Randomized Smoothing Networks

Maurice Herlihy
Computer Science Department
Brown University
mph@cs.brown.edu

Srikanta Tirthapura
Dept. of Electrical and Computer Engg.
Iowa State University
snt@iastate.edu

Abstract

A smoothing network *is a distributed data structure that accepts tokens on input wires and routes them to output wires. It ensures that however imbalanced the traffic on input wires, the numbers of tokens emitted on output wires are approximately balanced.*

We study randomized *smoothing networks, whose initial states are chosen at random. Randomized smoothing networks require no global initialization, and also require no global reconfiguration after faults.*

This paper makes the following contributions. We show that the well-known block *smoothing network (which is isomorphic to the* butterfly *network), when started in a random initial state, is $O(\sqrt{\log(w)})$-smooth with high probability, where w is the number of input/output wires. We show that as a corollary, the* bitonic *and* periodic *networks are also $O(\sqrt{\log(w)})$-smooth with high probability, when started in random initial states. In contrast, it is known that these networks are $(\log w)$-smooth in the worst case.*

1. Introduction

A *k-smoothing network* is a distributed data structure that accepts tokens on input wires and routes them to output wires. It ensures that no matter how imbalanced the traffic on input wires, the numbers of tokens emitted on output wires are approximately balanced, lying within k of one another, where k is a constant, independent of the number of active tokens.

Smoothing networks are well-suited for load-balancing applications where tokens represent requests for service. Clients send tokens to arbitrary input wires, and these tokens are routed to servers in such a way that all servers receive approximately the same number of tokens.

In a real distributed system, network switches may be rebooted or replaced dynamically, and it may not be practical to determine the correct initial state for each switch. An attractive approach to fault-tolerance (and maintenance) is simply to initialize the new switch to a random state, eliminating the need for any global coordination. In prior work [7], we showed that certain well-known 1-smoothing networks, when started in an arbitrary initial state (perhaps chosen by an adversary), produce outputs that are at worst $(\log w)$-smooth, where w is the number of input and output wires. This bound is tight for each of the networks that we considered.

In this paper, we show that randomization helps further. If the *block* smoothing network is initialized to a random initial state, and if an off-line adversary (who does not know the initial state) chooses an input sequence, we will show that the output of the network will be $O(\sqrt{\log w})$-smooth with high probability, a significant improvement over the $(\log w)$-smooth worst case. We use the above result to show that the smoothness of a randomly initialized *bitonic* or a *periodic* network is also $O\left(\sqrt{\log w}\right)$.

We conclude with a brief discussion of some informal but suggestive experimental results. By feeding random sequences to randomized networks, we observed a degree of smoothness worse than constant, and consistent with our bound. Randomly initialized networks were dramatically smoother than networks initialized to the default initial state.

1.1. Related Work

Aiello, Venkatesan and Yung [1] build smoothing networks using a different kind of randomized balancer: odd-numbered tokens leave on a randomly chosen wire, and even-numbered tokens leave on the opposite wire from their immediate predecessors. It can be easily verified that our claims still hold if we replace our random balancers with theirs. Their construction employs both deterministic and randomized balancers, and they show that their network has constant output smoothness. In our model, however, an adversary could set the orientations of the deterministic balancers, and their analysis does not hold under these conditions. A comparison of the smoothness properties of various networks appears in Figure 1.

Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04)

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

Network of width w	Depth	Deterministic or Randomized	Global Initialization Required/not	Smoothness
Klugerman and Plaxton [9]	$O(c^{\log^* w} \log w)$	Deterministic	Required	1
Aiello et. al. [1]	$O(\log w)$	Both	Required	2
Block [7]	$\log w$	Deterministic	Not required	$\log w$
Block (this paper)	$\log w$	Randomized	Not required	$O(\sqrt{\log w})$

Figure 1. Comparison of various networks

Czumaj, Kanares, Kutylowski and Lorys [5] analyze the behavior of a butterfly network with random switches for the purpose of generating *random permutations*. They show a $\log^{O(1)} w$ depth network of width w based on pipelined butterflies, which randomly permutes the input sequence so that the output sequence is nearly uniform.

