# A Super-Logarithmic Lower Bound for Shuffle-Unshuffle Sorting Networks [1]

*C. Greg Plaxton* [2]  *Torsten Suel* [3]

**Abstract**

Shuffle-unshuffle 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 shuffle-unshuffle 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 shuffle-unshuffle 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 shuffle-unshuffle sorting network. Our lower bound can be extended to certain restricted classes of non-oblivious sorting algorithms on hypercubic machines.

[1] A preliminary version of this paper was presented at the 21st International Colloquium on Automata, Languages, and Programming (ICALP), 1994.

[2] Department of Computer Science, University of Texas, Austin, TX 78712. 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.

[3] CIS Department, Polytechnic University, Brooklyn, NY 11201. Email: `suel@photon.poly.edu`. Part of this work was performed while the author was a graduate student at the University of Texas, and was supported by the Texas Advanced Research Program under Grant Nos. 003658–480 and 003658–461, and by a Schlumberger Graduate Fellowship.

## 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 [3]. 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 [2, 4, 15], 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 [7] 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. Partly because of these properties, Batcher's networks have been used 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, periodic sorting networks, and shuffle-unshuffle (also called hypercubic [14]) sorting networks.

*Shellsort* networks have a very simple structure that is based on the sequential *Shellsort* sorting algorithm. A class of Shellsort networks with depth $\Theta(\lg^2 n)$ was given by Pratt [19]. For Shellsort networks based on monotonically decreasing increment sequences, Cypher [6] has established a lower bound of $\Omega(\lg^2 n/\lg\lg n)$. Recently, more general lower bounds were shown [16, 18] that hold for arbitrary Shellsort networks and even sequential Shellsort algorithms, thus answering in the negative the longstanding open problem of whether a running time of $O(n \lg n)$ can be achieved by any Shellsort algorithm.

A comparator network is called a *periodic* sorting network if every input permutation can be sorted by repeatedly passing it through the network. The primary motivation for such periodic networks is the reduction in hardware cost achieved by applying the same network repeatedly to the input. A periodic sorting network of depth $O(\lg n)$ and running time $O(\lg^2 n)$ was given by Dowd, Perl, Rudolph, and Saks [9]. Very recently, Kutyłowski, Loryś, Oesterdiekhoff, and Wanka [12] have shown the existence of periodic networks of depth $5$ and running time $O(\lg^2 n)$ based on expanders. No non-trivial lower bounds for periodic sorting networks are currently known.

In this paper, we focus on the class of *shuffle-unshuffle* 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 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 the classes of sorting networks that we consider.

## 1.1 Shuffle-Unshuffle 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 $(\Pi_i, \vec{x}_i)$, $0 \leq i < \ell$, where $\Pi_i$ 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 $\ell$ consecutive steps. In step $i$, $0 \leq i < \ell$, the register contents are permuted according to $\Pi_i$, and then the operation stored in the $k$th 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.

It is well known that the two models of comparator networks described above are equivalent. (That is, given any network in one model, there exists a network in the other model with the same size and depth that implements the same mapping from inputs to outputs.) 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 $\Pi_i$.

The *shuffle permutation* $\pi_{sh}$ on $n = 2^d$ inputs may be defined as follows. If $j_{d-1} \cdots j_0$ denotes the binary representation of some integer $j$, $0 \leq j < n$, then $\pi_{sh}(j)$ has binary representation $j_{d-2} \cdots j_0 j_{d-1}$. A sorting network is called a *shuffle-unshuffle* network if $\Pi_i = \pi_{sh}$ or $\Pi_i = \pi_{sh}^{-1}$ holds for all $i$. A natural subclass of the shuffle-unshuffle 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 shuffle-unshuffle 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 [13] 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 shuffle-based sorting networks was proposed by Knuth [11, Exercise 5.3.4.47]. The best upper bound for this class is given by Batcher's bitonic sort [3], with a depth of $O(\lg^2 n)$. A lower bound of $\Omega(\lg^2 n/\lg\lg n)$ was established by Plaxton and Suel [17]. However, this lower bound does not extend to arbitrary shuffle-unshuffle networks.

The class of shuffle-unshuffle (also called hypercubic) sorting networks was defined by Leighton and Plaxton [14], who show the existence of a family of shuffle-unshuffle 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 [14], which sorts all but a super-polynomially small fraction of the possible input permutations. We point out that the depth of the above shuffle-unshuffle 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 shuffle-unshuffle 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 shuffle-unshuffle 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, and does not extend to randomized algorithms. Thus, neither the deterministic *Sharesort* sorting algorithm of Cypher and Plaxton [8], which achieves a running time of $O(\lg n \lg\lg n)$ (with preprocessing) on any of the hypercubic machines, nor the randomized approach by Reif and Valiant and subsequent related schemes [13, 20] are covered by 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.

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 shuffle-unshuffle 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 shuffle-unshuffle sorting networks. We will then describe the new proof ideas that are needed in order to get a lower bound for arbitrary shuffle-unshuffle 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 $\ell$ for the depth of a class of comparator networks by showing, for all networks in the class, the existence of an input permutation $\pi$, and of a set of adjacent values $\{m, \ldots, m+i\}$ in $\pi$, for some $i \geq 0$, such that no two elements of the set are compared up to level $\ell$ of the network. In the following, we will call such a set an *incomparable set*. If we apply this proof idea to a shuffle-unshuffle 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 is equivalent to 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). Thus, every $2$-input butterfly has two different collections of $\lg^3 n$ incomparable sets arriving on its two input wires. It is now possible to recombine these sets to get a new collection of roughly $\lg^3 n$ incomparable sets, containing all of the elements of the two collections.

More generally, 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 Shuffle-Unshuffle Sorting Networks

The above argument does not work for arbitrary shuffle-unshuffle 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 shuffle-unshuffle sorting networks. To do so, we introduce the class of shuffle-unshuffle networks with "bounded overlap".

