Asymptotically Tight Bounds for Computing with Faulty Arrays of Processors
(Extended Abstract)

C. Kaklamanis* A. R. Karlin† F. T. Leighton‡ V. Milenkovic § P. Raghavan¶
S. Rao ‖ C. Thomborson** A. Tsantilas ††

Summary

In the paper, we analyze the computational power of 2 and 3-dimensional processor arrays that contain a potentially large number of faults. We consider both a random a and worst-case fault model, and we prove that in either scenario, low-dimensional arrays are surprisingly fault-tolerant. For example, we show how to emulate an $n\sqrt{\log n} \times n\sqrt{\log n}$ fault-free array on an $n \times n$ array containing $\Theta(n^2)$ random faults with slowdown $O(\log n)$, the same slowdown that is used by a fault-free $n \times n$ array to perform the simulation. We also show how to route, sort, and perform systolic algorithms for problems such as matrix multiplication in optimal time on faulty arrays. In many cases, the running time is the same as if there were no faults in the array (up to constant factors). On the negative side, we show that any constant congestion embedding of an $n \times n$ fault-free array on an $n \times n$ array with $\Theta(n^2)$ random faults (or $\Theta(\log n)$ worst-case faults) requires dilation $\Theta(\log n)$. For 3-d arrays, we use knot theory to prove that the required dilation is $\Omega(\sqrt{\log n})$.

*Aiken Computation Laboratory, Harvard University, Cambridge, MA 02138. Supp. by NSF Grant NSF-CCR-87-04513

†DEC Systems Research Center, Palo Alto, CA 94301.

‡Mathematics Department and Laboratory for Computer Science, MIT, Cambridge, MA 02139. Research supported by the Defense Advanced Research Projects Agency under Contract N00014-87-K-825, the Office of Naval Research under Contract N00014-86-K-0593, the Air Force under Contract OSR-89-0271, and the Army under Contract DAAL-03-86-K-0171.

§Aiken Computation Laboratory, Harvard University, Cambridge, MA 02138.

¶IBM T.J. Watson Research Center, Yorktown Heights, NY 10598.

‖Aiken Computation Laboratory, Harvard University, Cambridge, MA 02138. Supp. by ONR Grant # N0014-88-K-0243.

**Computer Science Department, University of Minnesota, Duluth, MN 55812. Research supported by the National Science Foundation, through its Design, Tools and Test Program under grant number MIP 8706139.

††Aiken Computation Laboratory, Harvard University, Cambridge, MA 01238. Research supported in part by NSF Grants NSF-DCR-86-00379 and NSF-CCR-89-02500.

1 Introduction

In this paper, we study the problem of computing with processor arrays that contain a potentially large number of faulty processors. We consider a very strict model of processor failure; if a processor fails, it can neither compute nor communicate with its neighbors. (Messages cannot be routed through a faulty processor.) We consider both random and worst-case fault models. In the random model, we assume that each processor fails independently with some probability $p < p'$ where p' is a small constant. We refer to an array with such failures as a *p-faulty array*. In the worst-case model, we assume that an adversary chooses $k < n$ processors to fail.

The focus of our research is to devise algorithms for faulty arrays that have nearly the same performance as the best algorithms for fault-free arrays. Somewhat surprisingly, we show that this is possible for many problems, even if there is a relatively large number of random or worst-case faults in the array. For example, we show that almost every $n \times n$ p-faulty array (and any array with at most $n/3$ worst-case faults) can sort n^2 packets or multiply $n \times n$ matrices in $O(n)$ steps, the same time as it takes to do these operations on a fault-free $n \times n$ array (up to constant factors). We also show that almost all p-faulty arrays can route n^2 packets in $O(n)$ steps. In fact, we can route packets between any pair of processors for which there is a connection via a path of live processors in the network. This substantially generalizes the result of Raghavan [13], who devised a randomized routing algorithm that runs in $O(n \log n)$ steps with queues of size $O(\log n)$. In contrast, our algorithm is deterministic, runs in $O(n)$ steps, uses queues of size $O(1)$, and can support combining. Similar results hold for an array with $O(n)$ worst-case faults.

More generally, we consider the problem of simulating a fault-free $m \times m$ array on a faulty $n \times n$ array. Once again, we find that arrays are surprisingly resilient to faults. For example, we show that almost every p-faulty $n \times n$ array can simulate an $n\sqrt{\log n} \times n\sqrt{\log n}$ fault-free array with $O(\log n)$ slowdown. The result demonstrates another application

CH2925-6/90/0000/0285$01.00 © 1990 IEEE

for which p-faulty arrays are no worse than fault-free arrays. The result is proved by embedding a fault-free $n \times n$ array into a faulty $n \times n$ array with constant load and congestion and optimal dilation $\Theta(\log n)$.

We also consider the problem of computing on faulty 3-d arrays. As with 2-d arrays, we find that faulty 3-d arrays have much the same power as fault-free 3-d arrays, although the results are more difficult and the bounds are slightly weaker. A highlight of our work on 3-d arrays is the proof that any $O(1)$-to-1 embedding of a fault-free $n \times n \times n$ array into most any p-faulty $n \times n \times n$ array requires dilation $\Omega(\sqrt{\log_{1/p} n})$. The proof makes use of knot theory, and provides the first lower bound for reconfiguring arrays in higher dimensions. (The corresponding lower bounds for 2-dimensional arrays follow rather straightforwardly from earlier work on wafer-scale integration [5, 8, 9] that makes use of winding numbers.)

In summary, the results in this paper reveal that low-dimensional arrays are highly resilient to large numbers of randomly occurring faults as well as lesser numbers of worst-case faults. Although it was known that high bandwidth networks such as the hypercube [6] and multibutterfly [11] have strong fault-tolerance properties, relatively few such results were known for low-dimensional arrays [5, 8, 9]. In fact, we do find that low-dimensional arrays are somewhat less resilient to faults than hypercubes (e.g., a fault-free hypercube can be embedded into a p-faulty hypercube with constant dilation and congestion for any $p < 1$, but the corresponding result for 2-d arrays only works for small constant p, and only with $\Theta(\log n)$ dilation), but the differences are surprisingly small.

The remainder of the paper is divided into sections as follows. The results for 2-dimensional arrays are contained in Section 2. The upper bounds and algorithms for 3-dimensional arrays are contained in Section 3. Section 4 contains the lower bound on dilation for any embedding of fault-free 3-d array into a p-faulty array of the same size. Because of space limitations, we have omitted several details and some proofs from this extended abstract.

2 Algorithms for 2-d Arrays

Nearly all of our algorithms and upper bounds for 2-d arrays are based on a property that measures how much a grid with faults differs locally from a fault-free grid. In particular, we say that a faulty $n \times n$ array is (α, r)-*gridlike* ($0 \leq \alpha < 1, 0 < r \leq n$) if for every $r \times r$ subarray, there are at least $(1-\alpha)r$ fault-free paths connecting the left and right sides of the subarray and at least $(1-\alpha)r$ fault-free paths connecting the top and bottom of the subarray. Moreover, we require that the paths all have length at most $2r$, and that the horizontal (respectively vertical) paths be vertex-disjoint. Since we will mostly be concerned with the case when $\alpha = 1/3$, we also define an array to be *r-gridlike* if it is (α, r)-gridlike for $\alpha = 1/3$.

