputs. More precisely, a comparator network Λ on input wires W and output wires W' defines a mapping (which we also denote by Λ) 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. By this we mean that there exists a bijection $\rho : W \mapsto W'$ such that $\pi(w) = \pi'(\rho(w))$ holds for all w in W.

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^*. In some cases, we may want to impose certain special conditions on this connection between the output wires of the first network and the input wires of the second network. If no conditions are stated, then the connections can be made according to an arbitrary one-to-one mapping. As it happens, we 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.

Below we give an inductive definition of a class of comparator networks, called (d, s, r)-*hypercubic* networks, which properly contains the class of 2^d-input hypercubic networks with span $s \leq d$ and overlap r. Note that the 2^d output wires of a (d, s, r)-hypercubic network are partitioned into 2^{d-r} *output groups* of size 2^r.

Definition 3.3 For $r \leq s \leq d$, a 2^d-input comparator network Δ is called a (d, s, r)-*hypercubic* network if:

(a) $s - r = 0$, Δ is a network containing no comparators at all (i.e., the 2^d input wires are directly connected to the 2^d output wires), and the output wires of Δ have been partitioned into 2^{d-r} output groups of size 2^r, or

(b) $s - r > 0$ and Δ is an element of $(\Delta_0 \oplus \Delta_1) \otimes \Lambda$, where

- Δ_0 and Δ_1 are $(d-1, s-1, r)$-hypercubic networks, and
- Λ is the parallel composition of 2^{d-r-1} disjoint 2^{r+1}-input comparator networks Λ_i, $0 \leq i < 2^{d-r-1}$, of arbitrary size and depth, such that: (i) the 2^{r+1} input wires of each network Λ_i are connected to one output group of size 2^r of Δ_0 and one output group of size 2^r of Δ_1, and (ii) the 2^{r+1} output wires of each network Λ_i are partitioned to form two of the 2^{d-r} output groups of network Δ.

A comparator network Λ was identified with a mapping from the set of inputs to the set of outputs. The following definition extends Λ to a mapping 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 3.4 Given a comparator network Λ, an input pattern p_0, and an output pattern p_1 such that $p_1(W) = p_0(W)$, we define $\Lambda(p_0) = p_1 \Leftrightarrow \Lambda(p_0[V]) = p_1[V]$.

Definition 3.5 We say that input wires w_0 and w_1 collide in a network Λ under input π if the input values $\pi(w_0)$ and $\pi(w_1)$ are compared in Λ when π is given as input.

Given a network Λ and an input π, we can always determine whether two input values are compared or not. (Recall that we only consider inputs that are permutations.) 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 3.6 Let Λ be a comparator network, let p be an input pattern for Λ, and let 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.

Note that, if two wires collide (cannot collide) in some network Λ under an input pattern p, then they also collide (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 remainder of this section, we restrict our attention to a fixed pattern alphabet $P \stackrel{\text{def}}{=} \{\mathcal{S}_i, \mathcal{X}_{i,j}, \mathcal{M}_i, \mathcal{L}_i : i, j \geq 0\}$. The ordering $<_P$ on P is defined by $\mathcal{S}_i <_P \mathcal{S}_{i+1}$, $\mathcal{S}_i <_P \mathcal{X}_{0,0}$, $\mathcal{X}_{i,j} <_P \mathcal{X}_{i,j+1}$, $\mathcal{X}_{i,j} <_P \mathcal{M}_i$, $\mathcal{M}_i <_P \mathcal{X}_{i+1,0}$, $\mathcal{M}_i <_P \mathcal{L}_j$, and $\mathcal{L}_{i+1} <_P \mathcal{L}_i$, for all nonnegative integers i, j.

Definition 3.7 For a pattern p and a pattern symbol $\mathcal{P}$ we define the $[\mathcal{P}]$-set of p as the set $\{w \in W : p(w) = \mathcal{P}\}$.