Assume we are given an arbitrary shuffle-unshuffle network $\Lambda$ with $\ell$ levels $(\Pi_i, \vec{x}_i)$, $0 \le i < \ell$, as described in the register model of a comparator network. In order to define the "span" and "overlap" of $\Lambda$, it is convenient to introduce a number of auxiliary variables. Let $a_i = 1$ if $\Pi_i = \pi_{sh}$ and $a_i = -1$ if $\Pi_i = \pi_{sh}^{-1}$, $0 \le i < \ell$. (We remark that the value of $a_0$ has no impact on the definitions that follow.) Let $b_i = \sum_{1 \le j \le i} a_j$, $0 \le i < \ell$. The *span* of $\Lambda$ may now be defined as $|\{b_i : 0 \le i < \ell\}|$. The *overlap* of $\Lambda$ is the minimum integer $r \ge 0$ such that either: (i) $b_i \le b_j + r$ for all $0 \le i < j < \ell$, or (ii) $b_i \ge b_j - r$ for all $0 \le i < j < \ell$. Note that a network has overlap 0 iff $\Pi_i = \Pi_j$ for all $1 \le i < j < \ell$. 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 shuffle-unshuffle 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 shuffle-unshuffle network can be partitioned into a number of consecutive shuffle-unshuffle networks such that the overlap of each network in the partition is sufficiently smaller than its depth.

## 3 Shuffle-Unshuffle Networks with Small Overlap

In this section, we show that a large incomparable set can be effectively maintained over the levels of any shuffle-unshuffle network with sufficiently small overlap. The main result of this section is Lemma 3.6, which bounds the decrease in the size of the incomparable set that can occur in any $2^d$-input shuffle-unshuffle network with span $s \le d$ and overlap $r$. This lemma is used in Section 4 to establish our lower bound for arbitrary shuffle-unshuffle sorting networks.

The actual argument addressing the size of the incomparable set is contained in the proof of Lemma 3.5, 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 shuffle-unshuffle networks with span $s$ and overlap $r$. The proof of Lemma 3.5 has a very similar structure to that of Lemma 4.1 in [17], and uses many of the techniques introduced in that paper. However, in addition to these techniques, we need some new ideas to establish the result.

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 shuffle-unshuffle 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 $\pi$ that are permutations of $\{0, \ldots, n-1\}$, i.e., where $|W| = n$, $V = \{0, \ldots, n-1\}$, and $\pi$ 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 constraints 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 $p_0 \sqsupset_W p_1$) 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 $\pi$ be an input. We say that $p$ can be refined to $\pi$ (written $p \sqsupset_W \pi$) 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.

**Example 3.1** *Let $W \stackrel{\text{def}}{=} \{w_0, \ldots, w_{n-1}\}$, $P \stackrel{\text{def}}{=} \{\mathcal{S}, \mathcal{M}, \mathcal{L}\}$, and let the ordering $<_P$ on $P$ be given by $\mathcal{S} <_{\mathcal{P}} \mathcal{M} <_{\mathcal{P}} \mathcal{L}$. (Informally, the symbols $\mathcal{S}$, $\mathcal{M}$, and $\mathcal{L}$ may be interpreted as "Small", "Medium",* *and “Large”, respectively.) 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 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 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 presentation of the proof.

We usually think of an input pattern $p$ as a description of the set of inputs to which $p$ can be refined. This set is denoted $p[V] \overset{\text{def}}{=} \{\pi : \pi$ 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 $W \setminus U$.*

**Definition 3.3** *Let $U_0$ and $U_1$ be disjoint subsets of $W$, $p_0$ be an input pattern on $U_0$, and $p_1$ 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 $W \overset{\text{def}}{=} \{w_0, \ldots, w_{n-1}\}$ and $P \overset{\text{def}}{=} \{\mathcal{P}_i : i \geq 0\}$ 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 every pattern symbol $\mathcal{P}_i$ in $p$ by $\mathcal{P}_{i+k}$, for all $i$.*

## 3.2 Comparator Networks

We now further 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.

In the following, 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 $\Lambda$ on input wires $W$ and output wires $W'$ defines a mapping (which we also denote by $\Lambda$) 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 $\pi$. 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 $\Lambda_0^*$, $\Lambda_1^*$ be two sets of $n$-input comparator networks. Then $\Lambda_0^* \otimes \Lambda_1^*$, the *serial composition* of $\Lambda_0^*$ and $\Lambda_1^*$, denotes the set of all networks $\Lambda$ that can be obtained by connecting the output wires of a network from $\Lambda_0^*$ to the input wires of a network from $\Lambda_1^*$. 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 $\Lambda_0^*$ and $\Lambda_1^*$. In such a case, we may write, for example, $\Lambda_0 \otimes \Lambda_1$ (where $\Lambda_0$, $\Lambda_1$ are networks) rather than $\{\Lambda_0\} \otimes \{\Lambda_1\}$.

Given two comparator networks $\Lambda_0$ and $\Lambda_1$ on disjoint sets of input and output wires, we obtain the *parallel composition* of $\Lambda_0$ and $\Lambda_1$ 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 $\Lambda_0$ and $\Lambda_1$.