In Subsection 2.1, we will show that almost all p-faulty $n \times n$ arrays are $O(\log_{1/p} n)$-gridlike, and that all $n \times n$ arrays with $k \leq n/3$ faults are $O(k)$-gridlike. We will then show in later subsections that r-gridlike arrays can perform many of the same tasks as fault-free arrays (and in about the same time) provided that r is not too big. In particular, we will show in Subsection 2.2 how to embed an $n \times n$ fault-free array in an $n \times n$ r-gridlike array with load $O(1)$, congestion $O(1)$ and dilation $O(r)$. As a result, we will show how to simulate any $m \times m$ array on an $n \times n$ r-gridlike array with slowdown $O(r + \frac{m^2}{n^2})$. For $m = \Omega(\sqrt{r}n)$, this is the same as the slowdown required to simulate an $m \times m$ array on a fault-free $n \times n$ array, which means that the faulty array performs as well as a fault-free array (up to constant factors) for this application. If $m = n$, then we experience a slowdown of $O(r)$ for the simulation. When applied to p-faulty arrays, and arrays with k worst-case faults, the construction gives optimal results.

Many problems can be solved very efficiently on r-gridlike arrays. For example, in Subsection 2.3, we show how to sort and multiply matrices in $O(n)$ steps on an r-gridlike array ($r \leq n$). We also show how to route in $O(n + r^2)$ steps. When applied to p-faulty arrays or arrays with k worst-case faults, these algorithms again give optimal performance.

It is worth noting that although we will use probabilistic methods to prove that a random p-faulty array is $O(\log_{1/p} n)$-gridlike with high probability, all of the algorithms for r-gridlike arrays are deterministic. In particular, when we show how to sort in $O(n)$ steps on a p-faulty $n \times n$ array with high probability, the sorting algorithm is deterministic and guaranteed to work provided that the array is $O(\log_{1/p} n)$-gridlike. At no time do we use the randomness inherent in the fault pattern as a basis for pseudorandom coins in the algorithms.

2.1 Showing that Faulty Arrays are Gridlike

In what follows, we show that a worst case array with k faults is $O(k)$-gridlike, and that a p-faulty array is $O(\log_{1/p} n)$-gridlike with high probability. The result for worst-case faults is particularly straightforward.

Lemma 1 *An $n \times n$-array with $k < n$ faults is $(\alpha, k/\alpha)$-gridlike for any $k/n \leq \alpha < 1$.*

Proof. Consider any $k/\alpha \times k/\alpha$ subarray. (We know that $k/\alpha \leq n$ since $\alpha \geq k/n$.) Since there are at most k faults, the subarray has at least $k/\alpha - k = (1-\alpha)k/\alpha$ fault-free rows and columns. □

Theorem 2 *An $n \times n$ array with $k \leq n/3$ faults is $3k$-gridlike.*

Proof. Use $\alpha = 1/3$ in Lemma 1. □

The result for p-faulty arrays is a little more interesting, and makes use of the following simple pigeonhole argument.

Lemma 3 *If there are $(1-\alpha/2)r$ node-disjoint paths from one side of an $r \times r$ array to the opposite side, then at least $(1-\alpha)r$ of the paths have length at most $2r$.*

Proof. If more than $\frac{\alpha}{2}r$ of the $(1-\frac{\alpha}{2})r$ paths have length $2r$ or more, then the total number of nodes in all the paths would be more than $(1-\alpha)r^2 + \frac{\alpha}{2}r(2r) = r^2$. This is not possible since there are only r^2 nodes in an $r \times r$ array. □

Lemma 4 *Given any constant $\alpha > 0$, there is a constant $p_\alpha > 0$ such that a p-faulty $n \times n$ array is (α, r)-gridlike with probability at least $1 - 1/n$, where $r = O(\log_{1/p} n)$ and $p \leq p_\alpha$.*

Proof. If the array is not (α, r)-gridlike, then by Lemma 3, we know that it contains an $r \times r$ subarray R for which we cannot construct $(1-\frac{\alpha}{2})r$ node-disjoint paths from one side to the other. By Menger's Theorem [3] this means that there is a set of fewer than $(1-\frac{\alpha}{2})r$ live nodes whose removal from R (along with the faulty nodes) disconnects one side of R from the opposite side. Any set of nodes whose removal from an array disconnects the left side from the right side (say) contains a subset of nodes that form a simple path from the top to the bottom of the array, allowing 45 degree connections. [1]

Hence, R must contain a simple path from one side of the subarray to the opposite side which contains fewer than $(1-\frac{\alpha}{2})r$ live nodes. The number of such paths of length $l (l \geq r)$ is at most $2r7^{l-1} < r7^l$, since there are $2r$ places where we can start or end the path and at most 7 ways to continue at each step. By only considering minimal paths, this quantity can be reduced to $r5^l$ since we can rule out the possibility of extending to a neighbor of the previous node in the path. Given a particular path of length l, the probability that it has fewer than $x = (1-\frac{\alpha}{2})r$ live nodes is at most $\binom{l}{x}p^{l-x}$.

Combining the previous arguments, we find that the probability that there exists such a bad subarray R is at most

$$n^2 \sum_{l \geq r} r5^l \binom{l}{x} p^{l-x} \leq \frac{n^2 r e^x}{p^x x^x} \sum_{l \geq r} (5p)^l l^x$$

$$\leq O\left(\frac{n^2 r e^x (5p)^r r^x}{p^x x^x}\right)$$

provided that $(1-\alpha/2) < \ln(\frac{1}{5p})$. Substituting $x = (1-\alpha/2)r$, and simplifying, we find that this probability is at most

$$O\left(n^2 r \left[\frac{e^{1-\frac{\alpha}{2}} 5p^{\alpha/2}}{(1-\alpha/2)^{1-\alpha/2}}\right]^r\right).$$

For any constant $\alpha > 0$, there is a constant $p_\alpha > 0$ such that for $p \leq p_\alpha$ and $r \geq O(\log n / \log(1/p))$, the expression above is at most $1/n$. Hence, for any constant $\alpha > 0$ and $p \leq p_\alpha$, a p-faulty array is (α, r)-gridlike with probability at least $1 - 1/n$, where $r = O(\log_{1/p} n)$. □

Theorem 5 *There is a constant $p' > 0$ such that for any $p \leq p'$, a p-faulty $n \times n$ array is $O(\log_{1/p} n)$-gridlike with high probability.*

Proof. Apply Lemma 4 with $\alpha = 1/3$. □

2.2 Simulating a Fault-Free Array with a Gridlike Array

In what follows, we describe an efficient embedding of an $n \times n$ fault-free array in an r-gridlike faulty array. Nodes of the fault-free array are mapped $O(1)$-to-1 to live nodes of the faulty array, and edges of the fault-free array are mapped to live paths of the faulty array. We then apply this result to show how faulty arrays can simulate fault-free arrays with relatively little degradation in performance.

Note that, in general, when a guest graph G is embedded in a host graph H, we are interested in three properties: (1) *load*: the maximum number of nodes of G mapped on some node of H; (2) *congestion*: the maximum number of edges of G whose mappings pass through an edge of H; and (3) *dilation*: the maximum length of a path of H, that is the image of some edge of G.

Theorem 6 *An $n \times n$ array can be embedded in an $n \times n$ r-gridlike array with $O(r)$ dilation, $O(1)$ load, and $O(1)$ congestion.*

Proof. We first embed an $n/3 \times n/3$ array, and then adjust the load and congestion by a factor of 9 to obtain our $n \times n$ array. Without loss of generality, we will assume that $(r-1)$ divides $(n+1)$. Then we can evenly divide up the faulty array into edge-disjoint but abutting blocks of $r \times r$ nodes, so that adjacent blocks are bridged by a row or column of r edges. Since $2r/3$ of the nodes on each boundary of a block are connected by vertex-disjoint paths to

[1] For a proof of a generalization of this fact, see [12]

the opposing boundary, there are at least $r/3$ vertex-disjoint paths across any *pair* of contiguous blocks. We call these paths *highway segments*. Each highway segment spans $2r-1$ rows or columns and has length at most $4r$.