Definition 3.8 We say that a comparator network Λ has an *incomparable set* of size m if there exists an input pattern p such that some $[\mathcal{M}_i]$-set of p is of size m and is noncolliding in Λ under p.

We can now formally describe our proof strategy: To prove that a network Λ is not a sorting network, we will show that the network has an incomparable set of size at least 2. The input pattern p associated with the incomparable set can then be refined to an input such that the wires in the $[\mathcal{M}_0]$-set contain adjacent input values. This implies that Λ does not sort all inputs. The input pattern p will be constructed by stepwise refinement, starting with a pattern containing only the symbol $\mathcal{M}_0$.

3.3 Basic Lemmas

The following simple lemmas will be used in our lower bound argument.

Lemma 3.1 Let Λ be a comparator network in $\Lambda_0 \otimes \Lambda_1$, i be a nonnegative integer, and 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 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'.

Lemma 3.2 Let Λ be a comparator network, p be an input pattern for Λ, and 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 $\mathcal{S}_0$, all pattern symbols $\mathcal{P}$ with $\mathcal{M}_i <_P \mathcal{P}$ to $\mathcal{L}_0$, and all pattern symbols $\mathcal{M}_i$ to $\mathcal{M}_0$. If A is noncolliding in Λ under p, then A is also noncolliding in Λ under $\rho_i(p)$.

3.4 The Main Lemma

In this subsection, we establish our main lemma on the size of the incomparable set in a hypercubic network with small overlap. The main technical difficulty is in the proof of Lemma 3.3, which establishes the existence of a pattern p with a "large" $[\mathcal{M}_0]$-set that is noncolliding in a single (d, s, r)-hypercubic network under p.

Lemma 3.3 Let Δ be a (d,s,r)-hypercubic network with $r \leq s \leq d$, and 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 k be any positive integer. Then there exists an input pattern q with $p \supset_A q$ and $t(s) \stackrel{\text{def}}{=} 2^r \cdot k^3 + 2^r \cdot (s-r) \cdot k^2$ sets M_i, $0 \leq i < t(s)$, of input wires such that the following properties hold, where $B \stackrel{\text{def}}{=} \bigcup_{0 \leq i < t(s)} M_i$:

(1) Every M_i is the $[\mathcal{M}_i]$-set of q.
(2) Every M_i is noncolliding in Δ under q.
(3) $B \subset A$.
(4) $|B| \geq |A| - \frac{(s-r)\cdot|A|}{k^2}$.
(5) No two elements of any $[\mathcal{M}_i]$-set of $\Delta(q)$ are located in the same output group of Δ.

Proof: (Sketch) The proof is very similar to that of Lemma 4.1 in [17], and hence we only sketch the necessary modifications. A complete proof can be found in [18].

The proof is by induction over $s-r$ with base case $s-r=0$. Properties (1) to (4) are nearly the same as in [17]. In addition, the induction also has to maintain the new Property (5). In order to do so, the number of sets M_i has to be increased by a factor of 2^r. This means that the number of possible matchings between the sets $M_{0,i}$ and $M_{1,j}$ in the induction step also increases by a factor of 2^r. Since each element in a set $M_{0,i}$ can collide with at most 2^r elements in the sets $M_{1,j}$, by averaging there exists a matching under which Property (5) can be maintained without throwing away too many of the elements. $\square$

Lemma 3.4 Let Δ be a 2^d-input hypercubic network with span $3 \leq s \leq d$ and overlap r, and let Λ be an arbitrary comparator network with an incomparable set of size ν. Then any network in $\Lambda \otimes \Delta$ has an incomparable set of size $\nu' \geq \nu/(s^4 \cdot 2^r)$.

Proof: According to Definition 3.8, there exists an input pattern p_0 such that some $[\mathcal{M}_{i_0}]$-set C of p_0 is of size ν and is noncolliding in Λ under p_0. By Lemma 3.2, we can assume that $i_0 = 0$, and that p_0 contains only the symbols $\mathcal{S}_0$, $\mathcal{M}_0$, and $\mathcal{L}_0$.

