DIAGNOSIS OF WIRING NETWORKS: AN OPTIMAL RANDOMIZED ALGORITHM FOR FINDING CONNECTED COMPONENTS OF UNKNOWN GRAPHS*

WEIPING SHI† AND DOUGLAS B. WEST‡

Abstract. We want to find the vertex sets of components of a graph G with a known vertex set V and unknown edge set E. We learn about G by sending an oracle a query set $S \subseteq V$, and the oracle tells us the vertices connected to S. The objective is to use the minimum number of queries to partition the vertex set into components. The problem is also known as interconnect diagnosis of wiring networks in VLSI. We present a deterministic algorithm using $O(\min\{k, \lg n\})$ queries and a randomized algorithm using expected $O(\min\{k, \lg k + \lg\lg n\})$ queries, where n is the number of vertices and k is the number of components. We also prove matching lower bounds.

Key words. randomized algorithm, lower bound, fault diagnosis, graph, component, connection class.

AMS subject classifications. 68Q25, 68R10, 05C85, 05C40, 94C12

1. Introduction. We study how to find the vertex sets of components of an unknown undirected graph $G = (V, E)$ on a known vertex set V. Vertices u and v are connected if there is a path between them. The components of G are its maximal connected subgraphs. The connection relation is an equivalence relation on the vertex set V, and the vertex sets of the components are the equivalence classes of the connection relation, also called the *connection classes*. When we say "finding the components", we mean finding the connection classes. In our problem, we are given V but not E. We do not know the number of components or their sizes. The only operation we may use to obtain information about G is to query an oracle. For any query set $S \subseteq V$, the oracle will tell us $Q(S)$, the set of vertices connected to vertices of S:

$$Q(S) = \{v \in V : \text{ there exists } u \in S \text{ such that } u \text{ and } v \text{ are connected.}\}$$

Note that the response $Q(S)$ does not identify which vertex in S is connected to each vertex in $Q(S)$. The objective is to find the connection classes using the minimum number of queries.

This problem comes from the interconnect diagnosis of wiring networks of logic circuits. It has applications to design and testing of very large scale integration (VLSI), multi-chip module (MCM) and printed circuit board (PCB) systems [2, 3, 4, 5, 8]. A wiring network consists of a set of nets, each having one driver and one receiver. The logic value of a good net is controlled by its driver and observed by its receiver. When some nets are involved in a short fault, their receivers all receive the logical OR of the values of their drivers. To diagnose a wiring network, a test engineer sends a test vector of logical 0s and 1s from the drivers and observes the outputs from the receivers. Diagnosing a wiring network is the same as finding the

* Research of the first author was supported by NSF grant MIP-9309120. Research of the second author was supported by NSA/MSP grant MDA904-93-H-3040.

†Department of Computer Science, University of North Texas, Denton, Texas 76203 (wshi@cs.unt.edu).

‡Department of Mathematics, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801 (west@math.uiuc.edu).

connection classes of the graph of short faults, and applying test vectors to the nets is the same as querying the oracle for that graph.

Kautz [5] studied the problem for the special case of testing $G = \overline{K}_n$, which he phrased as testing whether there is any short among n nets. Garey, Johnson and So [3] observed that if we are given partial information about the edge set of G, then finding an optimal algorithm to test $G = \overline{K}_n$ becomes NP-complete (reduction from chromatic number). For our problem of finding all connection classes, Jarwala and Yau [4] provided a heuristic using $\lg n + n - k$ queries, where k is the number of components. There is also a non-adaptive version of the problem where the inputs of all queries are decided before asking the oracle any question [2, 8, 9]. The non-adaptive version is used in applications where the query set is built into the computer hardware and the test is performed automatically. Shi and Fuchs [8] showed that $n-1$ queries are necessary and sufficient to find all connection classes nonadaptively. Shi and Fuchs also presented a recursive version of the deterministic algorithm of Section 2, for the case of the interconnect diagnosis problem. Cheng, Lewandowski and Wu [2] studied a variation of the non-adaptive diagnosis problem where the objective is to report all vertices in components that contain more than one vertex, without having to identify the connection classes. Chen and Hwang [1] studied the problem under a different model, called group testing. In group testing, the inputs of each query are two sets S and T, and the oracle answers yes or no depending on whether some vertex in S is connected to some vertex in T. Kavraki, Latombe, Motwani and Raghavan [6] studied the problem of finding connection classes of an unknown graph, where the oracle looks at one entry of the adjacency matrix in each query. Recently, Shi and West [9] studied how to find the connection classes if partial information about the edge set of G is given.