By connecting the highway segments in adjacent pairs of blocks, we will build *full fledged highways* that span the entire faulty array. In particular, we will construct $r/3$ node-disjoint live paths of length $O(n)$ that go from one side of the faulty array to the other through each row and column of $r \times r$ blocks. We construct the highways from the highway segments by using an *interchange* in each $r \times r$ block along the way. The interchange is used to interconnect the $r/3$ highway segments in one pair of blocks to the $r/3$ highway segments in the next (overlapping) pair of blocks. An interchange can be made in the middle block of each set of three collinear blocks, using its r-gridlike property in the horizontal and vertical directions to define a $2r/3 \times 2r/3$ crossbar. In particular, assume the three blocks are horizontally adjacent; the vertical case is analogous. The highway segments entering the middle block across its left edge will intersect every vertical path through the crossbar; thus any traffic on the ith entering highway can be switched onto the ith vertical path in the crossbar, routed to the ith horizontal path in the crossbar, then to the $(2r/3-i)$th vertical path in the crossbar, and then out on the ith right-side highway segment. Note that our completed highways are vertex-disjoint, they span the entire array, and they are of length $O(n)$. The last step in the construction is to define the (j,k)th node of an $n/3 \times n/3$ virtual grid as the processor at the first intersection of the jth horizontal highway with the kth vertical highway.

This induces an embedding of an $n/3 \times n/3$ array in the r-gridlike array with congestion 2, load 1, and dilation $O(r)$. The dilation bound follows from the fact that the length of any highway within an $r \times r$ block is at most $O(r)$. □

Theorem 7 *With probability $1-O(1/n)$, a dilation of $\Theta(\log_{1/p} n)$ is necessary and sufficient for an $O(1)$-load and $O(1)$-congestion embedding of a $\Theta(n) \times \Theta(n)$ array in a p-faulty array, for sufficiently small p.*

Proof sketch. The upper bound is immediate from Theorems 5 and 6. The lower bound is obtained by techniques similar to those in previously-published work on wafer-scale integration [5, 8, 9]. The details will appear in the final version of the paper. □

Theorem 8 *A dilation of $\Theta(k)$ is necessary and sufficient for an $O(1)$-load and $O(1)$-congestion embedding of an $\Omega(n) \times \Omega(n)$ virtual grid in an $n \times n$ array with k faults, $k < (1-\epsilon)n$, $\epsilon > 0$.*

Proof sketch. The upper bound follows from Theorems 2 and 6. The lower bound is derived by arguments similar to those in [8]. Dead regions of area $\Theta(\log_{1/p} n)$ in the proof of [8] are replaced by dead strips of length $\Theta(k)$. A constant number of such strips are distributed throughout the array, one per $\Theta(n) \times \Theta(n)$ square. The rest of the proof is identical to that of Theorem 7. □

Theorem 9 *An r-gridlike, $n \times n$ array can simulate a fault-free $m \times m$ array with an $O(r + m^2/n^2)$ factor slowdown.*

Proof. Partition the $m \times m$ array into blocks of size $3m/n \times 3m/n$. The i,jth $3m/n \times 3m/n$ block will be simulated by the i,jth node in the $n/3 \times n/3$ virtual array described in Theorem 6. This node can simulate the computation of the $3m/n \times 3m/n$ block with $9m^2/n^2$ slowdown. It can simulate the communication of the block by sending $3m/n$ packets of information to each of its neighbors for each step of the $m \times m$ array. By Theorem 6, we know that all the packets can be routed in $O(r+m/n)$ steps by pipelining along the highways. Hence, the slowdown of the simulation is $O(r + m^2/n^2)$, as claimed. □

Theorem 10 *With probability 1 - 1/n, a p-faulty $n \times n$ array can simulate a fault-free $n\sqrt{\log_{1/p} n} \times n\sqrt{\log_{1/p} n}$ array with $O(\log_{1/p} n)$ slowdown.*

Proof. Immediate from Theorems 5 and 9 for any small constant p. □

The result in Theorem 10 is the best possible up to constant factors (since the load is $\log n$), and shows that a p-faulty array has as much capability as a fault-free array when simulating a slightly larger array.

2.3 Optimal Algorithms for Sorting, Routing, and Matrix Multiplication

In what follows, we show how to route, sort and multiply matrices in $O(n)$ steps on an r-gridlike $n \times n$ array for small r. For sorting and matrix multiplication we assume that the data is input and output at the virtual gridpoints of the array. The sorting result makes use of the following simple fact.

Fact 11 *[1] Any fixed permutation of n^2 elements can be performed on a (fault-free) $n \times n$ array of processors, by a three-stage method: a row routing, a column routing, then a row routing. By a row(column) routing, we mean a rearrangement of the items in each row(column).*

Theorem 12 *An r-gridlike $n \times n$ ($r \leq n$) array of processors, each with $O(1)$ storage, can sort $O(n^2)$ elements in $O(n)$ deterministic time.*

Proof. We adapt Columnsort [7] for this purpose. Columnsort sorts an $r \times s$ matrix ($r \geq s^2$) in 8 phases. Phases 2,4,6 and 8 consist of a fixed permutation of the matrix, and phases 1,3,5 and 7 consist of sorting the columns of the matrix. By using a second iteration of columnsort to sort the columns, we can devise an $O(1)$-phase algorithm to sort an $n \times n$ matrix where each phase consists of applying a fixed permutation to the matrix or of sorting the columns.

By Fact 11, this means that we can reduce the problem of sorting n^2 items to $O(1)$ routing and sorting operations in the rows and columns of an $n \times n$ array. Each row and column operation can be accomplished in $O(n)$ steps using the highways constructed for the embedding of the fault-free array in the faulty array. Although these highways have dilation $O(r)$, their total length is $O(n)$. Hence, we can treat the highways as $O(n)$-cell linear arrays, on which routing and sorting takes $O(n)$ steps. (In the case of sorting, we first pack the items to be sorted in the leftmost n cells of the linear array, whereupon we use bubblesort.) □

Corollary 13 *With probability $1-1/n$, a p-faulty $n \times n$ array can sort $O(n^2)$ elements in $O(n)$ deterministic time.*

Proof. Immediate from Theorems 5 and 12 for any small constant p. □

Since on-line packet routing can be reduced to sorting [2, 10], we can immediately obtain the following on-line routing result.

Theorem 14 *An r-gridlike $n \times n$ ($r \leq n$) array of processors, each with $O(1)$ storage can solve any routing problem among its virtual grid points on-line in $O(n)$ deterministic time.*

Proof. Follows directly from Theorem 12 and the reductions in [2, 10]. □

Note that Theorem 14 is stronger than the result in Fact 11, since the routing problem needed to be known in advance for Fact 11 to apply. The following result is stronger still, in that it allows packets to start and end at nodes of the faulty array that are not virtual grid points. The time bound is weaker for $r \geq \sqrt{n}$, however.

Theorem 15 *An r-gridlike $n \times n$ array with $O(1)$ storage per processor can perform any on-line routing of packets between live processors connected by live paths in $O(n + r^2)$ deterministic time.*

Proof. The vertical and horizontal highways partition the faulty array up into *regions*, consisting of processors which lie between two adjacent vertical and horizontal highways. By construction, each of these regions has size at most $O(r^2)$. If a processor wishes to route a packet to a processor in a different region, and those processors are connected by live paths, then there must be a path from the source processor to one of the surrounding vertical or horizontal highways and there must be a path from one of the vertical or horizontal highways to the destination processor.

Consider a partition of the array into $r \times r$ blocks as before. In an $O(r^2)$ step preprocessing phase, we will construct spanning trees for all connected components within each such block, including all regions which intersect each block (i.e. all four regions surrounding a virtual gridpoint are included in the spanning trees for the block containing that virtual gridpoint.) Each spanning tree has $O(r^2)$ processors, and if it contains any virtual gridpoint, it contains all $4r^2/9$ virtual gridpoints. These virtual gridpoints can be ordered in a well defined manner based on which highway intersections they correspond to. Furthermore, if we order all r^2 processors (both live and faulty ones) in each block in a canonical way, we can partition and associate them canonically and $O(1)$-to-1 to the virtual gridpoints of the block.