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 shuffle-unshuffle 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 3.4** *For $r \leq s \leq d$, a $2^d$-input comparator network $\Delta$ is called a $(d, s, r)$-*hypercubic *network if:*

*(a) $s - r = 0$, and $\Delta$ 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 $\Delta$ have been partitioned into $2^{d-r}$ output groups of size $2^r$, or*

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

- *$\Delta_0$ and $\Delta_1$ are $(d - 1, s - 1, r)$-hypercubic networks, and*
- *$\Lambda$ is the parallel composition of $2^{d-r-1}$ disjoint $2^{r+1}$-input comparator networks $\Lambda_i$, $0 \leq i < 2^{d-r-1}$, of arbitrary size and depth, such that: (i) the $2^{r+1}$ input wires of each network $\Lambda_i$ are connected to one output group of size $2^r$ of $\Delta_0$ and one output group of size $2^r$ of*

*$\Delta_1$, and (ii) the $2^{r+1}$ output wires of each network $\Lambda_i$ are partitioned to form two of the $2^{d-r}$ output groups of network $\Delta$.*

We remark that the class of $(d, s, r)$-hypercubic networks may not appear to be a very natural or interesting class of networks, and that we only introduce it to simplify the lower bound argument in this paper.

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

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

Note that this definition characterizes the behavior of a comparator network on an input pattern in the way 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. This implies that any set of inputs that can be expressed by an input pattern will produce a set of outputs that can be expressed by an output pattern.

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

According to the above definition, two wires whose respective values meet in a noncomparator element, that is, a “0” (do nothing) or “1” (exchange) switch, are not regarded as colliding. 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 to input patterns in the following.

Given a network $\Lambda$ and an input $\pi$, 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 occurrences of the same pattern symbol. This motivates the following definition of collision for input patterns:

**Definition 3.7** *Let $\Lambda$ be a comparator network, let $p$ be an input pattern for $\Lambda$, and let $w_0$ and $w_1$ be two input wires of $\Lambda$.*

(a) *We say that $w_0$ and $w_1$* collide *in $\Lambda$ under $p$ if they collide in $\Lambda$ under every input in $p[V]$.*

(b) *We say that $w_0$ and $w_1$* can collide *in $\Lambda$ under $p$ if there exists an input in $p[V]$ such that $w_0$ and $w_1$ collide in $\Lambda$.*

(c) *We say that $w_0$ and $w_1$* cannot collide *in $\Lambda$ under $p$ if there is no input in $p[V]$ such that $w_0$ and $w_1$ collide in $\Lambda$.*

(d) *A set $U \subset W$ is called* non-colliding *in $\Lambda$ under $p$ if any two wires in $U$ cannot collide in $\Lambda$ under $p$.*

**Example 3.3** *Let $W \stackrel{\text{def}}{=} \{w_0, w_1, w_2, w_3\}$, $P \stackrel{\text{def}}{=} \{\mathcal{S}, \mathcal{M}, \mathcal{L}\}$, and let the ordering $<_P$ on $P$ be given by $\mathcal{S} <_P \mathcal{M} <_P \mathcal{L}$. Let the network $\Lambda$ consist of a comparator between $w_1$ and $w_2$, followed by a comparator between $w_2$ and $w_3$, followed by a comparator between $w_0$ and $w_3$, where all comparators are directed such that the larger value is output on the wire with the larger index. Then the following holds under the input pattern $p$ that maps $w_0$ to $\mathcal{S}$, $w_1$ and $w_2$ to $\mathcal{M}$, and $w_3$ to $\mathcal{L}$:*

(1) *Wires $w_1$ and $w_2$ collide in $\Lambda$ under $p$ since the very first comparator is between these two wires.*

(2) *Wires $w_1$ and $w_3$ can collide in $\Lambda$ under $p$, since we can refine $p$ to an input $\pi$ that assigns a larger value to $w_1$ than to $w_2$. In that case, the input value assigned to $w_1$ will be compared to that of $w_3$ in the second comparator. Similarly, $w_2$ can collide with $w_3$ in $\Lambda$ under $p$.*

(3) *Wires $w_0$ and $w_3$ collide in $\Lambda$ under $p$, since no exchange can occur in the second comparator of the network under any input $\pi$ with $p \supset_W \pi$. Also, $w_0$ and $w_1$ (resp. $w_2$) cannot collide in $\Lambda$ under $p$.*

Note that, if two wires collide (cannot collide) in some network $\Lambda$ under an input pattern $p$, then they also collide (cannot collide) in $\Lambda$ under any refinement $p'$ of $p$. Similarly, if a set $U$ is non-colliding in $\Lambda$ under $p$, then it is also non-colliding in $\Lambda$ 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 is used throughout the lower bound argument:

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

The ordering $<_P$ on $P$ is defined by

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

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

for all nonnegative integers $i, j$.

Note that these inequalities imply that $\mathcal{X}_{i,j} <_P \mathcal{X}_{i+1,j}$, $\mathcal{M}_i <_P \mathcal{M}_{i+1}$, and $\mathcal{S}_i <_P \mathcal{M}_j <_P \mathcal{L}_k$ for all nonnegative $i, j, k$. (Thus, the letters $\mathcal{S}$, $\mathcal{M}$, and $\mathcal{L}$ are chosen to denote various classes of "small", "medium", and "large" input values.)

**Definition 3.8** *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 3.9** *We say that a comparator network $\Lambda$ has an* incomparable set *of size $m$ if there exists an input pattern $p$ and an integer $i$ such that the [$\mathcal{M}_i$]-set of $p$ is of size $m$ and is non-colliding in $\Lambda$ under $p$.*

We can now formally describe our proof strategy: To prove that a network $\Lambda$ 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 [$\mathcal{M}_i$]-set contain adjacent input values; this implies that $\Lambda$ does not sort all inputs in $p[V]$. The input pattern $p$ will be constructed using stepwise refinement, starting out 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.