Table 1 summaries our results, where n is the number of vertices and k is the number of components, which is not given as part of the input. No assumption is made on k other than $1 \le k \le n$. All logarithms in this paper are base 2. We measure the query complexity in terms of both the input size n and the number of components k. We present algorithms achieving the upper bounds and prove matching lower bounds. Note that randomization may permit an exponential reduction in the number of queries compared to the deterministic algorithm.

TABLE 1.1
Number of queries required to find connection classes.

Deterministic	$\Theta(\min\{k, \lg n\})$
Randomized	$\Theta(\min\{k, \lg k + \lg\lg n\})$
Nondeterministic	$\Theta(\lg k)$

2. Deterministic Algorithm. There is a straightforward deterministic algorithm to find the connection classes in k queries: Iteratively pick a vertex v and make a query. Record $Q(\{v\})$ as one class, and delete $Q(\{v\})$ from the vertex set. Each query finds one new class. The upper bound k can be improved when $n < 2^k$. We present in Algorithm 1 and Procedure $\mathcal{D}$ an algorithm that uses $\lceil \lg n \rceil$ queries to find all connection classes. Procedure $\mathcal{D}$ will also be used by the randomized algorithm in Section 3.

The algorithm iteratively maintains a *component structure* $P = \{(S_i, R_i) : i = 1, 2, \ldots, t\}$, where $S_1, S_2, \ldots, S_t$ are subsets of vertices that form a partition of V, and

$R_1, R_2, \ldots, R_t$ are "chosen" subsets of vertices such that $R_i \subseteq S_i$ and $Q(R_i) = S_i$. This property of the chosen subsets implies that each S_i is a union of connection classes. Algorithm 1 initializes with the component structure $P = \{(V, V)\}$, and then makes $\lceil \lg n \rceil$ calls to Procedure $\mathcal{D}$ to refine the partition, cutting the sizes of the chosen sets in half at each step. When the algorithm terminates, every R_i is reduced to a single vertex, and therefore every S_i is one connection classes.

Algorithm 1.
Input: A vertex set V of size n.
Output: Vertex sets of all components of unknown G.
1: $P \leftarrow \{(V, V)\}$.
2: **For** $i \leftarrow 1$ **to** $\lceil \lg n \rceil$ **do**
3: $P \leftarrow \mathcal{D}(P)$.
4: **For** each $(S_j, R_j) \in P$
5: Report S_j as one class.

Procedure $\mathcal{D}(P)$.
Input: A component structure $P = \{(S_j, R_j) : j = 1, 2, \ldots, t\}$.
Output: A refined component structure $P' = \{(S'_j, R'_j) : j = 1, 2, \ldots, t'\}$.
1: **For** $j \leftarrow 1$ **to** t **do**
2: Arbitrarily pick $R'_j \subseteq R_j$ such that $|R'_j| = \lceil \frac{1}{2}|R_j| \rceil$.
3: Perform the single query $\cup R'_j$, with result $S' \leftarrow Q(\cup R'_j)$.
4: $P' \leftarrow \emptyset$.
5: **For** $j \leftarrow 1$ **to** t **do**
6: Let $T_j = S_j \cap S'$.
7: $P' \leftarrow P' \cup \{(T_j, R'_j)\}$.
8: **If** $T_j \neq S_j$ **then** $P' \leftarrow P' \cup \{(S_j - T_j, R_j - T_j)\}$.
9: **Return** P'.