Klugerman and Plaxton [9] show the existence of counting networks (which are 1-smoothing networks with other additional properties) of depth $O(\log w)$ which use deterministic balancers. They also give an explicit construction of a counting network of depth $O(c^{\log^* w} \log w)$ (where c is a constant) using deterministic balancers.

In earlier work [7], we showed that the worst case output smoothness of the Block, Periodic and Bitonic networks of width w is exactly $\log w$ when the switches are initialized adversarially.

2. Balancing Networks

A *balancer* is an asynchronous switch with two input wires and two output wires, labeled "top" and "bottom". A balancer accepts a stream of tokens on its input wires. A balancer has two states: it is either oriented *up* or *down*. If the balancer is oriented *up*, then the next input token leaves on the top output wire, and the balancer becomes oriented *down*. If, however, the balancer is oriented *down*, then the next input token leaves on the bottom output wire, and the balancer becomes oriented *up*.

Definition 1 *A* randomized balancer *is a balancer which has been initialized to a random initial state, i.e.* up *or* down *with equal probability.*

Definition 2 *A* randomized balancing network *is a balancing network whose component balancers are independent randomized balancers.*

Let x denote the total number of input tokens to a randomized balancer b, and y_1, y_2 denote the number of tokens on the top and bottom output wires respectively. By the definition of the randomized balancer, we have

$$\begin{aligned} y_1 &= x/2 + D(x) \cdot r_b \\ y_2 &= x/2 - D(x) \cdot r_b \end{aligned} \quad (1)$$

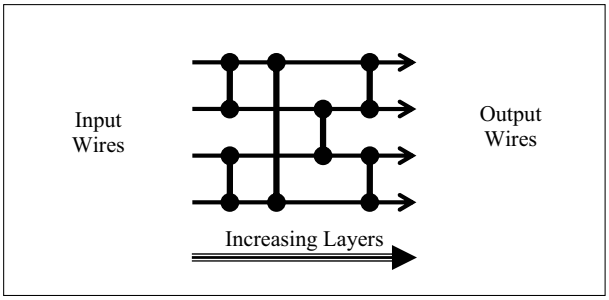

Figure 2. A balancing network of width 4, and depth 3. Horizontal segments are wires and the vertical segments balancers.

where

- r_b is a random variable, specific to balancer b which can take values $+1/2$ or $-1/2$ with equal probability.
- D is the odd-characteristic function, $D(x) = 1$ if x is an odd number, and 0 otherwise.

A *balancing network* is an acyclic network of balancers where output wires of some balancers are linked to input wires of others. An example network is shown in Figure 2.

The network's *input wires* are those wires not linked to the output of any balancer, and similarly for the network's *output wires*. In this paper, we consider balancing networks with the same number of input and output wires, called the network's *width*. Tokens enter the network on the input wires, typically several per wire, propagate asynchronously through the balancers, and leave on the output wires, typically several per wire. A balancing network is *quiescent* if every token that has entered the network has also left. Because balancing networks are acyclic (as directed graphs), each balancer can be assigned a unique *layer*, which is the length of the longest path from an input wire to that balancer.

A balancing network is a k-*smoothing network* if, starting from its initial state, the overall distribution of output tokens across the output wires in any quiescent state is k-smooth: exiting tokens are divided among the output wires

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

in such a way that the difference between the number of tokens output on different wires is never greater than k. In this paper, we will be concerned with the Block network [6] (which is isomorphic to the Butterfly), and the Periodic [6, 3] and the Bitonic [4, 3] networks.

We ask the following question. *How do these networks behave if they are initialized to a random state?* The motivation for this question is as follows. Consider a distributed load-balancing network overlaid on a local area network. If a switch crashes and needs to be reset, or if one switch replaces another, then it is difficult to determine the "right" state for the new switch, and it is not practical to reinitialize the entire network. On the other hand, it is easy to set the switch to a random state. We show the following results:

- The output of a randomized Block network of width w is $(2.83\sqrt{\log w})$-smooth with probability at least $1 - 4/w$.
- We show that the Bitonic network contains a Block network embedded inside it. The Periodic network consists of many pipelined Block networks, hence trivially contains a Block network inside it. Hence, the outputs of Periodic and Bitonic networks of width w are also $(2.83\sqrt{\log w})$-smooth with probability at least $1 - 4/w$.