The routing process then consists of three steps: packets destined for a different region (and connected to their destination) are moved along an $O(r^2)$-length Hamiltonian tour of their spanning tree to their associated virtual gridpoint; thus in $O(r^2)$ time $O(1)$ packets are delivered to each virtual gridpoint in the source block. Packets then are routed in $O(n)$ time to the virtual gridpoint associated with their destination; the routing is done by a well-known reduction to sorting. Finally packets are moved in $O(r^2)$ time on the spanning tree to their destinations within the destination block. Packets not destined for a different block can be routed in the local spanning tree. □

Theorem 16 *With probability $1-1/n$, a p-faulty $n \times n$ array can route packets among all live processors connected by live paths in $O(n)$ time.*

Proof. Immediate from Theorems 5 and 15 for any small constant p. □

Theorem 17 *Routing packets among all live processors connected by live paths in an $n \times n$ array with k faults, takes $\Theta(n + k^2)$ time.*

Proof. The lower bound arises when the k faults are situated in a continuous path around a square of area $\Omega(k^2)$, with only one exit. We consider any permutation which requires routing all packets in the square of area $\Omega(k^2)$ to processors outside that square. Since all of these packets must go through the single exit, the time it takes is at least $\Omega(k^2)$. As the diameter of the network is n, we obtain the lower bound. The upper bound follows from Theorems 2 and 15. □

Theorem 18 *An r-gridlike $n \times n$ array of processors, each with $O(r)$ storage, can multiply two $n \times n$ matrices in $O(n)$ time.*

Proof sketch. We simulate the standard systolic approach for multiplying matrices in a grid as follows. We simulate the computation in $r \times r$ blocks of the virtual grid, alternating $O(r)$-step computation phases with $O(r)$-step communication phases. The computation phase and the communication phase together simulate r steps of the standard algorithm. In a communication phase, each $r \times r$ subarray forwards $O(r^2)$ data elements to its neighboring $r \times r$ subarrays. Each processor stores every value appearing on its row and column streams, taking an inner product of these streams after the communication phase is complete. The matrix product is complete after $O(n/r)$ computation phases. □

Note that the technique of Theorem 18 will work for any systolic algorithm with unidirectional data dependencies. Also note that all our optimal algorithms depended directly on the r-gridlike property of a faulty array; if we were given only an $O(r)$-dilation embedding, we could not avoid an $O(r)$ slowdown.

3 Upper Bounds on 3-d Arrays

Our upper bounds for three-dimensional arrays are similar in form and spirit to those for two-dimensional arrays. In particular we say that an $n \times n \times n$ array with faults is (α, r)-cubelike, if for every $r \times r \times r$ subcube, at least $(1-\alpha)r$ of the planar sections of the subcube in each dimension satisfy the (α, r)-gridlike properties (*i.e.*, they are (α, r)-gridlike). More simply, we say that a 3-d array is *r-cubelike* if it is (α, r)-cubelike for $\alpha = 1/8$.

In what follows, we show that a 3-d array with k faults is $O(\sqrt{k})$-cubelike, and that a p-faulty 3-d array is $O(\sqrt{\log_{1/p} n})$-cubelike with high probability.

Lemma 19 *An $n \times n \times n$ array with k faults is $(\alpha, \sqrt{k}/\alpha)$-cubelike for any $\alpha \geq k/n$.*

Proof. Fix α and consider any $r \times r \times r$ subcube where $r = \sqrt{k}/\alpha$. Since at most $\sqrt{k}$ planar sections of the r planar sections along each of the dimensions can have more than $\sqrt{k}$ faults each, at least $r - \sqrt{k} = (1-\alpha)r$ planar sections have at most $\sqrt{k}$ faults each. By Lemma 1, each of these $(1-\alpha)r$ planar sections is $(\alpha, \sqrt{k}/\alpha)$-gridlike. □

Lemma 20 *For any constant $\alpha > 0$, there is a constant $p_\alpha > 0$ such that a p-faulty $n \times n \times n$ array is $(\alpha, O(\sqrt{\log_{1/p} n}))$-cubelike with high probability for $p \leq p_\alpha$.*

Proof. Similar to that of Lemma 4 except that we need to show that for each of the 3 dimensions the probability that αr planar sections of an $r \times r \times r$ subcube are not (α, r)-gridlike is very small when $r = \Theta(\sqrt{\log_{1/p} n})$.

By the analysis of Lemma 4, we know that the probability that a specific planar section is not (α, r)-gridlike is

$$q = O\left(r\left[\frac{e^{1-\frac{\alpha}{2}} 5p^{\alpha/2}}{(1-\alpha/2)^{1-\alpha/2}}\right]^r\right).$$

Since faults are independent from each other, the probability that αr or more of the planar sections in a given dimension are faulty is at most

$$\binom{r}{\alpha r} q^{\alpha r} \leq (2q^\alpha)^r = O\left(\left[\frac{2^{1/r} r^{\alpha/r} e^{1-\alpha/2} 5p^{\alpha/2}}{(1-\alpha/2)^{1-\alpha/2}}\right]^{r^2}\right).$$

For any $\alpha > 0$, there is a constant $p_\alpha > 0$ such that for $p \leq p_\alpha$ and $r = O(\sqrt{\log_{1/p} n})$, the above expression is at most $1/3n^4$. Hence the probability that any $r \times r \times r$ subcube fails the cubelike condition is at most $1/n$. □

We next show how to embed an $n \times n \times n$ array into an r-cubelike array with $O(1)$ load, $O(r)$ congestion and $O(r)$ dilation. The embedding is similar to that for 2-d arrays, but the details are more complicated. We begin our 3-d embedding by building highways in all $r \times r \times r$ subcubes. Our highway network is anisotropic, having somewhat better connectivity in two dimensions (dim 1 and 2, below) than in the third. Note: the "dim i faces" of a cube or subcube are those orthogonal to dimension i.

Lemma 21 *A $7r/8 \times 7r/8 \times 7r/8$ virtual grid can be embedded into any $r \times r \times r$ subcube of an r-cubelike array with $O(1)$ load, $O(r)$ dilation, and $O(r)$ congestion.*

Proof. Map virtual grid point (i, j, k) to the (i, j) gridpoint in the kth dim 3 planar section that is r-gridlike. To complete the construction, we need only make connections in dim 3. A constant fraction of the virtual grid points in the kth dim 3 planar section will be used to make the connections between the kth and $k+1$st sections. Since $7r/8$ of the dim 1 planar sections are r-gridlike, the kth and $k+1$st gridlike dim 3 planar sections are connected by $7r/8$ sets of $7r/8$ vertex-disjoint paths, where each path is restricted to the nodes on or between adjacent gridlike planar sections. At least $(2(7/8)^2 - 1)r^2$ of these $(7/8)^2 r^2$ paths have one endpoint which is a virtual grid point on the kth plane. Of those paths at least $(3(7/8)^2 - 2)r^2 > r^2/4$ have as their other endpoint a virtual grid point on the $k+1$st plane. Call the endpoints of these $r^2/4$ paths on the kth plane E_k and

their endpoints on the $k+1$st plane E_{k+1}. The embedding of the edge between virtual grid point (i,j,k) and virtual grid point $(i,j,k+1)$ is constructed from three subpaths: from (i,j,k) to some virtual grid point in E_k using the kth dimension 3 highway system, from that virtual grid point in E_k to the corresponding virtual grid point in E_{k+1} and from the virtual grid point in E_{k+1} to $(i,j,k+1)$ using the $k+1$st dimension 3 highway system. Since there are at least $r^2/4$ gridpoints in E_k (and in E_{k+1}) we can construct the first and third parts of these paths so that only a constant number of paths have any particular gridpoint in E_k and E_{k+1} as their intermediate point and hence the congestion due to the 2nd part of each path is constant. To see that the first and third parts of the paths have only $O(r)$ congestion, we observe that the $O(1)$ to 1 mappings between virtual grid points on the kth plane and gridpoints in E_k can be implemented in a constant number of (partial) permutation steps. Since these permutation routing steps can be executed in $O(r)$ time, at most $O(r)$ packets can use any edge. Hence, the embedded virtual grid has $O(1)$ load, $O(r)$ dilation, and $O(r)$ congestion. □

Surprisingly, it is possible to route on a subcube in $O(r)$ time, despite the congestion and dilation on its embedded subgrid. We use this result to complete the construction of our grid, and to design some of our optimal algorithms.