LEMMA 2.1. *Given a component structure in which the largest chosen subset has m vertices, $\lceil \lg m \rceil$ iterations of Procedure $\mathcal{D}$ finds all the connection classes.*

Proof. With each query, we divide the maximum size of the chosen subsets by 2. Hence there are at most $\lceil \lg m \rceil$ queries. To prove that the algorithm works it suffices to show that the property $Q(R_j) = S_j$ is maintained by Procedure $\mathcal{D}$. If so, then when all $|R_j| = 1$, all vertices in S_j are connected to the chosen vertex, and S_j is the connection class containing it.

For any $|R_j| > 1$, the procedure performs a query and splits S_j into T_j and $S_j - T_j$. Because $Q(R_j) = S_j$, the query $Q(\cup R'_j)$ tells us that $Q(R'_j) = T_j$ and there is no connection between T_j and $S_j - T_j$. If $S_j - T_j$ is nonempty, then since all of S_j were connected to R_j, we must have $Q(R_j - T_j) = S_j - T_j$. □

It follows immediately,

THEOREM 2.2. *For any graph with n vertices, Algorithm 1 finds all connection classes using $\lceil \lg n \rceil$ queries.*

An algorithm using $2(\min\{k, \lceil \lg n \rceil\})$ queries can be obtained by combining the k-query algorithm and the $\lceil \lg n \rceil$-query algorithm: We alternately perform one query for each of the two algorithms, stopping whenever one of the two algorithms has found all the connection classes.

3. Randomized Algorithm. In this section, we present a randomized algorithm that may reduce the number of queries exponentially. The algorithm first calls the randomized Procedure $\mathcal{R}$ for $\lceil \lg \lg n \rceil$ iterations, and then it calls the deterministic Procedure $\mathcal{D}$ to complete the refinement of the partition.

In Algorithm 2 we also maintain a partition and chosen subsets, and the idea is still to partition S_j into T_j connected to R'_j, and $S_j - T_j$ connected to $R_j - T_j$. The only difference between Procedure $\mathcal{R}$ and Procedure $\mathcal{D}$ is in step 2, where we randomly pick $\lceil \sqrt{|R|} \rceil$ vertices in $\mathcal{R}$, instead of $\lceil \frac{1}{2}|R| \rceil$ vertices in $\mathcal{D}$. This difference leads to more rapid reduction in the size of the query set.

Algorithm 2.
Input: A vertex set V of size n.
Output: Vertex sets of all components of unknown G.
1: $P \leftarrow \{(V, V)\}$.
2: **For** $i \leftarrow 1$ **to** $\lceil \lg \lg n \rceil$ **do**
3: $P \leftarrow \mathcal{R}(P)$.
4: **While** there exists $(S_j, R_j) \in P$ such that $|R_j| > 1$
5: $P \leftarrow \mathcal{D}(P)$.
6: **For** each $(S_j, R_j) \in P$
7: Report S_j as one class.

Procedure $\mathcal{R}(P)$.
Input: A component structure $P = \{(S_j, R_j) : j = 1, 2, \ldots, t\}$.
Output: A refined component structure $P' = \{(S'_j, R'_j) : j = 1, 2, \ldots, t'\}$.
1: **For** $j \leftarrow 1$ **to** t **do**
2: Randomly pick $R'_j \subseteq R_j$ such that $|R'_j| = \lceil \sqrt{|R_j|} \rceil$.
3: Perform the single query $\cup R'_j$, with result $S' \leftarrow Q(\cup R'_j)$.
4: $P' \leftarrow \emptyset$.
5: **For** $j \leftarrow 1$ **to** t **do**
6: Let $T_j = S_j \cap S'$.
7: $P' \leftarrow P' \cup \{(T_j, R'_j)\}$.
8: **If** $T_j \neq S_j$ **then** $P' \leftarrow P' \cup \{(S_j - T_j, R_j - T_j)\}$.
9: **Return** P'.

