Fault Diagnosis in a Small Constant Number of Parallel Testing Rounds

Richard Beigel*
Yale University

Grigorii Margulis†
Yale University and
Russian Academy of Sciences

Daniel A. Spielman‡
MIT

Abstract

Consider a set of processors, V, that can communicate with each other. Assume that each processor can be either "good" or "faulty". Also assume that the processors can be used to test each other. We provide a parallel algorithm that determines which processors are good and which are faulty in 32 rounds of testing, provided that a strict majority of the processors are good.

1. Introduction

Consider a set of processors, V, that can communicate with each other. Assume that each processor can be either "good" or "faulty". Also assume that the processors can be used to test each other. If a good processor tests a good processor, it outputs "good"; if a good processor tests a faulty processor, it outputs "faulty"; we make no assumptions about the output of faulty processors. The fault *diagnosis* problem is to determine which processors are good, and which are faulty. This model was introduced by Preparata, Metze, and Chien [12]. They observed that fault diagnosis is possible in general only if a majority of the processors are good. We will write n to denote the number of processors and t to denote an upper bound on the number of faulty processors. We assume throughout this paper that $t < n/2$.

In the *nonadaptive* version of the Preparata-Metze-Chien model, the choice of which processors to test must be made before any tests are performed; in the *adaptive* version, the choice of which processors to test is allowed to depend on the results of prior tests. The adaptive model was studied further in [3, 6, 7, 13, 2], and the nonadaptive model was studied further in [9, 5, 15, 2].

Hakimi and Schmeichel [7] asked how long the testing would take if the processors could test each other in parallel. A testing round can be viewed as a directed matching where the processors are the vertices and an edge is drawn from each testing processor to the processor it tests. Hakimi and Schmeichel [7] showed that $O(t+\log n)$ rounds suffice. Schmeichel, Hakimi, Otsuka, and Sullivan [13] improved this to $O(\log_{\lfloor n/t \rfloor} t)$ rounds.

Beigel, Kosaraju, and Sullivan [2] introduced a variation on the fault diagnosis model, allowing two processors to test each other simultaneously. We call such tests *undirected* tests; one processor testing another is called a *directed* test. It is clear that undirected tests can be simulated via two rounds of directed tests. The undirected model simplifies certain analysis. They [2] designed an algorithm that has two processors test each other if they are adjacent in certain bounded-degree expander graphs. Recall Vizing's Theorem [4], which says that a graph of degree d is a subset of $d+1$ matchings. This implies that $d+1$ rounds of undirected tests are sufficient to test each pair of vertices joined by an edge in a graph of degree d. They [2] concluded therefore that a constant number of testing rounds suffice. However, they did not present the constant explicitly (it's about 5000000), and they did not make any attempt to optimize it.

Warren Smith [14] conjectured that it should be possible to obtain a small constant by using random graphs. We validate his conjecture, by designing a deterministic algorithm based on random graphs that performs fault

*Dept. of Computer Science, P.O. Box 2158, Yale Station, New Haven, CT 06520-2158. rjb@cs.yale.edu.

Supported in part by NSF grant CCR-8958528.

†Dept. of Mathematics, P.O. Box 2155, Yale Station, New Haven, CT 06520-2155. margulis@math.yale.edu. On leave from the Institute for Problems in Information Transmission, Russian Academy of Sciences.

Supported in part by NSF grant DMS-9204270.

‡Dept. of Applied Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139. spielman@math.mit.edu.

Partially supported by the Fannie and John Hertz Foundation. Supported in part by NSF grant CCR-8958528 under an REU supplement while at Yale College.

diagnosis in 32 rounds of directed tests. Surprisingly, our analysis is easier in the stronger directed model.

Overview of our algorithm: We produce a certain kind of random directed graph of degree 28 on n vertices that will, with very high probability, for any set containing more than $\frac{1}{2}n$ good processors, induce a strongly connected component containing more than $\frac{1}{6}n$ good processors. The first 28 rounds of tests are determined nonadaptively according to any such graph. Fault diagnosis is completed adaptively in 4 additional rounds. Since at least one such graph exists for each n, we have a deterministic algorithm that uses 32 rounds of testing; however, it could take exponential preprocessing time to verify that the randomly generated graph has the desired property. In this sense, the algorithm is nonconstructive. (If we merely desire a 0-sided error randomized algorithm that works with high probability for each possible individual fault set, then even fewer rounds suffice. This is a topic for the full paper.)

In the last section, we use expander graphs to produce a constructive deterministic algorithm that determines which processors are good and which are faulty in 84 rounds of testing.

Both algorithms are much simpler and much more efficient than what was previously known. Graphs that induce a large strong component on any half of the vertices account for much of the improvement. This property is similar to expansion, but not the same, even in undirected graphs, although it follows from properties of eigenvalues in the undirected case. The use of directed graphs seems empirically to be worth a factor of two as well, at least in the nonconstructive algorithm.

2. Random Directed Hamiltonian Paths

Consider the following algorithm for choosing a directed Hamiltonian path uniformly at random.

> $E \leftarrow \phi$;
> choose any order $v_1, v_2, \ldots, v_n$ of the vertices in V;
> for $1 \le i \le n$ do
>   insert v_i into the path E at a location chosen uniformly at random.