**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 $\mathcal{S}_0 <_P q_0(w), q_1(w) <_P \mathcal{L}_0$ for all $w$ in $A$. Then from $p_{|W_0} \sqsupset_{A \cap W_0} q_0$ and $p_{|W_1} \sqsupset_{A \cap W_1} q_1$, we can infer $p \sqsupset_A q_0 \oplus q_1$.*

This lemma ensures that, given an input pattern $p$ for a network $\Lambda = \Lambda_0 \oplus \Lambda_1$, we obtain a refinement of $p$ if we separately refine the input patterns $p_{|W_0}$ for $\Lambda_0$ and $p_{|W_0}$ for $\Lambda_1$ according to the above rules, where $W_0$ and $W_1$ are the sets of input wires of $\Lambda_0$ and $\Lambda_1$, respectively.

**Lemma 3.2** *Let $\Lambda$ be a comparator network, $p$ be an input pattern for $\Lambda$, and $A$ be the [$\mathcal{M}_i$]-set of $p$, for some $i \geq 0$. If $A$ is non-colliding in $\Lambda$ under $p$, then for every input wire $w$ in $A$ there exists a unique output wire $w'$ such that $\pi(w) = \Lambda(\pi)(w')$ holds for all $\pi$ in $p[V]$.*

Informally, the above lemma states that an input value on a wire $w$ in a non-colliding [$\mathcal{M}_i$]-set follows the same "path" through the network under all inputs in $p[V]$. The proof of the lemma is by a simple induction on the depth of the network. This one-to-one correspondence between the input and output wires of a non-colliding [$\mathcal{M}_i$]-set is also the underlying idea in the next lemma.

**Lemma 3.3** *Let $\Lambda$ be a comparator network in $\Lambda_0 \otimes \Lambda_1$, $i$ be a nonnegative integer, and $p$ be an input pattern for $\Lambda_0$ such that its [$\mathcal{M}_i$]-set $A$ is non-colliding in $\Lambda_0$ under $p$. Let $q \overset{\text{def}}{=} \Lambda_0(p)$ be an input pattern for $\Lambda_1$ and $B$ be the [$\mathcal{M}_i$]-set of $q$. Then for every $q'$ with $q \sqsupset_B q'$ there exists a $p'$ with $p \sqsupset_A p'$ such that $q' = \Lambda_0(p')$. Furthermore, if the [$\mathcal{M}_i$]-set of $q'$ is non-colliding in $\Lambda_1$ under $q'$, then the [$\mathcal{M}_i$]-set of $p'$ is non-colliding in $\Lambda$ 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 $\Lambda$ be a comparator network, $p$ be an input pattern for $\Lambda$, and $A$ be the [$\mathcal{M}_i$]-set of $p$, for some $i \geq 0$. 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 non-colliding in $\Lambda$ under $p$, then $A$ is also non-colliding in $\Lambda$ under $\rho_i(p)$.*

## 3.4 The Main Lemma

In this subsection, we establish our main lemma (Lemma 3.6) on the size of the incomparable set in a shuffle-unshuffle network with small overlap. The main technical difficulty is in the proof of Lemma 3.5, which establishes the existence of a pattern $p$ with a "large" [$\mathcal{M}_0$]-set that is non-colliding in a single $(d, s, r)$-hypercubic network under $p$. By a direct application, we also obtain a strong lower bound for shuffle-unshuffle sorting networks with bounded overlap that generalizes the result in [17].

**Lemma 3.5** *Let $\Delta$ be a $(d, s, r)$-hypercubic network with $r \leq s \leq d$, and $p$ be an input pattern for $\Delta$ 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 $p \sqsupset_A q$ and $t(s) \overset{\text{def}}{=} 2^r \cdot k^3 + (s - r) \cdot 2^r \cdot k^2$ sets $M_i$, $0 \leq i < t(s)$, of input wires such that the following properties hold, where $B \overset{\text{def}}{=} \bigcup_{0 \leq i < t(s)} M_i$:*

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

*(2) Every $M_i$ is non-colliding in $\Delta$ 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 $\Delta$.*

**Proof:** The proof is by induction on $s - r$.

Base Case: $s - r = 0$

In this case the network $\Delta$ does not contain any comparator elements. We define the sets $M_i$, $0 \leq i < t(0)$, by partitioning $A$ into $2^r \cdot k^3$ sets such that no two elements in any set are located in the same output group. (Each output group has size $2^r \leq 2^r \cdot k^3$, so this is clearly possible.) If we define $q$ as the pattern obtained from $p$ by relabeling each wire in set $M_i$ with $\mathcal{M}_i$, for $0 \leq i < t(0)$, then Properties (1) to (5) are satisfied.

Induction Step: $s - r > 0$

A $(d, s, r)$-hypercubic network consists of two $(d-1, s-1, r)$-hypercubic networks $\Delta_0$ and $\Delta_1$, and a network $\Lambda$ satisfying the conditions of Definition 3.4. The input wires $W$ of $\Delta$ can be partitioned into the sets $W_0$ and $W_1$ of input wires of $\Delta_0$ and $\Delta_1$, respectively. Let $p_0 \overset{\text{def}}{=} p_{|W_0}$ and $p_1 \overset{\text{def}}{=} p_{|W_1}$. Then $A_0 \overset{\text{def}}{=} A \cap W_0$ is the [$\mathcal{M}_0$]-set of $p_0$ and $A_1 \overset{\text{def}}{=} A \cap W_1$ is the [$\mathcal{M}_0$]-set of $p_1$.

Applying the induction hypothesis to $\Delta_0$, $p_0$, and $A_0$, we can infer the existence of an input pattern $q_0$ with $p_0 \supset_{A_0} q_0$, and of $t(s-1)$ disjoint sets $M_{0,i}$, $0 \leq i < t(s-1)$, such that

- every $M_{0,i}$ is the [$\mathcal{M}_i$]-set of $q_0$,
- every $M_{0,i}$ is non-colliding in $\Delta_0$ under $q_0$,
- $B_0 \subset A_0$,
- $|B_0| \geq |A_0| - \frac{(s-r-1)\cdot|A_0|}{k^2}$, and
- no two elements of any [$\mathcal{M}_i$]-set of $\Delta_0(q_0)$ are located in the same output group of $\Delta_0$,

where $B_0 \overset{\text{def}}{=} \bigcup_{0 \leq i < t(s-1)} M_{0,i}$.