Every 2^d-input hypercubic network with span $s \leq d$ and overlap r is equivalent to a (d,s,r)-hypercubic network. Hence, we can apply Lemma 3.3 to Δ. Let $k = s$, $p = \Lambda(p_0)$, and A be the $[\mathcal{M}_0]$-set of p. Then by Lemma 3.3, there exists an input pattern q with $p \supset_A q$ and $t(s) \leq 2s^3 \cdot 2^r$ disjoint sets M_i, $0 \leq i < t(s)$ 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,
- $B \subset A$, and
- $|B| \geq \nu \cdot (1 - 1/s)$,

where $B \stackrel{\text{def}}{=} \bigcup_{0 \leq i < t(s)} M_i$. By averaging, there exists a set M_{j_0}, $0 \leq j_0 < t(s)$, of size at least $|B|/(2s^3 \cdot 2^r) \geq \nu/(s^4 \cdot 2^r)$, where the inequality follows from the fact that $\frac{1}{2}(1-1/s) \geq 1/s$ for $s \geq 3$. By Lemma 3.1, there exists an input pattern q_0 with $p_0 \supset_C q_0$ such that $q = \Lambda(q_0)$ and the $[\mathcal{M}_{j_0}]$-set of q_0 is noncolliding in $\Lambda \otimes \Delta$ under q_0. Since $q = \Lambda(q_0)$, the $[\mathcal{M}_{j_0}]$-set of q_0 also contains at least $\nu/(s^4 \cdot 2^r)$ elements. $\square$

By partitioning a hypercubic network of overlap r and depth ℓ into $\lceil \ell/d \rceil$ consecutive hypercubic networks of overlap r and depth at most d, and applying Lemma 3.4 to each of these networks, we obtain the following lower bound for hypercubic networks with bounded overlap. Note that for the special case $r = 0$, we obtain the result in [17]. However, if the overlap is $\Theta(d)$, we only get the trivial $\Omega(\lg n)$ lower bound.

Theorem 3.1 Any n-input hypercubic sorting network with overlap r has depth $\Omega\left(\frac{\lg^2 n}{\max\{r, \lg\lg n\}}\right)$.

4 A Lower Bound for Hypercubic Networks

In this section we establish our main result, a lower bound on the depth of arbitrary hypercubic sorting networks. In order to prove the result, we need one more lemma. Informally, Lemma 4.1 below states that we can maintain a fairly large incomparable set over the levels of any hypercubic network. The proof of the lemma is based on the idea that any hypercubic network with depth ℓ either has a small overlap relative to ℓ, or can be (recursively) partitioned into several consecutive networks satisfying this property. In the first case, we can use Lemma 3.4 to bound the size of the incomparable set. The second case is handled by induction.

Lemma 4.1 Let Δ be a hypercubic network with depth ℓ and span $s \leq d$, let $\alpha(l,s) \stackrel{\text{def}}{=} (\ell - s/2)/(\lg s/\lg\lg s)$, and let Λ be an arbitrary comparator network with an incomparable set of size ν. Then any network in $\Lambda \otimes \Delta$ has an incomparable set of size ν', where

$$\frac{\nu}{\nu'} = s^4 \cdot 2^{O(\alpha(l,s))}.$$

Proof: We prove that $\nu/\nu' \leq c \cdot s^4 \cdot 2^{9\cdot\alpha(l,s)}$ holds for some positive constant c. The proof is by induction on the depth ℓ of the network. For the base case, it can be checked that the statement is true for small constant values of ℓ. Now assume that the statement has been shown for all networks of depth less than ℓ.

For the induction step, we assume a hypercubic network Δ with depth ℓ, overlap r, and span $s \leq d$. Now suppose that $r \leq 9 \cdot \alpha(l,s)$. In this case, the claim follows for any $c \geq 1$ by a simple application of Lemma 3.4.