Lemma 1. *Let V be a set of n vertices. Let A and B be disjoint subsets of V of sizes αn and βn respectively. Let E be a random directed Hamiltonian path on V. Then*

$$\operatorname*{Prob}_E[\forall v \in A, \forall w \in B : (v,w) \notin E] = \frac{(n-\alpha n)!(n-\beta n)!}{n!(n-\alpha n-\beta n)!}.$$

Proof: We can assume without loss of generality that the algorithm for choosing the random Hamiltonian path first creates a random Hamiltonian path on all the vertices in $V - A$, and then adds the vertices in A one by one. If one of the vertices in A is added so that it points to a vertex in B, then no matter how the other vertices in A are added, one of them will point to a vertex in B. Thus, the probability that no vertex in A points to a vertex in B is the product of the probabilities that the ith vertex in A does not point to a vertex in B, for $1 \le i \le \alpha n$. The probability that the ith vertex of A added to the path does not point to a vertex in B is $\frac{n-\alpha n-\beta n+i}{n-\alpha n+i}$. Thus, the probability that no vertex in A points to a vertex in B is

$$\prod_{i=1}^{\alpha n} \frac{n-\alpha n-\beta n+i}{n-\alpha n+i} = \frac{(n-\alpha n)!(n-\beta n)!}{(n)!(n-\alpha n-\beta n)!}. \quad \blacksquare$$

3. Fourteen directed Hamiltonian paths suffice

Let H_d be the distribution of graphs on V obtained by taking the union of d directed Hamiltonian paths chosen independently at random and then removing all self-loops and duplicate edges. We will show that if E is chosen according to H_{14}, then it is very unlikely that there is any subset V' of V of size greater than $n/2$ on which E does not induce a strongly connected component of size greater than $n/6$.

To find a large strongly connected component, we use an extension of a lemma by Erdos and Renyi (see, for example, [11, p. 45]).

Lemma 2. *Let V be a set of n vertices and let E be a set of edges on V such that for all subsets V' of V of size $n/2$ and for all partitions (A,B) of V' where $n/6 \le |A| \le n/4$, E contains edges that cross both from A to B and from B to A. Then, for every subset W of V of size greater than $n/2$, E induces exactly one strongly connected component of size greater than $n/6$ on W.*

Proof: Let W be a subset of V of size greater than $n/2$. Let $C_1, \ldots, C_m$ be the strongly connected components of W under E. These components form a directed acyclic graph, so we can assume without loss of generality that $C_1, \ldots, C_m$ are chosen so that $i < j$ implies that there is no edge in E from C_j to C_i.

To prove the existence of a large strongly connected component, it suffices to assume that $|W| = n/2$. Assume by way of contradiction that E does not induce a strongly connected component of size greater than $n/6$ on W. Let $a_i = |C_i|/n$, for $1 \le i \le m$. By assumption, $a_i \le 1/6$, for all i. Since $\sum_{i=1}^m a_i = 1/2$, there exists a j, $1 < j < m$, such that $1/6 \le \sum_{i=1}^j a_i \le 1/3$. We let $A = \cup_{i=1}^j C_i$ and $B = \cup_{i=j+1}^m C_i$. By the order in which we chose the C_i's, there is no edge from B to A. By changing A and B if necessary, we can assume without loss of generality that $|A| \le n/4$.

Now, assume that there exist j and k such that $j < k$, $|C_j| > n/6$ and $|C_k| > n/6$. We let $A = \cup_{i=1}^{j} C_i$ and $B = \cup_{i=j+1}^{m} C_i$. By the order in which we chose the C_i's, there is no edge from B to A. $|A| > n/6$ and $|B| > n/6$, but W may be of size greater than $n/2$. To obtain the contradiction, discard vertices from A and B until $|A| + |B| = n/2$, being careful to preserve the property that $|A|$ and $|B|$ have size greater than $n/6$. ∎

Now, we can prove the following:

Theorem 3. *Let V be a set of n vertices. Choose E according to H_d. Then, with probability at least*

$$1 - e^{n\left(\frac{3}{2}\ln 2 + d\left(\frac{2}{3}\ln\frac{2}{3} + \frac{5}{6}\ln\frac{5}{6} + \frac{1}{2}\ln 2\right)\right) + O(1)},$$

for all subsets V' of V of size at least $n/2+1$, E induces exactly one strongly connected component on V' of size greater than $n/6$.

Proof: By Lemma 2, it suffices to prove that there is an exponentially small probability that there is a subset V' of V of size $n/2$ which has a partition (A, B) such that $1/6 \leq |A| \leq 1/4$ and either no edge of E goes from A to B or no edge of E goes from B to A. By Lemma 1, the probability that one random directed Hamiltonian path does not have an edge from A to B is

$$\frac{(n-\alpha n)!(n-\beta n)!}{n!(n-\alpha n-\beta n)!},$$

where $|A| = \alpha n$ and $|B| = \beta n$. Thus, the probability that E does not have an edge from A to B or E does not have and edge from B to A is at most

$$\left(2\frac{(n-\alpha n)!(n-\beta n)!}{n!(n-\alpha n-\beta n)!}\right)^d.$$

There are at most 2^n choices for V', and for each of these there are at most $2^{n/2}$ choices for the sets A and B, so the probability that there exists a subset V' of size $\frac{1}{2}|V|$ with a partition (A, B) such that E does not have an edge from A to B or E does not have an edge from B to A and $1/6 \leq |A| \leq 1/4$ is at most

$$2^{3n/2}\left(\frac{(n-\alpha n)!(n-\beta n)!}{n!(n-\alpha n-\beta n)!}\right)^d.$$