Correspondingly, for $\Delta_1$, $p_1$, and $A_1$, we get an input pattern $q_1$, disjoint sets $M_{1,i}$, $0 \leq i < t(s-1)$, and a set $B_1$, with the same properties.

We will now construct the sets $M_i$, $0 \leq i < t(s)$, by combining the sets $M_{0,i}$ of $\Delta_0$ with the sets $M_{1,j}$ of $\Delta_1$, according to some partial matching to be determined in the following.

Because no $[\mathcal{M}_i]$-set of $\Delta_0(q_0)$ (resp., $\Delta_1(q_1)$) contains any two elements that are located in the same output group of $\Delta_0$ (resp., $\Delta_1$), no element of any set $M_{0,i}$ (resp., $M_{1,i}$) can collide with any other element of the same set in $\Delta$.

Also, due to the topology of a $(d, s, r)$-hypercubic network, no element of a set $M_{0,i}$ can collide in $\Delta_0 \oplus \Delta_1$ with any element of a set $M_{1,j}$. By Lemma 3.2, we can determine for each $w$ in a set $M_{0,i}$ (resp., $M_{1,j}$) the output wire $w'$ of $\Delta_0$ (resp., $\Delta_1$) that receives the value $\pi(w)$ under all $\pi$ in $q_0[V]$ (resp., $q_1[V]$). Thus, for any such $w$ we can determine the subnetwork $\Lambda_\nu$ (where $\nu$ is some function $f$ of $w$) of $\Lambda$ that will receive $\pi(w)$ as an input value under all $\pi$ in $q_0[V]$ (resp., $q_1[V]$).

For $0 \leq i, j < t(s-1)$, we define $C_{i,j}$ as the set of all wires $w_0$ in $M_{0,i}$ such that $f(w_0) = f(w_1)$ holds for some $w_1$ in $M_{1,j}$. Note that the $C_{i,j}$'s are not pairwise disjoint. However, since each subnetwork $\Lambda_\nu$ receives only $2^r$ input values from $\Delta_1$, every element $w_0$ in $M_{0,i}$ is contained in at most $2^r$ sets $C_{i,j}$. Also, each $C_{i,j}$ contains all wires in $M_{0,i}$ that can collide in $\Lambda$ with some wire in $M_{1,j}$.

For $0 \leq i < 2^r \cdot k^2$ and $0 \leq j < t(s)$, we define

$$M(i,j) \overset{\text{def}}{=} \begin{cases} M_{0,j} & 0 \leq j < i, \\ (M_{0,j} \setminus C_{j,j-i}) \cup M_{1,j-i} & i \leq j < t(s-1), \\ M_{1,j-i} & t(s-1) \leq j < t(s-1) + i, \text{ and} \\ \emptyset & t(s-1) + i \leq j < t(s). \end{cases}$$

By their construction, the sets $M(i,j)$ are non-colliding in $\Delta$ under $q_0 \oplus q_1$. If we let $L_i \overset{\text{def}}{=} \bigcup_{i \leq j < t(s-1)} C_{j,j-i}$ for $0 \leq i < 2^r \cdot k^2$, then

$$\bigcup_{0 \leq j < t(s)} M(i,j) = (B_0 \setminus L_i) \cup B_1.$$

Since every element of $B_0$ can occur at most $2^r$ times in the sets $C_{i,j}$, every element of $B_0$ can occur at most $2^r$ times in the sets $L_i$. Hence, by averaging there exists an $i_0$, $0 \leq i_0 < k^2 \cdot 2^r$, such that $|L_{i_0}| \leq \frac{|B_0|}{k^2}$. We use this $i_0$ to determine the partial matching between the $M_{0,i}$'s and the $M_{1,j}$'s.

More precisely, for all $j$ such that $0 \leq j < t(s)$, we match the set $M_{0,j}$ with the set $M_{1,j-i_0}$ to obtain a new set $M_j \overset{\text{def}}{=} M(i_0, j)$. (Here we assume $M_{0,i}$ and $M_{1,i}$ to be the empty set for $i < 0$ and $i \geq t(s-1)$.) Thus, the new set $M_j$ is obtained by removing the wires in $C_{j,j-i_0}$ from $M_{0,j}$, and merging the resulting set with $M_{1,j-i_0}$. We now show that this choice of $M_j$ satisfies Properties (3) and (4). We have

$$B \overset{\text{def}}{=} \bigcup_{0 \leq j < t(s)} M_j$$

$$\begin{aligned} &= (B_0 \setminus L_{i_0}) \cup B_1 \\ &\subset B_0 \cup B_1 \\ &\subset A_0 \cup A_1 \\ &= A. \end{aligned}$$

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

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

To complete our proof, we construct a refinement $q$ of $p$ such that Properties (1), (2), and (5) 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 $q \overset{\text{def}}{=} q_0' \oplus q_1'$ 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 $i \geq t(s-1)$ to $\mathcal{M}_{i+2^r\cdot k^2}$ and $\mathcal{X}_{i+2^r\cdot k^2,j}$, respectively.
2. Then change the pattern symbols of all wires in $C_{i,i-i_0}$ with $i_0 \leq i < t(s-1)$ 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 $i \geq t(s-1)$ to $\mathcal{M}_{i+2^r\cdot k^2}$ and $\mathcal{X}_{i+2^r\cdot k^2,j}$, respectively.