The correctness of Algorithm 2 follows as in Lemma 2.1. To estimate the number of queries used by Algorithm 2, we need to estimate the size of the largest chosen subset after $\lceil \lg \lg n \rceil$ calls to Procedure $\mathcal{R}$. Define a sequence of random variables $X_0, X_1, \ldots, X_{\lg \lg n}$, where $X_0 = n$, and X_i is the size of the largest chosen subset after i calls to Procedure $\mathcal{R}$.

LEMMA 3.1. *For any positive integer m and real number $\alpha \in [0, 1]$,*

$$\alpha(1 - \alpha)^m < \frac{1}{em},$$

where $e = 2.71828\ldots$ is the base of the natural logarithm.

Proof. The derivative of $f(\alpha) = \alpha(1 - \alpha)^m$ is

$$\frac{df(\alpha)}{d\alpha} = (1 - \alpha)^m - m\alpha(1 - \alpha)^{m-1} = (1 - \alpha)^{m-1}(1 - \alpha - m\alpha).$$

If $f'(\alpha) = 0$, then one of the factors $(1-\alpha)$ and $(1-\alpha-m\alpha)$ must be zero, which happens only at $\alpha = 1$ or $\alpha = \frac{1}{m+1}$. Checking the values of f at these points and the boundary $\alpha = 0$, we find that $f(\alpha)$ reaches its maximum when $\alpha = \frac{1}{m+1}$. Therefore,

$$f(\alpha) \le \frac{1}{m+1}\left(1-\frac{1}{m+1}\right)^m < \frac{1}{m+1}\frac{1}{\left(1-\frac{1}{m+1}\right)}\frac{1}{e} = \frac{1}{em}.$$

□

LEMMA 3.2. *For the sequence of random variables X_i defined above, the conditional expectation $E[X_i \mid X_{i-1}]$ is bounded by*

$$E[X_i \mid X_{i-1}] < \sqrt{X_{i-1}} \cdot k^2.$$

Proof. Assume after $i-1$ calls to $\mathcal{R}$, we have a component structure $P = \{(S_j, R_j) : j = 1, 2, \ldots, t\}$ in which the largest chosen subset is R_l of size X_{i-1}. At iteration i when we apply $\mathcal{R}(P)$, the pair (S_l, R_l) is split into $(Q(R'_l), R'_l)$, and possibly $(S_l - Q(R'_l), R_l - Q(R'_l))$, where $|R'_l| = \sqrt{X_{i-1}}$. Denote the chosen subset $R_l - Q(R'_l)$ by R''_l. The size of R''_l is a random variable with its value anywhere from 0 to $X_{i-1} - \sqrt{X_{i-1}}$. Consider the conditional expectation:

$$\begin{aligned} E[\max\{|R'_l|, |R''_l|\} \mid X_{i-1}] &= E[\max\{\sqrt{X_{i-1}}, |R''_l|\} \mid X_{i-1}] \\ &\le E[\sqrt{X_{i-1}} + |R''_l| \mid X_{i-1}] \\ &= \sqrt{X_{i-1}} + E[|R''_l| \mid X_{i-1}]. \end{aligned}$$