This is maximized when $\alpha = 1/6$. Applying Stirling's formula, we find that

$$2^{3n/2}\left(\frac{(n-\alpha n)!(n-\beta n)!}{n!(n-\alpha n-\beta n)!}\right)^d \leq e^{n\left(\frac{3}{2}\ln 2 + d\left(\frac{2}{3}\ln\frac{2}{3} + \frac{5}{6}\ln\frac{5}{6} + \frac{1}{2}\ln 2\right)\right) + O(1)},$$

assuming that $1/6 \leq \alpha \leq 1/4$. ∎

Corollary 4. *Let V be a set of n vertices, where n is divisible by 6. Choose E according to H_{14}. Then, with probability at least*

$$1 - e^{-0.019n + O(1)},$$

for all subsets V' of V of size at least $n/2+1$, E induces a strongly connected component on V' of size greater than $n/6$.

4. Fault Diagnosis

We have shown that the graph H_{14}, with high probability, induces exactly one strong component consisting of more than $\frac{1}{6}n$ good processors. If there is only one big strong component, then obviously it is the good one and it can test the remaining $\frac{5}{6}n$ processors in 5 rounds. However, it may also induce one or two strong components consisting of more than $\frac{1}{6}n$ faulty processors. Now we present a simple algorithm that completes the diagnosis in 5 rounds even in these less easy cases. In fact, we present a 4 round algorithm in the Appendix; note that this beats the naive algorithm even in the easy case.

Lemma 5. *Let V be a set of n processors and let V' be the subset of good processors. Assume that $n \geq 2(k^2-k)$ and $k \geq 4$.*[1] *Let E be a graph on V that induces at least one strongly connected component of size greater than $\frac{1}{k}n$ on V'. Then we can determine which processors are good and which processors are faulty by performing the tests indicated by E and then performing $k-1$ more rounds of adaptive tests.*

Proof: Throughout the proof we will write *component* to mean "maximal strongly connected component of V induced by E," and *big component* to mean "component of size greater than $\frac{1}{k}n$." Observe that every component contains only good processors or only faulty processors. We call a set of processors *good* if every processor in it is good; we call it *faulty* if every processor in it is faulty.

Let m be the number of big components induced by E. We now have three cases to deal with:

Case 1: $m \leq \frac{k}{2}$**.** In $k - m$ rounds, we have some of the processors in each big component test all the processors that are not in big components. We consider two exhaustive cases.

Case 1a: the sets of vertices that each big component says are good are pairwise disjoint. Then exactly one of the big components must be good because good components must give identical diagnoses. Thus we can assume that processors from distinct big components dislike each other. The good component

[1] We remark that if k is even it suffices to assume that $n \geq k \geq 4$.

will be the unique component that likes a majority of all the processors.

Case 1b: there is a processor that at least two big components say is good. Call this processor the referee, and call these two big strong components Conan and Samson. Simultaneously, have the referee test Conan and have Conan test Samson (this is possible because by our assumptions each big component contains at least two nodes).

If the referee dislikes Conan, then Conan is faulty. If the referee likes Conan and Conan dislikes Samson, then Samson is faulty. If the referee likes Conan and Conan likes Samson, then we merge Conan, Samson, and the referee into a single big component. In all three of these subcases we eliminate at least one big component per round.

We proceed in this fashion until only one big component remains or until there is no processor that two big components say are good. We will have used at most $m-1$ additional rounds.

Case 2: k is odd and $m = \lceil \frac{k}{2} \rceil$. In this case k is odd so we can write $k = 2j+1$ and $m = j+1$. We now use $j+1$ rounds to have the processors in each big component test all the processors that are not in big components. Since we required that $n \geq k^2 - 1$, we can allow two processors in each big component to remain idle and still have enough processors left to perform all these tests. As opposed to letting these processors remain idle, we will use them to perform tests from every big component to every other big component. Since the complete graph on $j+1$ vertices is contained in the union of $j+1$ matchings, we can perform these tests in the same $j+1$ rounds. If there are two big components that say each other are good, we can merge these two big components together. We will then be left with j big components, and can finish in an additional $j-1$ rounds by following the strategy described in Case 1.

If no two big components say each other are good, then we know that j of these big components are faulty and only one is good. Since the majority of the processors are good, this good component must like at least $\frac{n}{2} - \frac{n}{k}$ of the processors not in the big components. Accordingly, we can declare as faulty any big component that likes fewer than this many processors. Since there are at most $\frac{n}{2} - \frac{n}{2k}$ processors not in big components, and since $n \geq 4k$, there are at least two processors that are liked by at least three big components. Thus, in one round, we can use these processors as referees to eliminate at least two of the big components. We can then finish the testing as we did in Case 1.

Case 3: $m > \lceil \frac{k}{2} \rceil$. We reduce Case 3 to Case 1 or Case 2 by pretending that there are only $\lceil \frac{k}{2} \rceil$ big components. That is, we treat the other big components as if they were not big components. Since most of the processors are good, we know that at least one of the big components that we keep is good. ∎

What we have proved so far suffices to produce a 33-round testing algorithm. The Appendix's improved version of Lemma 5 for the case $k = 6$ gets this bound down to 32.