$2'$. Then change all pattern symbols $\mathcal{M}_i$ and $\mathcal{X}_{i,j}$ with $0 \leq i < t(s-1)$ 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 $t(s-1) \leq i < t(s)$ from the patterns. 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 hence that no symbols $\mathcal{M}_i$ and $\mathcal{X}_{i,j}$ with $i \geq t(s-1)$ exist in the pattern. However, in order to simplify our induction hypothesis, we have chosen not to make this assumption.

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 the set $C_{i,j}$, which contains all input wires of $M_{0,i}$ that can collide with an input wire of $M_{1,j}$ in $\Lambda$ 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 non-colliding in $\Delta_0$ under $q_0'$ and, thus, also non-colliding in $\Delta$ under $q$. Similarly, the sets $M_{1,j}$ are non-colliding in $\Delta$ under $q$. Hence,

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

is non-colliding in $\Delta$ under $q$.

Finally, due to the definition of the sets $C_{i,j}$ that were removed from the matched sets, no two elements of any [$\mathcal{M}_i$]-set of $\Delta(q)$ are in the same output group of $\Delta$. This establishes Property (5).

□

**Lemma 3.6** *Let $\Delta$ be a $2^d$-input shuffle-unshuffle network with span $s \leq d$ and overlap $r$, and let $\Lambda$ be an arbitrary comparator network with an incomparable set of size $\nu$. 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 3.9, there exists an input pattern $p_0$ such that some [$\mathcal{M}_{i_0}$]-set $C$ of $p_0$ is of size $\nu$ and is non-colliding in $\Lambda$ under $p_0$. By Lemma 3.4, 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 shuffle-unshuffle network with span $s \leq d$ and overlap $r$ is equivalent to a $(d, s, r)$-hypercubic network. Hence, we can apply Lemma 3.5 to $\Delta$. Let $k = s$, $p = \Lambda(p_0)$, and $A$ be the [$\mathcal{M}_0$]-set of $p$. Then by Lemma 3.5, 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 $\Delta$ such that

- every $M_i$ is the [$\mathcal{M}_i$]-set of $q$,
- every $M_i$ is non-colliding in $\Delta$ under $q$,
- $B \subset A$, and
- $|B| \geq \nu \cdot (1 - 1/s)$,

where $B \overset{\text{def}}{=} \bigcup_{0 \le i < t(s)} M_i$. By averaging, there exists a set $M_{j_0}$, $0 \le j_0 < t(s)$, of size at least

$$\frac{|B|}{2s^3 \cdot 2^r} \ge \frac{\nu}{s^4 \cdot 2^r},$$

where the inequality follows from the fact that $\frac{1}{2}(1 - 1/s) \ge 1/s$ for $s \ge 3$. (For $s < 3$, the claim follows from $\nu' \ge \nu/2^s$.) By Lemma 3.3, 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 non-colliding 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. □

The following lemma can be established by partitioning a shuffle-unshuffle network of overlap $r$ and depth $\ell$ into $\lceil \ell/d \rceil$ consecutive shuffle-unshuffle networks of overlap $r$ and depth at most $d$, and applying Lemma 3.6 to each of the networks.

**Lemma 3.7** *Let $\Lambda$ be an $n$-input shuffle-unshuffle network with depth $\ell$ and overlap $r \le d = \lg n$. Then $\Lambda$ has an incomparable set of size at least*

$$\frac{n}{(d^4 \cdot 2^r)^{\lceil \ell/d \rceil}}.$$

Lemma 3.7 immediately implies the following lower bound for shuffle-unshuffle 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 shuffle-unshuffle sorting network with overlap $r$ has depth $\Omega\left(\frac{\lg^2 n}{r + \lg\lg n}\right)$.*

## 4 A Lower Bound for Shuffle-Unshuffle Sorting Networks

In this section we establish our main result, a lower bound on the depth of arbitrary shuffle-unshuffle 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 shuffle-unshuffle network of span at most $d$. The proof of the lemma is based on the idea that any shuffle-unshuffle network with depth $\ell$ either has a small overlap relative to $\ell$, or can be (recursively) partitioned into several consecutive networks satisfying this property. In the first case, we can use Lemma 3.6 to bound the size of the incomparable set. The second case is handled by induction.

**Lemma 4.1** *Let $\Delta$ be a shuffle-unshuffle network with depth $\ell$ and span $s \le d$, let $\alpha(\ell, s) \overset{\text{def}}{=} \frac{\ell - s/2}{\lg s / \lg\lg s}$, and let $\Lambda$ be an arbitrary comparator network with an incomparable set of size $\nu$. Then any network in $\Lambda \otimes \Delta$ has an incomparable set of size $\nu'$, where*

$$\frac{\nu}{\nu'} \ge s^4 \cdot 2^{9 \cdot \alpha(\ell, s)}.$$

**Proof:** The proof is by induction on the depth $\ell$ of the network.

Base Case: $\ell \leq 2^{16}$

Using $s \leq \ell \leq 2^{16}$, we obtain $\lg s / \lg \lg s \leq 4$ and

$$9 \cdot \alpha(\ell, s) \;\geq\; \frac{9 \cdot \ell/2}{4} \;\geq\; \ell \;\geq r.$$

Then the claim follows by a simple application of Lemma 3.6.

Induction Step: $\ell > 2^{16}$

For the induction step, we assume a shuffle-unshuffle network $\Delta$ with depth $\ell$, overlap $r$, and span $s \leq d$. Now suppose that $r \leq 9 \cdot \alpha(\ell, s)$. In this case, the claim follows by a simple application of Lemma 3.6.

Hence, in the following we assume that

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

Note that $s \geq r > 9 \cdot \alpha(\ell, s)$ and $\ell > 2^{16}$ imply $s > 2^{16}$ and $\lg \lg s / \lg s < 1/4$.