In contrast, it is known [7] that if we initialize the above networks to *arbitrary* states, they are $(\log w)$-smooth in the worst case.

3. The Periodic and Block Networks

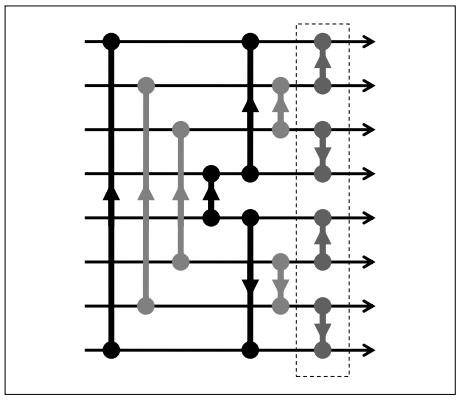

Figure 3. A BLOCK[8] Network, also isomorphic to the Butterfly network.

The PERIODIC[] smoothing network [3] is isomorphic to the *Periodic* sorting network of Dowd, Perl, Rudolph and Saks [6]. At its heart is a component BLOCK$[w]$ network, defined inductively as follows. The BLOCK$[2]$ network is a single balancer. The BLOCK$[2w]$ network is constructed from two BLOCK$[w]$ networks as follows. Given a sequence X, represent each index (subscript) as a binary string. The *A-cochain* of X, denoted X^A, is the subsequence whose indexes have low-order bits 00 or 11. For example, the *A-cochain* of the sequence $x_0, \ldots, x_7$ is x_0, x_3, x_4, x_7. The *B-cochain* x^B is the subsequence whose low-order bits are 01 and 10.

The input sequence X is fed into two parallel BLOCK$[w]$ networks, which we will call the *A-block* and the *B-block*. X^A goes to the A-block, and X^B to the B-block. Their output sequences, call them Y^A and Y^B, are fed into an EVENODD$[2w]$ network, which simply balances each element of Y^A with the corresponding element of Y^B. The PERIODIC$[w]$ network is just $\log w$ BLOCK$[w]$ networks in series. In this paper, however, we focus on a single BLOCK$[w]$ network. Our upper bounds for the Block network directly carry over to the Periodic network. Figure 3 shows a BLOCK$[8]$ network. It can be easily shown that the Block network is isomorphic to the popular Butterfly network, so that our results carry over to the Butterfly too.

3.1. Smoothness of the randomized BLOCK$[w]$ network

BLOCK$[w]$ consists of $\log w$ layers of balancers, numbered from 1 (input layer) to $\log w$ (output layer). The output wires of a balancer in layer ℓ are said to be in layer ℓ. The input wires to the network are said to be in layer 0. Each layer consists of $w/2$ balancers. The balancers in layer i are denoted by ordered pairs. The topmost balancer is $(i, 1)$ and the bottommost balancer $(i, w/2)$.

Consider an execution of the smoothing network, taking it from a random initial state to a quiescent state, where a total of I tokens have entered and left the network. Let $n_{i,j}$ denote the number of tokens entering balancer (i, j) and $r_{i,j}$ the random variable corresponding to that balancer. Let $x_{i,j} = D(n_{i,j})r_{i,j}$. From Equation 1 we know that the number of tokens exiting balancer (i, j) on the top output wire is $n_{i,j}/2 + x_{i,j}$ and the number exiting on the bottom output wire is $n_{i,j}/2 - x_{i,j}$.

We can think of the BLOCK$[w]$ network as a tree. Each node in this tree is a set of one or more balancers.

- The root of the tree is the set of all the $w/2$ balancers at the input layer (layer 1).
- The ith layer is divided into 2^{i-1} nodes, with each node consisting of $w/2^i$ balancers. Nodes at layer i are numbered from $v_{i,1}$ (topmost node in layer i) to $v_{i,2^{i-1}}$ (bottommost). In the tree, there is an edge from node

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

$v_{i,j}$ to nodes $v_{i+1,2j-1}$ and $v_{i+1,2j+1}$, since wires connect balancers located in node $v_{i,j}$ to balancers located in $v_{i+1,2j-1}$ and $v_{i+1,2j+1}$.