Theorem 6. *For a set of processors V, there exists an algorithm that will determine which processors are good and which processors are faulty in 32 testing rounds, assuming that a majority of the processors are good.*

Proof: By Corollary 4, there exists a directed graph E of degree 28 such that for every subset of more than $\frac{1}{2}|V|$ processors, E induces exactly one connected component of size $\frac{1}{6}|V|$ in that subset. We can split each Hamiltonian path in E into two matchings, and perform all the tests indicated by E in 28 rounds. By Lemma 11, we can then determine which processors are good and which are faulty in an additional 4 rounds of testing. ∎

We remark that a graph chosen at random from H_{14} has an exponentially high probability of being sufficient for this theorem. We could also produce a randomized algorithm that chooses its Hamiltonian paths at random each time it is given a problem instance. This algorithm uses fewer Hamiltonian paths and hence fewer testing rounds; it has zero-sided error since it will be able to tell if it has not found a strongly connected component of more than $n/6$ processors. We will discuss this further in the full paper.

5. Nonadaptive Algorithms

Preparata, Metze, and Chien [12] proved that any nonadaptive fault-diagnosis algorithm must use a testing graph of degree at least t, where t is the number of faults to be tolerated. Thus, it is difficult to perform complete fault-diagnosis nonadaptively. A problem that nonadaptive algorithms can handle is the "diagnosis-with-repair" problem [9, 12], in which the diagnoser must identify at least one faulty processor or certify that there are none. Beigel, Kosaraju, and Sullivan [2] show that there exist a constant γ and a 3-regular undirected test graph that will identify a good processor and a faulty processor (if there is one), assuming that the number of faulty processors is less than γn. Since a 3-regular undirected graph is contained in the union of 4 undirected matchings, their test sequence may use as many as 8 rounds of tests in the directed model.

Beigel, Kosaraju, and Sullivan [2] also showed that a test graph consisting of a single undirected Hamiltonian cycle can solve the diagnosis-with-repair problem only when $t = \theta(\sqrt{n})$. In contrast, we show that a test graph consisting of the union of 2 directed Hamiltonian paths can solve the diagnosis-with-repair problem if as many as $\frac{1}{13}n$ faults are to be tolerated. This uses 4 rounds of testing.

Lemma 7. *Let V be a set of n processors. Choose E according to H_d. Then, the probability that there is a choice of at most $\frac{1}{s}|V|$ faulty processors such that E fails to induce a strongly connected component of size greater than $\frac{1}{s}|V|$ in the good processors is at most*

$$\binom{n}{\frac{1}{s}n} 2^{\frac{s-1}{s}n} \left(\frac{(n - \frac{s-2}{2s}n)!(n - \frac{1}{2}n)!}{(n)!(n - \frac{s-2}{2s} - \frac{1}{2}n)!} \right)^d .$$

Proof: Given a choice for the faulty processors, pick any $\frac{s-1}{s}n$ of the good processors. If there is no strongly connected component of size greater than $\frac{1}{s}n$ in these good processors, then by Lemma 2, there is a partition (A, B) of these good processors such that no edge points from A to B or no edge points from B to A, and $\frac{s-2}{2s}n \leq |A| \leq \frac{1}{2}n$. There are at most $\binom{n}{\frac{1}{s}n}$ possible choices for the faulty processors. For each of these, there are at most $2^{\frac{s-1}{s}n}$ possible partitions of the $\frac{s-1}{s}n$ good processors. Thus, by Lemma 1, the probability that there exists a choice of the faulty processors such that there is a partition (A, B) of the good processors such that no edge points from A to B or no edge points from B to A, and $\frac{s-2}{2s}n \leq |A| \leq \frac{1}{2}n$ is at most

$$\binom{n}{\frac{1}{s}n} 2^{\frac{s-1}{s}n} \left(\frac{(n - \alpha n)!(n - \beta n)!}{(n)!(n - \alpha n - \beta n)!} \right)^d ,$$

where $\alpha + \beta = \frac{s-1}{s}$ and $\frac{s-2}{2s} \leq \alpha \leq 1/2$. This is maximized when $\alpha = \frac{s-2}{2s}$. ∎

Theorem 8. *Let V be a set of n processors. Then there exists an oblivious algorithm which will solve the diagnosis-and-repair problem in 4 rounds of testing, assuming that at most $\frac{1}{13}|V|$ of the processors are faulty.*

Proof: Choose E according to the subset of graphs in H_2 that are strongly connected. That is, we choose E according to H_2, and if E does not contain a cycle that traverses all the vertices, we choose again until we pick an E which does contain such a cycle. It is easy to see that with probability at least one half, E is strongly connected. Why? Let H and I be the two Hamiltonian paths of which E is composed. If I contains a path from the end of H to the beginning of H, then E is strongly connected. Note that I contains such a path if and only if the reverse of I does not. Thus, for every graph E which is not strongly connected, there is at least one that is. Call this distribution H_2'

We will show that, with exponentially high probability, there does not exist a choice for the faulty processors such that E chosen from H_2 fails to induce a strongly connected component of size greater than $\frac{1}{13}|V|$ in the good processors. Since the space of H_2' contains at least one half the space of H_2, the same will hold for graphs E chosen according to H_2'. Since we are assuming that there are at most $\frac{1}{13}|V|$ faulty processors, we know that all the processors in the big strongly connected component must be good. Now, to find a faulty processor, we just follow a cycle through all the vertices as it leaves a good processor. Every processor that it certifies as good is a good processor. Thus, we can just follow the cycle until we find a good processor that certifies another processor as faulty. If all the tests along a cycle report good, then all the processors are good.