Due to the definition of overlap, there exist shuffle-unshuffle networks $\Delta_i$, $0 \leq i < 2$, with depth $\ell_i$ and span $s_i$, such that $\Delta$ 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 $\Lambda$ and $\Delta_0$ , and then to $\Lambda \otimes \Delta_0$ and $\Delta_1$, we obtain

$$\begin{aligned} \frac{\nu}{\nu'} &\leq s_0^4 \cdot 2^{9 \cdot \alpha(\ell_0, s_0)} \cdot s_1^4 \cdot 2^{9 \cdot \alpha(\ell_1, s_1)} \\ &= s_0^4 \cdot s_1^4 \cdot 2^{9x}, \end{aligned}$$

where $x \overset{\text{def}}{=} \alpha(\ell_0, s_0) + \alpha(\ell_1, s_1)$. Using $\min\{s_0, s_1\} \geq r$, $\max\{s_0, s_1\} \leq s$, and Equation (1) we obtain

$$\begin{aligned} \min\left\{\frac{\lg s_0}{\lg \lg s_0}, \frac{\lg s_1}{\lg \lg s_1}\right\} &\geq \frac{\lg r}{\lg \lg s} \\ &\geq \frac{1}{\lg \lg s} \cdot \lg\left(\frac{9s}{2 \lg s / \lg \lg s}\right) \\ &\geq \frac{1}{\lg \lg s} \cdot \lg\left(\frac{s}{\lg s}\right) \\ &= \frac{\lg s}{\lg \lg s} \cdot \left(1 - \frac{\lg \lg s}{\lg s}\right). \end{aligned}$$

Using this bound, and the fact that $1/(1-\epsilon) \leq 1 + 2\epsilon$ holds for $\epsilon = \lg \lg s / \lg s < 1/2$, we obtain

$$x \;\leq\; \left(\frac{\ell_0 - s_0/2}{\lg s / \lg \lg s} + \frac{\ell_1 - s_1/2}{\lg s / \lg \lg s}\right)\left(\frac{1}{1 - \lg \lg s / \lg s}\right)$$

$$\begin{aligned}
&\leq \left(\frac{\ell_0 - s_0/2}{\lg s/\lg\lg s} + \frac{\ell_1 - s_1/2}{\lg s/\lg\lg s}\right)(1 + 2\lg\lg s/\lg s) \\
&= \frac{\ell_0 - s_0/2}{\lg s/\lg\lg s} + \frac{\ell_1 - s_1/2}{\lg s/\lg\lg s} + (\ell_0 - s_0/2 + \ell_1 - s_1/2)\cdot 2\left(\frac{\lg\lg s}{\lg s}\right)^2 \\
&= \frac{\ell - s/2 - r/2}{\lg s/\lg\lg s} + (\ell - s/2 - r/2)\cdot 2\left(\frac{\lg\lg s}{\lg s}\right)^2 \\
&\leq \alpha(\ell, s) - \frac{r}{2\lg s/\lg\lg s} + (\ell - s/2)\cdot 2\left(\frac{\lg\lg s}{\lg s}\right)^2.
\end{aligned}$$

Note that Equation (1) implies

$$\ell - s/2 < \frac{r\lg s}{9\lg\lg s},$$

and hence

$$\begin{aligned}
x &\leq \alpha(\ell, s) - \frac{r}{2\lg s/\lg\lg s} + \frac{2r}{9\lg s/\lg\lg s} \\
&= \alpha(\ell, s) - \frac{5r}{18\lg s/\lg\lg s} \\
&\leq \alpha(\ell, s) - \frac{5s}{4(\lg s/\lg\lg s)^2} \\
&\leq \alpha(\ell, s) - \frac{4\lg s}{9},
\end{aligned}$$

where the last two inequalities follow from Equation (1) and $s > 2^{16}$, respectively. By using $\max\{s_0, s_1\} \leq s$ we obtain

$$\begin{aligned}
\frac{\nu}{\nu'} &\leq s_0^4 \cdot s_1^4 \cdot 2^{9x} \\
&\leq s_0^4 \cdot s_1^4 \cdot 2^{9\cdot\alpha(\ell,s) - 4\lg s} \\
&\leq s^4 \cdot 2^{4\lg s} \cdot 2^{9\cdot\alpha(\ell,s) - 4\lg s} \\
&= s^4 \cdot 2^{9\cdot\alpha(\ell,s)}.
\end{aligned}$$

□

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

**Proof:** Let $\Delta$ be an $n$-input shuffle-unshuffle network of depth $\ell$, $n = 2^d$. We partition $\Delta$ into $k \leq \lceil \ell/d \rceil$ consecutive shuffle-unshuffle networks $\Delta_i$, $0 \leq i < k$, with depth $\ell_i$ and span $d$. This can be done by defining $\Delta_0$ as the shortest prefix of the levels of the network $\Delta$ with span $d$, $\Delta_1$ as the shortest prefix of $\Delta - \Delta_0$ with span $d$, and so on. (At the end, we may have to add some additional levels to the network in order to get a span of exactly $d$ for $\Delta_{k-1}$. Adding such additional levels to the network can certainly not increase the size of the largest incomparable set.)

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

$$\frac{n}{n'} = \prod_{0 \le i < k} d^4 \cdot 2^{\left(\frac{9(\ell_i - d/2)}{\lg d / \lg\lg d}\right)} \le 2^{\left(\frac{9\ell}{\lg d / \lg\lg d}\right)},$$

for $d$ sufficiently large. Hence, if $\ell < 9 \cdot d \lg d / \lg\lg d$, then $n' > 1$, and it follows that $\Delta$ cannot be a sorting network. $\square$

## 5 Extensions and Limitations

This section discusses a few extensions and limitations of our proof technique.

First, we point out that the proof of our lower bound also holds for certain restricted classes of non-oblivious sorting algorithms on hypercubic machines. More precisely, we can allow our sorting networks 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 $i$th level with elements from $\{+, -, 0, 1\}$ can depend on the outcome of all the comparisons made in all previous levels. Recall that in our lower bound arguments, 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.