Lemma 22 *An r-cubelike $r \times r \times r$ subarray of processors, each with $O(1)$ storage, can perform any fixed routing of $O(r^3)$ packets among the processors in the virtual subcube, in $O(r)$ time.*

Proof. By Theorem 14, we can route packets among the virtual grid points in $O(r)$ steps within each dim 3 planar section. The tricky part is routing between dim 3 sections. To route packets between dim 3 sections, we will use the $(7r/8)^2$ paths contained in the $7r/8$ $(7/8,r)$-gridlike dim 2 sections described in the proof of Lemma 21. The problem is to assign each of the $O(r^3)$ packets to one of the $(7r/8)^2$ paths so that no more than $O(1)$ packets enter or exit a path in the same dim 3 section.

By the proof of Lemma 21, we know that for any pair of gridlike dim 3 sections (not necessarily consecutive), at least $r^2/4$ of the $(7r/8)^2$ paths intersect virtual grid points on both sections. Hence each packet has at least $r^2/4$ paths that it might use to go to the correct dim 3 plane. To make an assignment, we construct a coloring problem on a graph with one node for each packet, and an edge between two packets if they begin or end on the same dim 3 section. We have one color for each of our vertex-disjoint paths, so that each node has at least $r^2/4$ legal colorings. The degree of each node is at most $2r^2$, so we can color nodes so that each has $O(1)$ neighbors with the same color.

We have now partitioned the routing problem into 3 phases. In the first phase, we route each packet within its dim 3 section to the virtual grid point that intersects its desired dim 3 path (i.e., the path corresponding to the packet's color). By the coloring argument, at most $O(1)$ packets are routed to any node, and we can use Theorem 15 to accomplish the routing in $O(r)$ steps. In the second phase, we treat the $(7r/8)^2$ paths as linear arrays and route each packet to its correct dim 3 section. Since at most $O(1)$ packets start or end at any point of each path, this can be accomplished in $O(r)$ steps. We complete the routing in the third phase by routing each packet to its correct destination within its dim 3 section, again taking $O(r)$ steps.

□

We now show that a 3-d grid embedded in a p-faulty array has $O(\sqrt{\log_{1/p} n})$ dilation, and that a 3-d grid embedded in an array with k faults has $O(\sqrt{k})$ dilation. In Section 4, we proved that the dilation bound for p-faulty arrays is tight.

Theorem 23 *An $n \times n \times n$ grid graph can be embedded in a r-cubelike $n \times n \times n$ array with $O(r)$ dilation, $O(r)$ congestion, and $O(1)$ load.*

Proof sketch. We will embed a $7n/8 \times 7n/8 \times 7n/8$ grid graph, then adjust the load and congestion by $(8/7)^3$ to obtain an embedding of an $n \times n \times n$ grid.

By Lemma 21, we have defined a $7r/8 \times 7r/8 \times 7r/8$ grid in all our $r \times r \times r$ subcubes that has $O(r)$ dilation, $O(r)$ congestion and load 1. It remains only to connect up the outermost virtual gridpoints in adjacent subcubes. For dimension 3 connections, we can use the same argument as in the proof of Lemma 21 to route the wires using $O(r)$ dilation and $O(r)$ congestion. (This is because the dim 3 sections to be connected can be at most $3r/4$ apart and are thus contained in a single $r \times r \times r$ subcube that overlaps the original subcubes.) For dim 1 and dim 2 connections, we use the fact that adjacent $r \times r \times r$ blocks have at least $(2(7/8)-1)r$ common dim 3 planes that are gridlike in both subcubes. Choose one of the common planes and notice that the planes are connected by at least $(2(7/8)-1)r$ highway segments. By Lemma 22, this means that we can make all of the connections with $O(r)$ congestion and dilation using a single common dim 3 plane. □

By being more careful and using $\Theta(r)$ of the common dim 3 planes in the proof of Theorem 23, we could reduce the congestion of the dim 1 and dim 2 connections to $O(1)$. Unfortunately, we do not know how to reduce the congestion in all dimensions simultaneously. Nevertheless, we can still obtain an optimal simulation of an $m \times m \times m$ array on an $n \times n \times n$ r-cubelike array for $m \geq nr$, as we show in what follows.

Theorem 24 *An r-cubelike $n \times n \times n$ array of processors, each with $O(m^3/n^3)$ storage can simulate a fault-free $m \times m \times m$ array with $O(m^3/n^3 + m^2 r/n^2)$ slowdown.*

Proof sketch. Each virtual node of the $n \times n \times n$ array will simulate an $8m/7n \times 8m/7n \times 8m/7n$ block of the $n \times n \times n$ array. This requires $O(m^3/n^3)$ slowdown. For communication each virtual node needs to send $O(m^2/n^2)$ messages to its neighbors in the virtual grid. By Lemma 22 and the construction in the proof of Theorem 23, one message can be sent from and delivered to each virtual processor in $O(r)$ steps. Hence the communication slowdown is $O(m^2 r/n^2)$. □

Theorem 25 *With probability $1 - 1/n$, a p-faulty $n \times n \times n$ array can simulate a fault-free $n\sqrt{\log_{1/p} n} \times n\sqrt{\log_{1/p} n} \times n\sqrt{\log_{1/p} n}$ array with $O(\log_{1/p}^{3/2} n)$ slowdown.*

Proof. Immediate from Lemma 20 and Theorem 24. □

We can also generalize most of the routing and sorting results from Section 2.3 to hold for r-cubelike arrays. For example, an r-cubelike $n \times n \times n$ array can sort or route n^3 items stored in its virtual gridpoints in $O(n)$ time for $r \leq n$. Although we do not have room to explain the details here, the basic idea is to emulate r steps of columnsort (or linear array routing) in $O(r)$ steps on the r-cubelike array by using Lemma 22 to map the dim i highway segment operations to the $O(1)$ existing congestion dim 1 or 2 highway segments in each $r \times r \times r$ block, and to facilitate the passing of $\Theta(r^3)$ items from one block to any of its neighboring blocks in $O(r)$ steps.

In fact, the only routing or sorting result from Section 2.3 that does not nicely generalize is Theorem 15. The reason is that we can construct examples of $\omega(r^3)$-size connected components in an r-cubelike array that do not intersect any of the highway segments. We can still generalize Theorems 16 and 17, however, since the nasty examples for r-cubelike arrays cannot occur for the special cases when the array is p-faulty or has only k faults. For p-faulty arrays, we can on-line route between all live processors that are connected by live paths in $O(n)$ time with high probability, and for arrays with k worst-case faults, we can perform the routing in $O(n + k^{3/2})$ time. The details will appear in the final version of the paper.

4 Lower Bound on 3-d p-Faulty Arrays

In what follows, we prove that with high probability any embedding of an $\Omega(n) \times \Omega(n) \times \Omega(n)$ array into a p-faulty $n \times n \times n$ array requires dilation of $\Omega(\sqrt{\log_{1/p} n})$.