Applying Stirling's formula to Lemma 7 and taking logarithms, we find that

$$\begin{aligned} \ln \left(\binom{n}{\frac{1}{s}n} 2^{\frac{s-1}{s}n} \left(\frac{(n - \frac{s-2}{2s}n)!(n - \frac{1}{2}n)!}{(n)!(n - \frac{s-2}{2s} - \frac{1}{2}n)!} \right)^d \right) = \\ n \left(\frac{s-1}{s} \ln 2 - \frac{1}{s} \ln(\frac{1}{s}) - \frac{s-1}{s} \ln(\frac{s-1}{s}) \right. \\ \left. + d \left(\frac{s+2}{2s} \ln \frac{s+2}{2s} + \frac{1}{2} \ln \frac{1}{2} - \frac{1}{s} \ln \frac{1}{s} \right) \right) \\ - \frac{1}{2} \ln n + O(1) . \end{aligned}$$

Substituting in $s = 13$ and $d = 2$, we find that the coefficient of n is negative, so the probability is exponentially small. ∎

The number 1/13 could be replaced by a slightly larger real number in the analysis above.

6. Using Expanders

Expander graphs have the "large component" property that we make use of in our fault detection algorithm. Thus, we can make our fault detection algorithm constructive with a moderate decrease in efficiency.

Let G be a d-regular graph. We define the adjacency matrix of G to be a matrix T whose (i, j)-th entry is $\frac{1}{d}$ if G contains an edge between vertex i and vertex j, and which is 0 otherwise. We have chosen this definition so that the largest eigenvalue of T is 1. All constant vectors are eigenvectors for this eigenvalue. We show that if the second largest eigenvalue of the adjacency matrix of a graph is small, then the graph has the "large component" property.

Lemma 9. *Let $G = (V, E)$ be a d-regular graph on n vertices and let T be its adjacency matrix. Assume that all the eigenvalues of T except the greatest have absolute value at most λ. Then, for any two disjoint subsets A*

and B of V where $|A| = an$ and $|B| = bn$ and

$$\frac{\sqrt{ab}}{\sqrt{(1-a)(1-b)}} > \lambda ,$$

G contains an edge between some vertex in A and some vertex in B.

Proof: We use the following inner product on length-n vectors:

$$\langle \vec{x}, \vec{y} \rangle = \frac{1}{n} \sum_{i=1}^{n} \vec{x}(i)\vec{y}(i) .$$

Assume $V = \{1, \ldots, n\}$, and let $\vec{\chi}_A$ be the characteristic vector of the set A and $\vec{\chi}_B$ be the characteristic vector of the set B. Assume by way of contradiction that G contains no edge between A and B. It then follows that

$$\langle T(\vec{\chi}_A), \vec{\chi}_B \rangle = 0 .$$

We now write $\vec{\chi}_A = \vec{a} + (\vec{\chi}_A - \vec{a})$ and $\vec{\chi}_B = \vec{b} + (\vec{\chi}_B - \vec{b})$, where $\vec{a}$ and $\vec{b}$ denote the constant vectors that are all a's and b's, respectively. We also note that $\sum_{i=1}^{n}(\vec{\chi}_A(i) - \vec{a}(i)) = 0$ and $\sum_{i=1}^{n}(\vec{\chi}_B(i) - \vec{b}(i)) = 0$, which implies that $\vec{\chi}_A - \vec{a}$ and $\vec{\chi}_B - \vec{b}$ are perpendicular to the first eigenvector of T. Since $\vec{\chi}_A - \vec{a}$ has no component in the direction of the first eigenvector, we have that

$$||T(\vec{\chi}_A - \vec{a})|| \leq \lambda ||\vec{\chi}_A - \vec{a}|| ,$$

which implies

$$\begin{aligned} \left| \left\langle T(\vec{\chi}_A - \vec{a}), \vec{\chi}_B - \vec{b} \right\rangle \right| &\leq \lambda ||\vec{\chi}_A - \vec{a}|| ||\vec{\chi}_B - \vec{b}|| \\ &= \lambda \sqrt{ab(1-a)(1-b)} . \end{aligned}$$

We will need two more equalities. First, we see that

$$\left\langle T(\vec{a}), (\vec{\chi}_B - \vec{b}) \right\rangle = 0$$

because $\vec{a}$ is an eigenvector for 1 and $(\vec{\chi}_B - \vec{b})$ is perpendicular to all eigenvectors for 1. Second, the equation

$$\left\langle T(\vec{\chi}_A - \vec{a}), \vec{b} \right\rangle = 0$$

follows because $\vec{\chi}_A - \vec{a}$ is perpendicular to eigenvectors for 1, which implies that $T(\vec{\chi}_A - \vec{a})$ is perpendicular to eigenvectors for 1, but $\vec{b}$ is an eigenvector for 1.