Hence, in the following we have

$$r \;>\; 9 \cdot \alpha(l,s) \;\geq\; \frac{9s}{2\lg s/\lg\lg s}. \tag{1}$$

Due to the definition of overlap, there exist hypercubic networks Δ_i, $0 \leq i < 2$, with depth ℓ_i and span s_i, such that Δ belongs to $\Delta_0 \otimes \Delta_1$, $\ell_0 + \ell_1 = \ell$, and $s_0 + s_1 = s + r$. By applying the induction hypothesis first to Λ and Δ_0 , and then to $\Lambda \otimes \Delta_0$ and Δ_1, we obtain

$$\frac{\nu}{\nu'} \;\leq\; c \cdot s_0^4 \cdot 2^{9\cdot\alpha(l_0,s_0)} \cdot c \cdot s_1^4 \cdot 2^{9\cdot\alpha(l_1,s_1)} \;=\; c^2 \cdot s_0^4 \cdot s_1^4 \cdot 2^{9x},$$

where $x \stackrel{\text{def}}{=} \alpha(l_0,s_0) + \alpha(l_1,s_1)$. Using $s_0 \geq r$, $s_1 \geq r$, and Equation (1) we can show that

$$\min\left\{\frac{\lg s_0}{\lg\lg s_0}, \frac{\lg s_1}{\lg\lg s_1}\right\} \geq \frac{\lg s}{\lg\lg s} \cdot \left(1 - \frac{2\lg\lg s}{\lg s}\right).$$

Using this bound, and the fact that $1/(1-\epsilon) \leq 1 + 2\epsilon$ holds for sufficiently small $\epsilon > 0$, we obtain

$$\begin{aligned} x \;&\leq\; \frac{\ell_0 - s_0/2}{\lg s/\lg\lg s} + \frac{\ell_1 - s_1/2}{\lg s/\lg\lg s} + (\ell_0 - s_0/2 + \ell_1 - s_1/2) \cdot 4\left(\frac{\lg\lg s}{\lg s}\right)^2 \\ &=\; \frac{\ell - s/2 - r/2}{\lg s/\lg\lg s} + (\ell - s/2 - r/2) \cdot 4\left(\frac{\lg\lg s}{\lg s}\right)^2, \end{aligned}$$

where the last step follows from $\ell_0 + \ell_1 = \ell$ and $s_0 + s_1 = s + r$. Applying Equation (1), we obtain

$$x \leq \alpha(l,s) - \frac{r}{2\lg s/\lg\lg s} + \frac{4r}{9\lg s/\lg\lg s} = \alpha(l,s) - \frac{r}{18\lg s/\lg\lg s}.$$

Hence, for s sufficiently large, we have $\nu/\nu' \leq c \cdot s^4 \cdot 2^{9\cdot\alpha(l,s)}$. $\square$

Theorem 4.1 Any n-input hypercubic sorting network has depth $\Omega\left(\frac{\lg n \lg\lg n}{\lg\lg\lg n}\right)$.

Proof: Let Δ be an n-input hypercubic network of depth ℓ, $n = 2^d$. Then we can partition Δ into $k = \lceil \ell/d \rceil$ consecutive hypercubic networks Δ_i, $0 \leq i < k$, with depth ℓ_i and span at most d.

Let Λ be a network containing no comparator elements at all. Clearly, Δ belongs to $\Lambda \otimes \Delta$, and Λ has an incomparable set of size n. We now apply Lemma 4.1 once for each network Δ_i, $0 \leq i < k$. It follows that there exists an incomparable set of size n' in Δ, such that

$$\frac{n}{n'} = \prod_{0 \leq i < k} d^4 \cdot 2^{O\left(\frac{(\ell_i - d/2)}{\lg d/\lg\lg d}\right)} = 2^{O\left(\frac{\ell}{\lg d/\lg\lg d}\right)}.$$

Hence, if $\ell < c \cdot d \lg d/\lg\lg d$ for some sufficiently small positive constant c, we find that $n' > 1$, and it follows that Δ cannot be a sorting network. $\square$