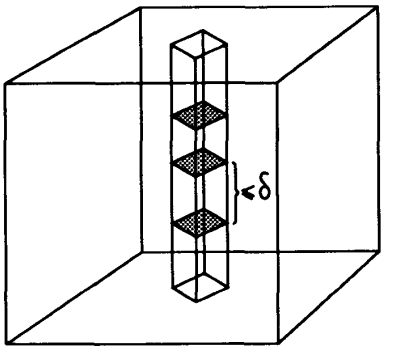

Figure 1: A $W \times W \times W$ subcube, one of the three cylinders of cross section A through its center, and dead cross-sections.

We do this by considering the embedding of the live mesh into the faulty mesh to be a mapping of the live mesh into 3-dimensional space. We then use concepts from knot theory to show that some edge must be mapped to a long path in space.

Our main theorem is stated in terms of the following structural property of faulty arrays. Consider a set of parallel planes at distance W apart, in all dimensions, partitioning the $n \times n \times n$ array into $(n/W)^3$ $W \times W \times W$ subcubes with sides parallel to the coordinate planes. Call this a W-subdivision of the array. For each such subcube consider the three right circular cylinders (in the L_1 "Manhattan" metric) of cross-sectional area A, each of whose central axis passes through the center of the cube and is parallel to a coordinate axis. A planar cross-section of a cylinder is *dead* if all A nodes in that cross-section are faulty. If for all cylinders in every subcube the distance between consecutive dead cross-sections is at most δ, we say that the faulty array is (W, A, δ)-*defective* (see Figure 1).

We now state the main results of the section, deferring the proof of our first theorem.

Theorem 26 *An $an \times an \times an$ grid, G, cannot be embedded in a (W, A, δ)-defective $n \times n \times n$ array, F, with dilation $d \leq \sqrt{A}/8$ and load l or less when*

- $an \geq 542 W^3 A l^{5/3}$, *and*
- $an \geq W/d \geq \max(lA\delta/32, 2)$.

Theorem 27 *For any constant $a > 0$ there exist constants $\epsilon, c > 0$ with the constraint $\epsilon + c < 1/3$ such that, for large enough n, any constant load embedding of an $an \times an \times an$ grid in a p-faulty $n \times n \times n$ array has dilation at least $\frac{1}{8}\sqrt{c \log_{1/p} n}$ with probability greater than $1 - O\left(n^{-n^{\epsilon}+3}\right)$.*

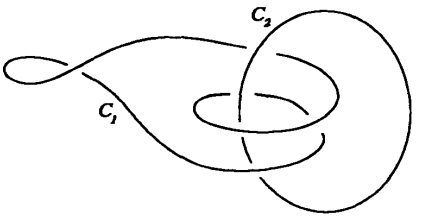

Figure 2: C_1 has a nonzero linking number with C_2.

Proof. Consider a p-faulty $n \times n \times n$ array and its W-subdivision. A cross-section is dead when all its A nodes are faulty and the probability that this happens is p^A. The array is non-defective if one or more of the three cylinders in one or more of the n^3/W^3 subcubes is non-defective, that is, if one of the cylinders has a run of δ non-dead cross-sections. The probability that a cylinder of length W has a run of δ non-dead cross-sections is at most $W(1-p^A)^\delta$. Therefore, the probability that the array is not (W, A, δ)-defective is

$$\leq 3\frac{n^3}{W^2}(1-p^A)^\delta \leq 3\frac{n^3}{W^2}\exp\left(-\delta p^A\right)$$

We choose $A = c\log_{1/p} n$, $\delta = n^{\epsilon+c}\ln n$ and $W = lc^{3/2}n^{\epsilon+c}(\ln n)^{5/2}$. Then the above probability is at most $3n^3W^{-2}n^{-n^\epsilon}$. If $\epsilon + c < 1/3$ and if we choose n large enough, then the constraints of Theorem 26 are satisfied. We have thus proved Theorem 27. □

4.1 Proof of 3d Lower Bound

In this section we present a proof of a central lemma in the proof of Theorem 26. In what follows, we use G to denote the $an \times an \times an$ grid and F the faulty array.

First, we present an elementary concept from knot theory ([14, 15]). Consider two non-intersecting cycles (closed curves) C_1 and C_2 in 3-dimensional space, and suppose each cycle has an orientation (traversal direction). Intuitively, these cycles are linked if they cannot be pulled apart without one of the cycles passing through the other cycle (Figure 2). C_1 is allowed to intersect or pass through itself, and similarly for C_2. A quantity called the *linking number of* C_1 *and* C_2 is nonzero if and only if the cycles are linked in this intuitive sense. We denote this quantity by (C_1, C_2). The absolute value of this quantity is the minimum number of times C_1 has to pass through C_2 in order for the cycles to be unlinked (e.g., it is two in Figure 2). The orientation of the cycles affects only the sign of the linking number. For now, this is all that we need to know about the linking number.

We proceed by considering a (W, A, δ)-defective cube and its subdivision into subcubes. Two subcubes are said to be *adjacent* if and only if they share a face. A simple cycle of subcubes can be defined relative to this adjacency relation. Notice that given a cycle of subcubes $\widetilde{C}$, a cycle consisting of paths between centers of adjacent subcubes through their cores can be traced. We call this cycle, the *core cycle* of the cycle of subcubes. For the sake of convenience assume that W is even, hence the center of a cube does not lie on a mesh vertex and the core cycle does not intersect any mesh vertices or edges. We use $\widetilde{C}$ to denote both the cycle of subcubes and its core cycle as it will be clear from the context which of the two is meant.

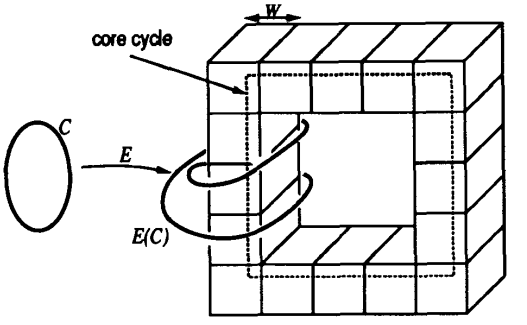

Figure 3: $\mathcal{E}(C)$ links $\widetilde{C}$.

Now consider a simple cycle in the graph, G. We call a cycle, C, a *planar simple cycle* if and only if all its nodes lie in a single plane in the cube.

Consider a cycle C in G, and a simple cycle of subcubes $\widetilde{C}$ in F. Consider an embedding, $\mathcal{E}$, of the guest cube G in the faulty cube, F. In what follows, we use $\mathcal{E}(\cdot)$ to refer to the image of a node, an edge, a set of nodes, or a set of edges.

Definition 1 *We say that* C links $\widetilde{C}$ *under* $\mathcal{E}$ *if no node in* $\mathcal{E}(C)$ *is embedded in any of the subcubes in* $\widetilde{C}$ *and the linking number of* $\mathcal{E}(C)$ *and the core cycle of* $\widetilde{C}$ *is nonzero (Figure 3).*

With these definitions we can state a central lemma:

Lemma 28 *For any embedding,* $\mathcal{E}$, *of an* $an \times an \times an$ *mesh,* G, *in a* (W, A, δ)*-faulty mesh,* F, *with maximum load* l, *such that* $an \geq W/d \geq \max(lA\delta/32, 2)$,

either: there is an edge whose embedding has length greater than $d = A^{1/2}/8$,

or: no planar simple cycle of G *links any cycle of subcubes* $\widetilde{C}$ *under* $\mathcal{E}$.