Combining these equations with the inequality, we obtain our contradiction:

$$\begin{aligned} 0 &= \langle T(\vec{\chi}_A), \vec{\chi}_B \rangle \\ &= \left\langle T(\vec{a}), \vec{b} \right\rangle + \left\langle T(\vec{a}), (\vec{\chi}_B - \vec{b}) \right\rangle \\ &+ \left\langle T(\vec{\chi}_A - \vec{a}), \vec{b} \right\rangle + \left\langle T(\vec{\chi}_A - \vec{a}), (\vec{\chi}_B - \vec{b}) \right\rangle \\ &= ab + \left\langle T(\vec{\chi}_A - \vec{a}), (\vec{\chi}_B - \vec{b}) \right\rangle \\ &\geq ab - \lambda \sqrt{ab(1-a)(1-b} \\ &> 0 \end{aligned}$$

because we assumed that $\frac{\sqrt{ab}}{\sqrt{(1-a)(1-b)}} > \lambda$. ∎

Theorem 10. *There exists a constructive algorithm that will determine, for infinitely many values of n, which of n processors are good and which processors are faulty in 84 rounds of testing.*

Proof: Lubotzky, Phillips, and Sarnak [8] and independently Margulis [10] construct certain Cayley graphs which have second-largest eigenvalue $\frac{2\sqrt{p}}{p+1}$ and degree $p+1$ whenever p is congruent to 1 modulo 4. (The number of vertices is $q(q^2-1)/2$ where q is any prime that is congruent to 1 modulo 4 and p is a quadratic non-residue mod q.) For $p = 37$, $a = \frac{1}{4} + \frac{1}{14}$ and $b = \frac{1}{4} - \frac{1}{14}$ we have that

$$\frac{\sqrt{ab}}{\sqrt{(1-a)(1-b)}} > \frac{2\sqrt{p}}{p+1} .$$

Thus, by applying the techniques of Lemma 2, we see that the graphs corresponding to $p = 37$ will induce at least one connected component of size greater than $n/7$ on the good processors. This graph is contained in the union of 78 directed matchings, so we can perform all the tests that it indicates in at 78 rounds. By Lemma 5 we need at most 6 more rounds to complete the testing. ∎

Incidentally, Alon and Chung [1] have proved that expanders induce a long path on each sufficiently large subset of the vertices. This observation could have been used to obtain constructively a deterministic testing algorithm that runs in about 1000 rounds.

A. The Final Four Rounds of Testing

Lemma 11. *Let V be a set of n processors and let V' be the subset of good processors. Assume that E is a graph on V that induces exactly one strongly connected component of size greater than $\frac{1}{6}n$ on V' and induces at most two strongly connected components of size greater than $\frac{1}{6}n$ on $V - V'$. Then we can determine which processors are good and which processors are faulty by performing the tests indicated by E and then performing four more rounds of adaptive tests.*

Proof: Throughout the proof we will write *component* to mean "strongly connected component of V induced by E," and *big component* to mean "component of size greater than $\frac{1}{6}n$." Observe that every component contains only good processors or only faulty processors. We call a set of processors *good* if every processor in it is good; we call it *faulty* if every processor in it is faulty.

Case I: E induces exactly one big component.

Comment. Let $b = \lceil \frac{1}{6}n \rceil$. Let B consist of exactly b processors from the big component, and let X consist of exactly $2b$ of the remaining processors. Let $y = n - 3b$, and observe that $y \leq 3b$. Let Y_1 consist of $\lceil \frac{1}{2}y \rceil$ of the remaining y processors, and Y_2 the other $\lfloor \frac{1}{2}y \rfloor$ of them. (Note that if the big component contains more than $\lceil \frac{1}{6}n \rceil$, then the extras will be placed into X, Y_1, or Y_2.)

Rounds 1 and 2. The processors in B test all of the processors in X. Simultaneously, the processors in Y_1 and Y_2 test each other according to a maximal undirected bipartite matching.

Comment. Let G be the set of good processors found in X, and let $g = |G|$. $B \cup X$ contains at least one-half of the processors, and exactly $b + g$ of them are good. Therefore, because a majority of all the processors are good, fewer than $b + g$ of the processors in $Y_1 \cup Y_2$ can be faulty.

We will refer to the edges in the undirected bipartite matching between Y_1 and Y_2 as the Y_1–Y_2 *edges*. A Y_1–Y_2 edge is called a *happy* edge if both processors on it report that the other is good; it is called an *unhappy* edge otherwise. Call a processor *happy* if it is on a happy edge; call it *unhappy* if it is on an unhappy edge. If $y_1 > y_2$, then one processor in Y_1 is unmatched; it is neither happy nor unhappy. Call a Y_1–Y_2 edge a *good-good* edge if both processors on it are good. Call it a *faulty-faulty* edge if both processors on it are faulty. We will retain this terminology in Case 2. Clearly

- Every good-good edge is a happy edge.
- Every happy edge is either a good-good edge or a faulty-faulty edge.
- Every unhappy edge contains at least one faulty processor.

Because there are fewer than $b + g$ faulty processors among Y_1 and Y_2 there must be more than $\lfloor \frac{1}{2}y \rfloor - b - g$ good-good edges. Let h be the number of happy edges. Then $h > \lfloor \frac{1}{2}y \rfloor - b - g$.

Subcase a: $h \geq b$.

Round 3.a. The processors in B test the first b happy processors in Y_1

Round 4.a. The processors in B test the remaining $h - b$ happy processors in Y_1 and all of the $y - 2h$ unhappy or unmatched processors.

Comment. Note that Round 4.a involves at most $y - h - b \leq y - 2b \leq b$ tests.

Subcase b: $h < b$.

Round 3.b. The processors in $B \cup G$ test each happy processor in Y_1 and also test an additional $b + g - h$ unhappy processors.

Comment. Let G' consist of the good processors found on happy edges (both endpoints), and let $g' = |G'|$. Then $g' > 2(\lfloor \frac{1}{2}y \rfloor - b - g)$, so $g' \geq y - 2b - 2g$. Therefore $b + g + g' \geq y - b - g$.