5 Extensions

The lower bound for hypercubic sorting networks can be extended to certain restricted classes of non-oblivious sorting algorithms on hypercubic machines. More precisely, the particular function computed by a comparator (that is, "+", "-", "0", or "1") may depend on the outcomes of all comparisons made in previous levels of the network. Also, the lower bounds still hold in the case where a node can hold more than one element, provided that elements cannot be copied. It remains unclear whether our results can be extended to a model where copying of elements is allowed. Finally, our lower bound technique can also be applied to some restricted classes of sorting algorithms on multi-dimensional meshes. Examples of algorithms in these classes were recently given by Corbett and Scherson [4] and Wanka [20].

On the other hand, our lower bounds do not apply to "probabilistic" sorting networks that sort the vast majority of input permutations, or to "randomized" sorting networks that contain additional "randomizing" circuit elements. For these types of networks, Leighton and Plaxton [12] have given hypercubic constructions of depth $O(\lg n)$. A more detailed discussion of these extensions and limitations can be found in [18].

6 Concluding Remarks

In this paper, we have established an $\Omega\left(\frac{\lg n \lg\lg n}{\lg\lg\lg n}\right)$ lower bound on the depth of hypercubic sorting networks. The proof technique also applies to certain restricted classes of non-oblivious sorting algorithms on hypercubes and multi-dimensional meshes. A gap remains between our lower bound and the best upper bound known, and it would certainly be an interesting improvement to narrow or close this gap.

An important open question is whether we can extend our lower bounds to more general classes of non-oblivious sorting algorithms on the hypercube. Of particular interest in this respect would be the class of normal comparison-based sorting algorithms, or any other natural class of algorithms including the *Sharesort* algorithm of Cypher and Plaxton [7].

Another possible direction for future research would be to consider other restricted classes of sorting networks. Finally, it is an open problem whether our lower bound technique can be applied to selection networks.

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] V. Chvátal. Lecture notes on the new AKS sorting network. Technical Report DCS–TR–294, Department of Computer Science, Rutgers University, 1992.

[4] P. F. Corbett and I. D. Scherson. Sorting in mesh connected multiprocessors. *IEEE Transactions on Parallel and Distributed Systems*, 3:626–632, 1992.

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

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

[7] R. E. Cypher and C. G. Plaxton. Deterministic sorting in nearly logarithmic time on the hypercube and related computers. *JCSS*, 47:501–548, 1993.

[8] M. Dowd, Y. Perl, L. Rudolph, and M. Saks. The periodic balanced sorting network. *JACM*, 36:738–757, 1989.

[9] M. Kik, M. Kutyłowski, and G. Stachowiak. Periodic constant depth sorting networks. In *Proceedings of the 11th Symposium on Theoretical Aspects of Computer Science*, pages 201–212, February 1994.

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

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

[12] 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.

[13] F. T. Leighton and C. G. Plaxton. Hypercubic sorting networks. Unpublished manuscript, August 1993.

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

[15] 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.

[16] C. G. Plaxton, B. Poonen, and T. Suel. Improved lower bounds for Shellsort. In *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 226–235, October 1992.

[17] C. G. Plaxton and T. Suel. A lower bound for sorting networks based on the shuffle permutation. In *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 70–79, June 1992. To appear in *Mathematical Systems Theory.*

[18] C. G. Plaxton and T. Suel. A super-logarithmic lower bound for hypercubic sorting networks. Technical Report TR–94–08, University of Texas at Austin, Department of Computer Science, April 1994. Available via anonymous ftp from `ftp.cs.utexas.edu`.

[19] V. R. Pratt. *Shellsort and Sorting Networks.* PhD thesis, Stanford University, Department of Computer Science, December 1971. Also published by Garland, New York, 1979.

[20] R. Wanka. Fast general sorting on meshes of arbitrary dimension without routing. Technical Report TR–RI–91–087, Department of Computer Science, University of Paderborn, August 1991.