Proof. The proof is by contradiction. Thus, we assume that all the embedded edges have length at most $d =$

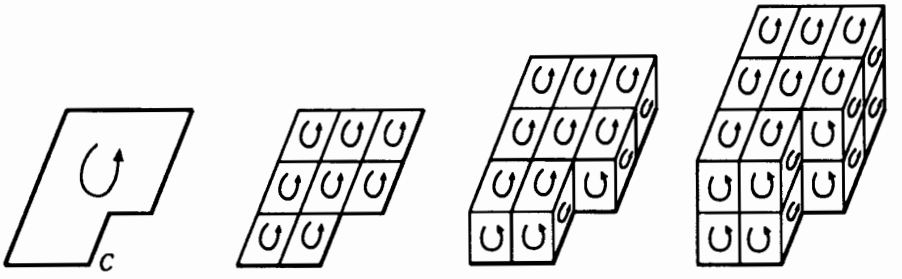

Figure 4: Three decompositions of a planar cycle C into 4-cycles.

$A^{1/2}/8$ and we assume that some cycle in C in some plane Π links some cycle of subcubes $\widetilde{C}$. (Without loss of generality we can assume that C lies in the plane Π consisting of the set of nodes whose third coordinate is fixed.)

Here is an intuitive outline of the proof: we consider many 4-cycle decompositions of the cycle C (i.e., decompositions into cycles of length four). Since $\mathcal{E}(C)$ links the core cycle of $\widetilde{C}$ then the embedding of some 4-cycle in each decomposition must link the core cycle. In fact, we can show that if these decompositions are carefully chosen then each will contain a distinct 4-cycle which links the core-cycle. Unlike $\mathcal{E}(C)$, the embedding of an individual 4-cycle is very small because we have assumed a small dilation, and therefore the embedding of the 4-cycle will closely encircle the core cycle. Then we will show that the embeddings of all the distinct 4-cycles lie between the same two consecutive dead cross-sections. Finally, we argue that there is not enough space between successive dead planes of a core for all the 4-cycles to fit. This is the desired contradiction.

To proceed, we formally define the above concepts. First, recall that the linking number of two cycles is computed in terms of oriented cycles, so we consider oriented traversals of cycles and their decompositions. We use standard notions from graph theory (see for example [3] chapter 2, or [4] chapter 12). Let $G = (V, E)$ be a graph and $E = \{e_1, e_2, \ldots, e_m\}$ its set of edges. Assign an arbitrary direction to the edges. Consider a cycle in (the undirected) G together with a traversal orientation. Then this cycle can be written as a vector of length m with entries $\{-1, 0, 1\}$ where the ith entry is 0 if the edge is not contained in the cycle and $+1$ or -1 depending on whether the cycle traverses the edge along its direction or its opposite. These vectors span a subspace of $\mathbf{R}^m$, called the *cycle space,* which for connected graphs has dimension $|E| - |V| + 1$. [N.B. Here we are interested only in scalars which are integers. For a given set of cycles each entry in the corresponding vector denotes the number of times each edge is traversed and in what direction.] In the case of the 3-dimensional mesh consider the set of its oriented 4-cycles (cycles of length four). We can easily see that these form a spanning set of the cycle space (though not a base). Hence every oriented cycle in the mesh can be *decomposed* into 4-cycles.

Our planar cycle C has many decompositions into 4-cycles. (We are in fact interested in the images of these decompositions under $\mathcal{E}$.) We define a set of $W/2d$ such decompositions as follows (see Figure 4). Recall that C is a simple planar cycle. Assume an arbitrary orientation of it. Consider the right cylinders defined by C and having a base in the plane of C and a base in a plane parallel to it at distance h ($1 \leq h < W/2d$). (This can be done since $an > W/d$.) For each such cylinder, the 4-cycles in its sides and in its top base, when appropriately oriented, constitute a decomposition of C into 4-cycles. To complete our discussion of the decompositions of C, we describe a way to transform one decomposition into another. Define an *incremental change* to a decomposition to be one obtained by the addition or deletion of a set of 4-cycles belonging to a single unit ($1 \times 1 \times 1$) cube, each appropriately oriented. The following fact holds:

Fact 29 *Any of our decompositions of C can be obtained from the planar decomposition of C by a sequence of incremental changes.*

Next we discuss a decomposition of the core cycle of $\widetilde{C}$. Consider the points on $\widetilde{C}$ where the core cycle intersects the dead cross-sections. Join these points to a point at infinity via lines perpendicular to the core cycle through the dead cross-sections. We thus decompose the core cycle into *section cycles* by expressing it as the sum of the cycles formed by the section of the core cycle between two dead planes and the paths to and from its endpoints and the point at infinity (Figure 5).

Before we proceed, we mention the following fact about cycle decompositions and linking numbers. It is a direct consequence of the definition (in fact, of

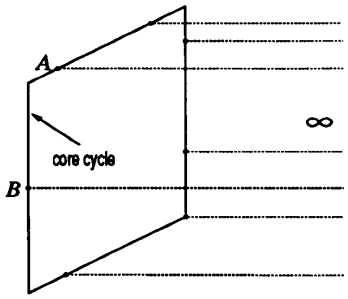

Figure 5: Decomposition of $\widetilde{C}$ into section cycles (e.g. $\infty AB\infty$). Points on the core cycle are intersection points with dead cross-sections.

the many equivalent definitions [15]) of the linking number.

Fact 30 *Consider two cycles C^1 and C^2 and two decompositions $C^1 = \sum_i C_i^1$ and $C^2 = \sum_j C_j^2$. Then*

$$(C^1, C^2) = \sum_i (C_i^1, C^2) = \sum_j (C^1, C_j^2) = \sum_{i,j} (C_i^1, C_j^2).$$

Recall we assumed that $(\mathcal{E}(C), \widetilde{C}) \neq 0$. Having described the decomposition of C into 4-cycles and of $\widetilde{C}$ into section cycles we now prove the following claim:

Claim 31 *There is a section cycle such that for each one of the $W/2d$ decompositions of C there exists a 4-cycle whose image has non-zero linking number with both that section cycle and the core cycle. Further, these 4-cycles are distinct for each decomposition.*

Proof of Claim: First observe that every 4-cycle that links the core cycle links exactly one section cycle. This follows from the fact that every node in the image of such a 4-cycle has to be within distance $2d$ of the core cycle. Hence the j-cycle is confined to a cylinder between two consecutive dead cross-sections (Figure 6). The linking number of such a 4-cycle with the core cycle is equal to its linking number with the unique section cycle that it links.

Let $\mathcal{S}$ be the set of section cycles in the section-cycle decomposition of the core cycle $\widetilde{C}$. Let $\mathcal{A}$ be the images of the 4-cycles in some 4-cycle decomposition of C. Under this notation, $\mathcal{E}(C) = \sum_{\alpha \in \mathcal{A}} \alpha$ and $\widetilde{C} = \sum_{\sigma \in \mathcal{S}} \sigma$. From Fact 30 we have

$$(\mathcal{E}(C), \widetilde{C}) = \sum_{\alpha \in \mathcal{A}} (\alpha, \widetilde{C}) = \sum_{\substack{\alpha \in \mathcal{A} \\ (\alpha, \widetilde{C}) \neq 0}} (\alpha, \widetilde{C})$$

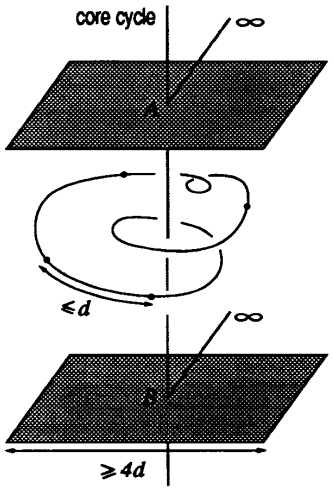

Figure 6: The embedding of a 4-cycle that links the core cycle as well as the section cycle $\infty AB\infty$ is confined to a cylinder between two consecutive dead cross-sections.

$$= \sum_{\substack{\alpha \in \mathcal{A} \\ (\alpha, \widetilde{C}) \neq 0}} \sum_{\sigma \in \mathcal{S}} (\alpha, \sigma) = \sum_{\sigma \in \mathcal{S}} \sum_{\substack{\alpha \in \mathcal{A} \\ (\alpha, \widetilde{C}) \neq 0}} (\alpha, \sigma).$$

Therefore,

$$(\mathcal{E}(C), \widetilde{C}) = \sum_{\sigma \in \mathcal{S}} \sum_{\substack{\alpha \in \mathcal{A} \\ (\alpha, \widetilde{C}) \neq 0 \\ (\alpha, \sigma) \neq 0}} (\alpha, \sigma).$$

The constraints, $(\alpha, \widetilde{C}) \neq 0$ and $(\alpha, \sigma) \neq 0$, could be added since, clearly, the α's that do not satisfy them do not contribute in the above sum.