- The leaves of the tree are the balancers at the output layer.

Denote the total number of tokens entering node $v_{i,j}$ by $m_{i,j}$. Since the tokens on the top output wires of the balancers in $v_{i,j}$ enter $v_{i+1,2j-1}$, the number of tokens entering node $v_{i+1,2j-1}$ is (using Equation 1):

$$m_{i+1,2j-1} = m_{i,j}/2 + \sum_{(k,l)\in v_{i,j}} x_{k,l} \qquad (2)$$

We will now express the number of tokens that are output at the topmost output wire as a function of the number of input tokens and the random variables corresponding to the different balancers. The tokens that exit from the topmost output wire must follow the path $v_{1,1} \rightarrow v_{2,1} \rightarrow v_{3,1} \rightarrow \ldots \rightarrow v_{\log w,1}$ and further exit on the top output wire of balancer $(\log w, 1)$.

The number of tokens entering $v_{1,1}$ is I, the total number of tokens. Let X_1 denote the number of tokens exiting the topmost output wire. Applying Equation 2 repeatedly, and finally Equation 1, we get:

$$\begin{aligned} m_{2,1} &= I/2 + \textstyle\sum_{j=1}^{w/2} x_{1,j} \\ m_{3,1} &= m_{2,1}/2 + \textstyle\sum_{j=1}^{w/4} x_{2,j} \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ X_1 &= m_{\log w,1}/2 + x_{\log w,1} \end{aligned}$$

Combining the above, we get:

$$X_1 = \frac{I}{w} + \frac{\sum_{j=1}^{w/2} x_{1,j}}{w/2} + \frac{\sum_{j=1}^{w/4} x_{2,j}}{w/4} + \ldots + \frac{\sum_{j=1}^{2} x_{\log w-2,j}}{2} + x_{\log w,1}$$

Rewriting the above: $X_1 = I/w + V$ where

$$V = \sum_{i=1}^{\log w} \sum_{j=1}^{w/2^i} \frac{2^i x_{i,j}}{w} \qquad (3)$$

We will now analyze V. The main problem is that V is not the sum of independent random variables. The various random variables $x_{i,j}$ are heavily dependent on each other, since the fact whether the number of tokens entering balancer (i,j) is even or odd depends upon the decisions taken by balancers at earlier layers.

Recall that $x_{i,j} = r_{i,j} D(n_{i,j})$. We consider an alternate random variable W, defined as follows. W is easier to handle, since it is the sum of independent random variables.

$$W = \sum_{i=1}^{\log w} \sum_{j=1}^{w/2^i} \frac{2^i r_{i,j}}{w} \qquad (4)$$

We now prove a key lemma, showing that in order to bound the probability that V deviates significantly from its expected value, it is enough to bound the probability that W deviates from its expected value.

Lemma 1 *For any* $\delta > 0$, $\mathbf{Pr}\{|V| > \delta\} \leq 2\mathbf{Pr}\{|W| > \delta\}$.

Proof: Consider the conditional probability

$$\alpha = \mathbf{Pr}\{W > \delta | V > \delta\}$$

V and W can be written as:

$$V = \sum_{\{((i,j)\in S)\wedge(D(n_{i,j})=1)\}} c_{i,j} r_{i,j}$$

$$W = \sum_{\{(i,j)\in S\}} c_{i,j} r_{i,j}$$

for some set S of balancers, and constants $c_{i,j}$, which can be derived from equations 3 and 4 (the exact nature of S and the $c_{i,j}$'s are not important for this proof). From the above, $V > \delta$ and $W \leq \delta$ can happen only if

$$\sum_{\{((i,j)\in S)\wedge(D(n_{i,j})\neq 1)\}} c_{i,j} r_{i,j} < 0$$

This can happen with probability at most 1/2, since for each balancer (i,j), the random variable $r_{i,j}$ is distributed symmetrically about zero, and is chosen independent of $n_{i,j}$. Thus, $\alpha \geq 1/2$, and we get

$$\begin{aligned} \mathbf{Pr}\{W > \delta\} &\geq \mathbf{Pr}\{W > \delta | V > \delta\} \cdot \mathbf{Pr}\{V > \delta\} \\ &\geq \mathbf{Pr}\{V > \delta\}/2 \end{aligned}$$

Similarly, we can show the other direction, that $\mathbf{Pr}\{V < -\delta\} \leq 2\mathbf{Pr}\{W < -\delta\}$, and the lemma follows.