Round 4.b. The processors in $B \cup G \cup G'$ test the remaining $y - 2h - (b + g - h) = y - b - g - h$ unhappy or unmatched processors.

Comment. Round 4.b can be performed because $b + g + g' \geq y - b - g \geq y - b - g - h$.

Case II: E induces exactly two big components.

Comment. Observe that one of the big components contains only good processors and the other big component contains only faulty processors. Let $b = \lceil \frac{1}{6}n \rceil$. Let B_1 consist of exactly b processors from the first big component, and let B_2 consist of exactly b processors from the second big component. Let $x = \lceil \frac{1}{2}(n - 2b) \rceil$, i.e., x is the least number that is not smaller than one-half of the number of remaining processors. Let X consist of exactly x of the remaining processors. Let $y = n - 2b - x$. Let Y_1 consist of $\lceil \frac{1}{2}y \rceil$ of the remaining processors, and let Y_2 consist of the $\lfloor \frac{1}{2}y \rfloor$ others. Observe that $2b \geq x \geq y \geq x - 1$.

Rounds 1 and 2. The processors in B_1 and B_2 test all of the processors in X. Simultaneously, the processors in Y_1 and Y_2 test each other according to a maximal undirected matching. Let X_i be the set of processors that B_i labels good in X, and let $x_i = |X_i|$, for $i = 1, 2$.

Subcase a: $x_1 > \frac{1}{2}x$, and X_1 and X_2 are not disjoint.

Comment. Let $p \in X_1 \cap X_2$. Then processor p must be good.

Round 3.a. Processor p tests one processor in B_1. Simultaneously, the processors in B_2 test all the processors in Y_2.

Comment. If the result of p's test is "good," then all processors in B_1 and therefore all processors in X_1 are good. Otherwise all processors in B_2 are good.

Round 4.a. If B_1 is good, then the processors in $B_1 \cup X_1$ test the processors in $Y_1 \cup Y_2$. Otherwise the processors in B_2 test the processors in Y_1.

Subcase a′: $x_2 > \frac{1}{2}x$, and X_1 and X_2 are not disjoint.

Swap B_1 and B_2, and then proceed as in Subcase a.

Subcase b: $x_1 \leq \frac{1}{2}x$ and $x_2 \leq \frac{1}{2}x$.

Comment. Then at most one-half of the processors in $B_1 \cup B_2 \cup X$ are good. Therefore a majority of the processors in $Y_1 \cup Y_2$ are good. Because every unhappy edge contains at least one faulty processor, a majority of the unmatched and happy processors must be good.

Round 3.b. The happy and unmatched processors in Y_1 test some of the processors in B_1; the remaining processors in B_1 test the unhappy processors in Y_1. The processors in B_2 test the unhappy processors in B_2.

Comment We give two votes to each happy processor in Y_1 (so it can cast a proxy for its neighbor in Y_2 as well) and one vote to the unmatched processor (if there is one). Because a majority of the happy and unmatched processors in $Y_1 \cup Y_2$ are good, the majority vote will determine whether B_1 is good or faulty, and consequently whether B_2 is good or faulty. If B_i is good, then Round 3b also diagnoses the unhappy processors in Y_i.

Round 4.b. Let B_i be the good big component as determined in Round 3b. The processors in B_i test the processors in Y_{3-i} and the unmatched processor in Y_1 if there is one.

Comment. Note that the total number of tests in Round 4.b is $\lceil \frac{1}{2}y \rceil \leq b$. The status of the unhappy processors in Y_i was determined in Round 3b. The status of happy processors in Y_i is the same as their neighbors in Y_{3-i}.

Subcase c: $x_1 > \frac{1}{2}x > x_2$, and X_1 and X_2 are disjoint.

Comment. Because exactly one-half of the processors in $B_1 \cup B_2$ are good, a majority of the processors in $X \cup Y_1 \cup Y_2$ must be good. Since $x \geq y$, more than y of the processors in $X \cup Y_1 \cup Y_2$ are good.
Suppose that B_2 is good. Then exactly x_2 of the processors in X are good. Therefore more than $y - x_2$ of the processors in $Y_1 \cup Y_2$ are good. Therefore more than $y - x_2 - \lceil \frac{1}{2}y \rceil = \lfloor \frac{1}{2}y \rfloor - x_2$ of the Y_1–Y_2 edges are good-good.
Suppose that B_2 is faulty. Then exactly x_1 of the processors in X are good. Therefore more than $y - x_1$ of the processors in $Y_1 \cup Y_2$ are good.

Subsubcase 1: At most $\lfloor \frac{1}{2}y \rfloor - x_2$ of the Y_1–Y_2 edges are happy.

Comment. Then at most $\lfloor \frac{1}{2}y \rfloor - x_2$ of them are good-good, so we may conclude that B_2 is faulty and therefore $B_1 \cup X_1$ is good.

Rounds 3.c.1 and 4.c.1. The processors in $B_1 \cup X_1$ test the processors in $Y_1 \cup Y_2$.

Subsubcase 2: More than $\lfloor \frac{1}{2}y \rfloor - x_2$ of the Y_1–Y_2 edges are happy.

Comment. Because B_1 says that X_2 is faulty and B_2 says that X_1 is good, X_2 is good if and only if B_2 is good.