We know that $(\mathcal{E}(C), \widetilde{C}) \neq 0$, and thus

$$\exists \sigma_0 \in \mathcal{S} \text{ such that } \sum_{\substack{\alpha \in \mathcal{A} \\ (\alpha, \widetilde{C}) \neq 0 \\ (\alpha, \sigma_0) \neq 0}} (\alpha, \sigma_0) \neq 0. \qquad (*)$$

Since this sum is non-zero, there must be some 4-cycle in the decomposition $\mathcal{A}$ of C whose image has non-zero linking number with both the section cycle σ_0 and the core cycle.

The sum in $(*)$ has the same value for each of our $W/2d$ decompositions. We show this by using the fact that we can transform any of these decompositions to any other decomposition by a sequence of incremental changes. Recall that an incremental change is the addition of the set of cycles, $\mathcal{G}$, of a single unit cube

(appropriately oriented). An incremental change adds

$$\sum_{\substack{\gamma\in\mathcal{E}(\mathcal{G})\\(\gamma,\sigma_0)\neq 0\\(\gamma,\widetilde{C})\neq 0}} (\gamma,\sigma_0), \qquad (**)$$

to the value of $(*)$ for the new decomposition. Thus, we must simply show that the sum $(**)$ is zero.

The sum $(**)$ is nonzero only if there exists $\gamma_0 \in \mathcal{E}(\mathcal{G})$ satisfying $(\gamma_0, \widetilde{C}) \neq 0$ and $(\gamma_0, \sigma_0) \neq 0$, since these are constraints on the sum. When this happens, we argue that for all $\gamma \in \mathcal{E}(\mathcal{G})$, $(\gamma, \widetilde{C}) = (\gamma, \sigma_0)$. The argument is essentially the same as the one given in the beginning of this proof: if γ_0 encircles a section of $\widetilde{C}$ between two consecutive dead cross-sections, then the embedding of the entire cube is confined to the cylinder between these two cross-sections, since an embedded edge of G cannot be stretched more than d. Finally, the sum of the cycles in $\mathcal{G}$ is zero, thus $\sum_{\gamma\in\mathcal{E}(\mathcal{G})}(\gamma,\sigma_0) = 0$. So

$$0 = \sum_{\gamma\in\mathcal{E}(\mathcal{G})} (\gamma,\sigma_0) = \sum_{\substack{\gamma\in\mathcal{E}(\mathcal{G})\\(\gamma,\sigma_0)\neq 0}} (\gamma,\sigma_0) = \sum_{\substack{\gamma\in\mathcal{E}(\mathcal{G})\\(\gamma,\sigma_0)\neq 0\\(\gamma,\widetilde{C})\neq 0}} (\gamma,\sigma_0).$$

(We safely added the second constraint, $(\gamma, \widetilde{C}) \neq 0$, since $(\gamma, \widetilde{C}) = (\gamma, \sigma_0)$.) Hence an incremental change does not change the sum $(*)$, and $(*)$ is the same for all our decompositions.

We therefore conclude that each decomposition contains a 4-cycle whose image links the core cycle at a particular section. These 4-cycles are distinct since there are cycles on the top bases of the cylinders on which the decompositions were defined. The common cycles of the decompositions are cycles on the sides of the cylinders. These are at distance at most $W/2d$ from C, therefore their embeddings are at lest $W/2$ away from the core cycle and therefore cannot link it. This concludes the proof of Claim 31. □

Finally we finish the proof of the lemma as follows: There must be at least $W/2d$ times 4 points with distance $2d$ of a δ length line. But the volume is at most $4d^2\delta$. So only $4ld^2\delta$ points can fit in this volume. So when $W/d > l(A/32)\delta$ all the points cannot fit. Thus we have our contradiction, and Lemma 28 has been proved. □

We now sketch the final stage of the proof. We want to prove the theorem by contradiction. Assume that there exists a small dilation embedding of G in F. We are going to show that there exist two points on the same plane in G whose embeddings are far apart. Furthermore, we show the existence of a path between these two points whose embedding passes through a small diameter doughnut (consisting of $W \times W \times W$ subcubes). In addition, we show that the two points are connected by many paths. By Lemma 28 all these paths must pass through the doughnut. But there is not enough room, which leads to a contradiction. This concludes the proof sketch of Theorem 26. □

Acknowledgments

We are extremely grateful to Greg Nelson for numerous helpful discussions. We would also like to thank Nati Linial, Yanzhang Lu and Shiaoling Peng for useful discussions.

References

[1] F. Annexstein and M. Baumslag. A unified approach to off–line permutation routing on parallel networks. In *Second ACM Annual Symposium on Parallel Algorithms and Architectures*, pages 398–406, 1990.

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

[3] C. Berge. *Graphs*. North-Holland, 1985.

[4] J. A. Bondy and U.S.R. Murty. *Graph Theory With Applications*. American Elsevier, 1977.

[5] J.W. Greene and A. El Gamal. A framework for solving VLSI graph layout problems. *Journal of the ACM*, 31:694 – 717, 1984.

[6] J. Hastad, F.T. Leighton, and M. Newman. Reconfiguring a hypercube in the presence of faults. In *19th Annual Symposium on Theory of Computing*, pages 274–284, 1987.

[7] F.T. Leighton. Tight bounds on the complexity of parallel sorting. *IEEE Transactions on Computers*, C–34(4):344–354, April 1985.

[8] F.T. Leighton and C.E. Leiserson. *A Survey for Algorithms for Integrating Wafer-Scale Systolic Arrays*, In *Wafer Scale Integration*, (G. Saucier, J. Trilhe eds.), pp. 177–195, North-Holland, 1986.

[9] F.T. Leighton and C.E. Leiserson. Wafer-scale integration of systolic arrays. *IEEE Transactions on Computing*, C-34, 1985.

[10] F.T. Leighton, C.E. Leiserson, and Dina Kravets. Advanced parallel and VLSI computation: Lecture notes for 18.435/6.848, May 1990. MIT/LCS/RSS 8.

[11] F.T. Leighton and B. Maggs. Expanders might be practical: Fast algorithms for routing around faults in multibutterflies. In *30th Annual Symposium on Foundations of Computer Science*, pages 384–389, 1989.

[12] Nathan Linial and Mike Saks. Low diameter graph decomposition. *Manuscript*, 1990.

[13] P. Raghavan. Robust algorithms for packet routing in a mesh. In *First ACM Annual Symposium on Parallel Algorithms and Architectures*, pages 344–350, 1989.

[14] K. Reidemeister. *Egebnisse der Mehtematik and ihre Grenzgebiete(Alte Folge)*. Band 1, Heft 1, Springer-Verlag, 1932. Reprint, Chelsea 1948; English Translation, *Knot Theory*, BCS Associates, Moscow, Idaho, 1983.

[15] D. Rolfsen. *Knots and Links, Mathematics Lecture Series 7*. Publish or Perish, 1976.