Lemma 2

$$\mathbf{Pr}\left\{|V| > \sqrt{2\log w}\right\} < \frac{4}{w^2}$$

Proof: We will first bound the probability that $|W|$ is too large, and then use Lemma 1 to bound the probability that $|V|$ is too large. Since W is the sum of independent random variables, we use Hoeffding's inequality (stated in Lemma 3) to bound W. In our case, W is the sum of $(w/2 + w/4 + \ldots + 1) = w - 1$ independent random variables.

First, the expectation of W: $E[W] = 0$, since $E[r_{i,j}] = 0$ for each i, j.

Next, the ranges of various random variables.

$$\frac{2^i r_{i,j}}{w} \in \left[\frac{-2^{i-1}}{w}, \frac{+2^{i-1}}{w}\right]$$

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

The term $\sum (b_i - a_i)^2$ in our case is:

$$\frac{w}{2}(2/w)^2 + \frac{w}{4}(4/w)^2 + \ldots + 1 = 2 - 2/w$$

Setting $\epsilon = \frac{\sqrt{2\log w}}{w-1}$ in Lemma 3, we get

$$\mathbf{Pr}\left\{W > \sqrt{2\log w}\right\} \leq \exp\{-2\log w\} \leq 2^{-2\log w} \leq 1/w^2 \quad (5)$$

Since W is the weighted sum of $r_{i,j}$'s, each of which is symmetrically distributed about zero, W is also symmetrically distributed about zero. Hence

$$\mathbf{Pr}\left\{W < -\sqrt{2\log w}\right\} = \mathbf{Pr}\left\{W > \sqrt{2\log w}\right\} \leq 1/w^2 \quad (6)$$

Equations 5 and 6 combined with Lemma 1 yield the proof of the lemma.

Lemma 3 *[Hoeffding's bound [8]]*
Suppose $S_n = Y_0 + Y_1 + \ldots Y_{n-1}$ where the Y_i's are independent random variables, and for each i, $Y_i \in [a_i, b_i]$. Then, for any $\epsilon > 0$,

$$\mathbf{Pr}\{S_n - E[S_n] > n\epsilon\} \leq \exp\left\{\frac{-2n^2\epsilon^2}{\sum_{i=0}^{n-1}(b_i - a_i)^2}\right\}$$

Theorem 1 *The output sequence of* BLOCK$[w]$ *with randomized balancers is $2.83\sqrt{\log w}$-smooth with probability at least $1 - 4/w$.*

Proof: Let $X_1, X_2, \ldots, X_w$ denote the output sequence of BLOCK$[w]$ with randomized balancers, where X_1 corresponds to the topmost output wire.

From Lemma 2, we have

$$\mathbf{Pr}\left\{|X_1 - I/w| > \sqrt{2\log w}\right\} \leq 4/w^2$$

The random variables $X_1 \ldots X_w$ all have the same distribution, due to symmetry, and a similar equation holds for all of them.

Using the union bound on probabilities, the probability that for some i the value $|X_i - I/w|$ exceeds $\sqrt{2\log w}$ is not greater than $4/w$. Thus, with probability at least $1 - 4/w$, all outputs X_i are within $\sqrt{2\log w}$ of I/w, and hence within $2\sqrt{2\log w}$ of each other, and the theorem follows.

4. The Bitonic Network

The BITONIC$[w]$ counting network [3] is isomorphic to the *Bitonic* sorting network of Batcher [4]. This network has a simple inductive structure, shown in Figure 4. The BITONIC$[2]$ network is a single balancer. The BITONIC$[2w]$ network is constructed by feeding the $2w$ input wires into two parallel BITONIC$[w]$ networks, and feeding their outputs into a MERGER$[2w]$ network.