Round 3.c.2. Let the processors in Y_2 and the unhappy and unmatched processors in Y_1 test some of the processors in $B_2 \cup X_2$.

Comment. (Round 3.c.2 can be performed because there are at most x_2 unhappy or unmatched processors in Y_1.) Count two votes for each happy processor that performs a test and one vote for each unhappy or unmatched processor that performs a test. The total of the "good" votes is at least as large as the number of good processors in $Y_1 \cup Y_2$ that would report that B_2 is good. The total of the "faulty votes is at least as large as the number of good processors in $Y_1 \cup Y_2$ that would report that B_2 is faulty.
If B_2 were in fact good, then more than $y - x_2$ of the processors in $Y_1 \cup Y_2$ would be good, and they would report that B_2 is good. If B_2 were in fact faulty, then more than $y - x_1$ of the processors in $Y_1 \cup Y_2$ would be good, and they would report that B_2 is faulty. These events cannot both occur because, then there would be more than $y - x_1 + y - x_2 \geq 2y - x \geq y$ votes, but there are exactly y votes. Thus B_2 is good if and only if there are more than $y - x_2$ votes for "good."

Round 4.c.2. If B_2 is good, then the processors in $B_2 \cup X_2$ test the processors in Y_2 and the unhappy and unmatched processors in Y_1. If B_1 is good, then the processors in $B_1 \cup X_1$ test the processors in $Y_1 \cup Y_2$.

Subcase c′: $c_2 > \frac{1}{6} > c_1$. Swap B_1 and B_2, and then proceed as in Subcase c.

Case III: E induces exactly three big components.

Comment. Observe that one of the big components contains only good processors and the other two big components contain only faulty processors. Let $b = \lceil \frac{1}{6}n \rceil$. Let B_i consist of exactly b processors from the ith big component, for $i = 1, 2, 3$. Let $x = n - 3b$. Let X consist of the remaining x processors.

Rounds 1, 2, and 3. The processors in B_i test the processors in X for $i = 1, 2, 3$.

Comment. If B_i is good, then the other big components must be faulty, so B_i must report that more than $\frac{1}{2}n - b$ of the processors in X are good. Because $b \geq \frac{1}{6}n$, we have

$$\frac{1}{2}n - b \geq \frac{2}{3}(n - 3b) = \frac{2}{3}x\,,$$

so each contender reports that strictly more than two-thirds of the processors in X are good.

Subcase a: exactly one big component is a contender.

Comment. That contender is the unique good big component, so no more testing is necessary.

Subcase b: exactly two big components are contenders.

Comment. There is a processor p that both contenders report is good. Then p must be good.

Round 4.b. Processor p tests one processor from one of the contenders.

Comment. The result of that test determines which contender is good.

Subcase c: exactly three big components are contenders.

Comment. There is a processor p that all three contenders report is good. Then p must be good. There is another processor q that B_2 and B_3 both report is good. If B_1 is faulty, then q is good.

Round 4.c. Processor p tests one processor from B_1, and processor q tests one processor from B_2.

Comment. The result of p's test determines whether B_1 is good. If B_1 is not good, then the result of q's test determines which of B_2 and B_3 is good.

Observe that E cannot induce four or more big components because three of them would be faulty. ∎

Acknowledgments. We would like to thank Warren Smith and Steven Salzberg for helpful discussions.

References

[1] N. Alon and F. R. Chung. Explicit construction of linear sized tolerant networks. *Discrete Math.*, 72:15–19, 1988.

[2] R. Beigel, R. Kosaraju, and G. Sullivan. Locating faults in a constant number of parallel testing rounds. In *Proc. SPAA*, pp. 189–198, 1989.

[3] P. M. Blecher. On a logical problem. *Discrete Math.*, 43:107–110, 1983.

[4] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications.* North Holland, New York, 1976.

[5] A. T. Dahbura and G. M. Masson. An $O(n^{2.5})$ fault identification algorithm for diagnosable systems. *IEEE Trans. Comput.*, 1984.

[6] S. L. Hakimi and K. Nakajima. On adaptive system diagnosis. *IEEE Trans. Comput.*, C-33(3):234–240, 1984.

[7] S. L. Hakimi and E. F. Schmeichel. An adaptive algorithm for system level diagnosis. *J. of Alg.*, 5:526–530, 1984.

[8] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

[9] U. Manber. System diagnosis with repair. *IEEE Trans. Comput.*, C–29:934–937, 1980.

[10] G. A. Margulis. Explicit group-theoretical constructions of combinatorial schemes and their application to the design of expanders and concentrators. *Problems of Information Transmission*, 24:39–46, 1988. Translated from *Problemy Peredachi Informatsii*, Vol. 24, No. 1, pp. 51–60, Jan-Mar, 1988.

[11] E. M. Palmer. *Graphical Evolution.* John Wiley & Sons, New York, 1985.

[12] F. P. Preparata, G. Metze, and R. T. Chien. On the connection assignment problem of diagnosable systems. *IEEE Trans. Electron. Comput.*, EC–16:848–854, 1967.

[13] E. Schmeichel, S. Hakimi, M. Otsuka, and G. Sullivan. On minimizing testing rounds for fault identification. In *18th Int'l Symp. Fault-Tolerant Comput.*, 1988.

[14] W. Smith, 1989. Personal communication.

[15] G. F. Sullivan. A polynomial time algorithm for fault diagnosability. In *Proc. 25th FOCS*, pp. 148–156, 1984.