Note that this model of a non-oblivious comparator network is quite powerful, and that it allows, for instance, the on-line routing of permutations in logarithmic time (where the permutation to be routed can be an arbitrary function of the outcomes of all previous comparisons in the network). Similarly, we can also show that our lower bounds still hold in the case where a node can hold more than one element, provided that elements cannot be copied. It is unclear whether our techniques can be extended to a model where copying of elements is allowed, or to a model where the sequence of shuffle and unshuffle permutations can be chosen adaptively depending on the outcome of previous comparisons.

We can also extend our lower bounds to some restricted classes of sorting algorithms on multi-dimensional meshes. In [23], Wanka describes the following natural extension of the class of ascend algorithms to multi-dimensional meshes. In an ascend algorithm on a $d$-dimensional mesh of side length $m$, the dimensions are visited in strictly ascending order. Whenever we visit a dimension, we perform $m$ steps of communication across this dimension. Thus, in a single visit to a dimension, we can completely sort the elements in each linear array along that dimension. Note that this class of algorithms corresponds to the class of sorting networks built from $m$-input comparator gates, where consecutive levels of the network are connected by an $m$-way unshuffle permutation (as defined in the register model of a comparator network).

An example of an ascend algorithm on the two-dimensional mesh is the *Shearsort* algorithm [21, 22], which alternatingly sorts along the rows and along the columns. Recently, Corbett and Scherson [5] and Wanka [23] have described two different generalizations of this algorithm to meshes of arbitrary dimension. Both of the algorithms can be implemented as ascend algorithms, and they achieve a running time of $O(d^2 m \lg m)$ on the $d$-dimensional mesh of sidelength $m$.

Using the techniques in this paper, we can show a lower bound of $O(d^2 m \lg m / \lg(dm))$ for the class of ascend sorting algorithms on multi-dimensional meshes (assuming, as before, that the algorithms are comparison-based, and that no copying of elements is allowed). For meshes with nonconstant dimension, this implies that no ascend algorithm can achieve an asymptotically optimal running time. Similarly, we can define natural extensions of the classes of normal algorithms, and normal algorithms with overlap, to multi-dimensional meshes. Using our proof techniques, we can show lower bounds for these classes that improve asymptotically on the distance bound, though we are not aware of any algorithms of this type.

Finally, note that our lower bounds do not apply to probabilistic sorting networks, that is, networks that sort the vast majority of input permutations, but are not sorting networks in the strict sense. In fact, Leighton and Plaxton [14] have designed a shuffle-unshuffle comparator network of depth $O(\lg n)$ that sorts all but a super-polynomially small fraction of the inputs. Similarly, we cannot hope to extend our lower bounds to "randomized" sorting networks, which may contain additional "randomizing" circuit elements that interchange the input values with probability $1/2$, and leave them unchanged otherwise. In [14], Leighton and Plaxton also show how to construct a randomized shuffle-unshuffle network of depth $O(\lg n)$ that sorts every input permutation with high probability.

## 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 shuffle-unshuffle sorting networks. Our techniques also apply 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 that includes the *Sharesort* algorithm of Cypher and Plaxton [8].

Another possible direction for future research would be to consider other restricted classes of sorting networks. As a natural extension of the shuffle-unshuffle networks, we could define the class of *leveled hypercubic networks,* whose structure corresponds to the class of algorithms on the hypercube where in each step communication only occurs across a single dimension, but the sequence of dimensions can be arbitrary. (Note that this class of algorithms cannot be emulated with constant slowdown on any of the bounded-degree variants of the hypercube.) Other classes of interest would be sorting networks based on a single permutation, or periodic sorting networks [9, 10, 12]. Finally, it is an open problem whether our lower bound technique can also be applied to selection networks.

**Acknowledgement**

We would like to thank Marcin Kik 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 686–692, 1992.

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

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

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

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

[7] R. E. Cypher. Theoretical aspects of VLSI pin limitations. *SIAM Journal on Computing*, 22:58–63, 1993.

[8] R. E. Cypher and C. G. Plaxton. Deterministic sorting in nearly logarithmic time on the hypercube and related computers. *Journal of Computer and Systems Sciences*, 47:501–548, 1993.

[9] M. Dowd, Y. Perl, L. Rudolph, and M. Saks. The periodic balanced sorting network. *Journal of the ACM*, 36:738–757, 1989.

[10] 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, 1994.

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

[12] M. Kutyłowski, K. Loryś, B. Oesterdiekhoff, and R. Wanka. Fast and Feasible Periodic Sorting Networks of Constant Depth. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 369-380, 1994.

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

[14] F. T. Leighton and C. G. Plaxton. Hypercubic sorting networks. *SIAM Journal on Computing*, 27(1):1–47, 1998.

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

[16] C. G. Plaxton and T. Suel. Lower bounds for Shellsort. *Journal of Algorithms*, 23:221–240, 1997.

[17] C. G. Plaxton and T. Suel. A lower bound for sorting networks based on the shuffle permutation. *Mathematical Systems Theory*, 27:491–508, 1994.

[18] B. Poonen. The worst case in Shellsort and related algorithms. *Journal of Algorithms*, 15:101–124, 1993.

[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] J. Reif and L. Valiant. A Logarithmic Time Sort for Linear Size Networks. *Journal of the ACM*, 34(1):60–76, 1987.

[21] K. Sado and Y. Igarashi. Some parallel sorts on a mesh-connected processor array. *Journal of Parallel and Distributed Computing*, 3:389–410, 1986.

[22] I. D. Scherson and S. Sen. Parallel sorting in two-dimensional VLSI models of computation. *IEEE Transactions on Computers*, 38:238–249, 1989.

[23] 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.