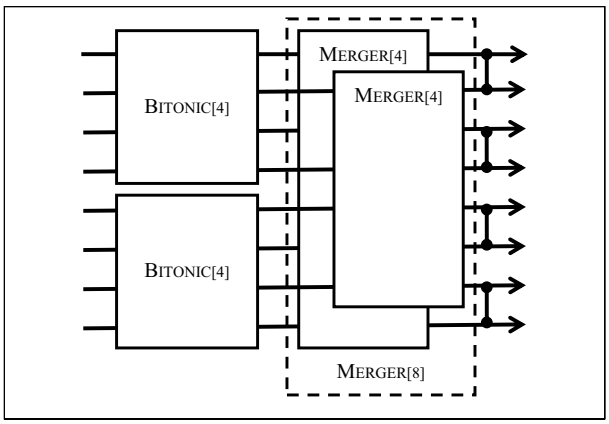

Figure 4. Recursive Structure of a BITONIC$[8]$ **Counting Network.**

The MERGER$[2w]$ balancing network takes two input w-sequences X and Y and produces an output $2w$-sequence Z. The MERGER$[w]$ network is also defined inductively. The MERGER$[2]$ network is a single balancer. We construct the MERGER$[2w]$ network from two MERGER$[w]$ networks and a EVENODD$[2w]$ network to be described. Let X^E denote the even subsequence $x_0, x_2, \ldots, x_{w-2}$ of X, and X^O denote the odd subsequence $x_1, x_3, \ldots, x_{w-1}$. Similarly define Y^E and Y^O.

The input to the first MERGER$[w]$ network is the w-sequence formed by the concatenation of X^E and Y^O, denoted by $X^E \cdot Y^O$. Call that network's output sequence U. Symmetrically, the input to the second MERGER$[w]$ network is the w-sequence $X^O \cdot Y^E$. Call that output sequence V. The final layer of the network, called EVENODD$[w]$, simply joins the each output wire of the first MERGER$[w]$ network with the corresponding output wire of the second MERGER$[w]$ network.

Two networks $\mathcal{N}$ and $\mathcal{M}$ are *isomorphic* if the underlying directed graphs are isomorphic.

Lemma 4 *The* MERGER$[w]$ *network is isomorphic to the* BLOCK$[w]$ *network.*

Proof: We argue by induction on w. When w is 1, both networks consist of a single balancer. Assume the claim for the MERGER$[w]$ network, and consider the MERGER$[2w]$ network. In the BLOCK$[2w]$ network, the final layer connects the i-th wire of one component BLOCK$[w]$ network with the i-th wire of the other. Likewise, in the MERGER$[2w]$ network, the final layer connects the i-th wire of one component BLOCK$[w]$ network with the i-th wire of the other, preserving the isomorphism.

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

We note that additional layers of balancers can never increase the smoothness of a sequence, and can only decrease it. The above lemma and Theorem 1 lead to the following corollary.

Corollary 1 *The* MERGER$[w]$ *network, and hence, the* BITONIC$[w]$ *network are* $2\sqrt{2\log w}$*-smooth with probability at least* $1 - 4/w$.

5. Experiments

To develop an intuition whether our bound can be improved, we conducted a number of simple experiments testing the behavior of randomized Block networks of different sizes. These results, of course, do not prove anything, but they are suggestive. The results are shown in Figure 5.

In our experiments, the number of tokens input into each wire is randomly chosen between 1 and 100000, and the output smoothness is averaged over 10 runs. We simulated networks of width up to 2^{24}. The results show that when $\log w$ changes from 1 to 24, the (average) output smoothness increases very gradually, from 0.7 to 3.2. We do not expect to be able to simulate much larger networks.

To compare with the randomized network, we simulated a Block network whose balancers were all initialized to *up*. Using the same setup as for the random network, (the number of tokens into each wire is a random number between 1 and 100000, average taken over 10 runs), the results obtained are shown below. It can be seen that the output smoothness is very nearly equal to $\frac{\log w}{2}$.

While we know that the worst case smoothness of a Block network initialized by an adversary is $\log w$, these experiments suggest that the smoothness of the network over random inputs when initialized very regularly (not by an adversary) is no better than $\log w/2$. Moreover, for applications needing approximate load balancing, a randomized Block network would work much better than a deterministic one in the *average* case.

6. Open Problems

- Our bounds for the smoothness of these networks do not make use of structure that may be present in the input sequence. *Can we obtain better bounds if the input is already fairly smooth?*
- A related question is: *Can we get better bounds on the output smoothness of the randomized Periodic or Bitonic networks?*
- *How tight is the* $O(\sqrt{\log n})$ *upper bound for the Block network? Can we get a matching lower bound?*

References

[1] W. Aiello, R. Venkatesan, and M. Yung. Coins, weights and contention in balancing networks. In *Proceedings of the annual ACM symposium on Principles of Distributed Computing*, pages 193–205, August 1994.

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

[3] J. Aspnes, M. Herlihy, and N. Shavit. Counting networks. *Journal of the ACM*, 41(5):1020–1048, 1994.

[4] K. Batcher. Sorting networks and their applications. In *Proceedings of the AFIPS Spring Joint Computer Conference*, volume 32, pages 338–334, 1968.

[5] A. Czumaj, P. Kanarek, M. Kutylowski, and K. Lorys. Switching networks for generating random permutations. In D.-Z. Du and H. Ngo, editors, *Switching Networks: Recent Advances*. Kluwer Academic Publishers, 2001.

[6] M. Dowd, Y. Perl, L. Rudolph, and M. Saks. The periodic balanced sorting network. *Journal of the ACM*, 36(4):738–757, October 1989.

[7] M. Herlihy and S. Tirthapura. Self stabilizing smoothing and counting. In *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS)*, pages 4–11, 2003.

[8] W. Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Association Journal*, 58:13–30, 1963.

[9] M.R.Klugerman and C.G.Plaxton. Small-depth counting networks. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 417–428, 1992.

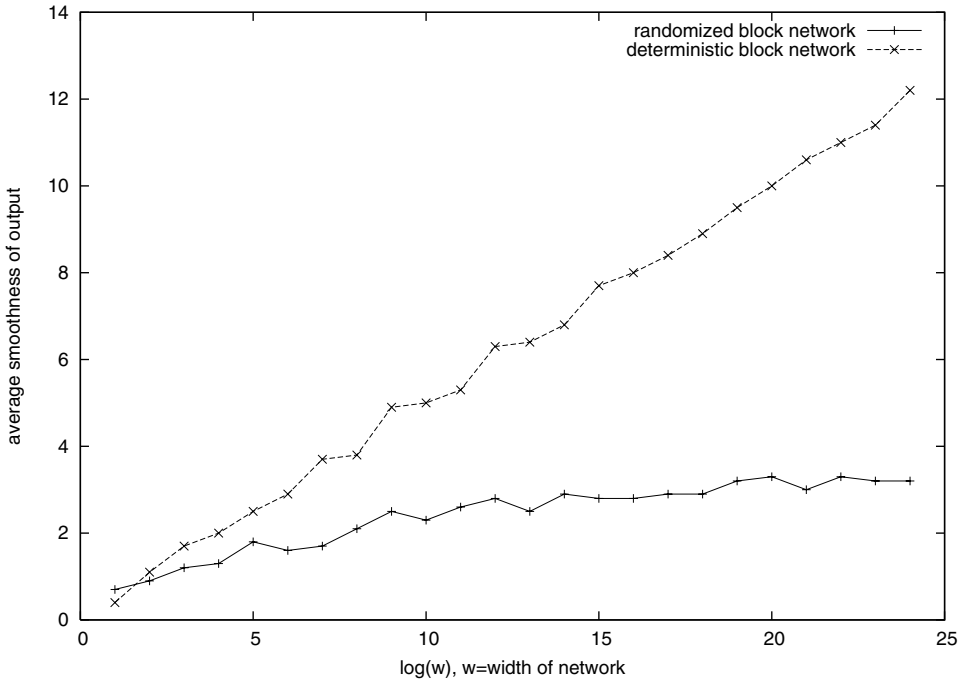

Figure 5. The observed smoothness of a $\textsc{Block}[w]$ network with randomized and deterministic balancers

0-7695-2132-0/04/$17.00 (C) 2004 IEEE