Let the k components of G be $G_1, G_2, \ldots, G_k$, and their contributions to R_l have sizes $n_1, n_2, \ldots, n_k$, with $n_j = |V(G_j) \cap R_l|$. The vertices in R''_l belong to the components not hit by the query $Q(R'_l)$. Each successive vertex selected for R'_l has probability at least $n_j/|R_l|$ of belonging to G_j. Hence the probability of missing G_j completely is at most $(1 - n_j/|R_l|)^{\sqrt{|R_l|}}$. When we do miss G_j, it contributes n_j vertices to R''_l. Therefore,

$$\begin{aligned} E[|R''_l| \mid X_{i-1}] &\le \sum_{j=1}^{k} n_j \left(1 - \frac{n_j}{X_{i-1}}\right)^{\sqrt{X_{i-1}}} \\ &= X_{i-1} \sum_{j=1}^{k} \frac{n_j}{X_{i-1}} \left(1 - \frac{n_j}{X_{i-1}}\right)^{\sqrt{X_{i-1}}}. \end{aligned}$$

Letting $\alpha_j = n_j/X_{i-1}$, we have $\sum_{j=1}^{k} \alpha_j = 1$ and $0 \le \alpha_j \le 1$ for $j = 1, 2, \ldots, k$. Using Lemma 3.1 we obtain

$$\begin{aligned} E[|R''_l| \mid X_{i-1}] &\le X_{i-1} \sum_{j=1}^{k} \alpha_j (1-\alpha_j)^{\sqrt{X_{i-1}}} \\ &\le X_{i-1} \cdot k \cdot \max_{0 \le \alpha \le 1} \alpha(1-\alpha)^{\sqrt{X_{i-1}}} \\ &< \frac{X_{i-1} \cdot k}{e\sqrt{X_{i-1}}} = \frac{k\sqrt{X_{i-1}}}{e}. \end{aligned}$$

This computation depends only on $|R_l|$, which equals X_{i-1}. Because the expression grows with $|R_l|$, it also provides an upper bound on the expected size of the larger piece in the refinement of some other R_j. Also, although we don't know which piece will provide the new maximum, certainly its size will be less than the sum of maximum sizes from all the pieces. These two comments yield an upper bound on $E[X_i \mid X_{i-1}]$ by taking k times the bound from R_l:

$$\begin{aligned} E[X_i \mid X_{i-1}] &\le k \cdot E[\max\{|R_l'|, |R_l''|\} \mid X_{i-1}] \\ &< k \cdot \left(\sqrt{X_{i-1}} + \frac{k\sqrt{X_{i-1}}}{e} \right). \end{aligned}$$

For $k \ge 2$, we conclude that $E[X_i \mid X_{i-1}] < \sqrt{X_{i-1}} \cdot k^2$. □

THEOREM 3.3. *For a graph with n vertices and k components, the expected number of queries used by Algorithm 2 to find all connection classes is $O(\lg k + \lg\lg n)$.*

Proof. Consider the maximum size X_i of the chosen subsets after i iterations. We prove by induction on i that $E[X_i] < n^{2^{-i}} k^4$. This is immediate for $i = 0$. Using Lemma 3.2, the inequality $E[f(X)] \le f(E[X])$ for concave f (such as square root), and the induction hypothesis, we have

$$\begin{aligned} E[X_i] = E(E[X_i|X_{i-1}]) &< E[\sqrt{X_{i-1}} \cdot k^2] \le \sqrt{E[X_{i-1}]} \cdot k^2 \\ &\le \sqrt{n^{2^{-(i-1)}} k^4} \cdot k^2 = n^{2^{-i}} k^4. \end{aligned}$$

With $i = \lceil \lg\lg n \rceil$, we obtain $E[X_{\lg\lg n}] < 2k^4$. From Markov's inequality,

$$\Pr[X_{\lg\lg n} \ge 2k^4 \lg n] < \frac{2k^4}{2k^4 \lg n} = \frac{1}{\lg n}.$$

If $X_{\lg\lg n} < 2k^4 \lg n$, then $\lg(2k^4 \lg n)$ additional queries suffice. With probability at most $\frac{1}{\lg n}$, we are left with chosen subsets greater than $2k^4 \lg n$ after $\lceil \lg\lg n \rceil$ iterations of Procedure $\mathcal{R}$. Procedure $\mathcal{D}$ can resolve these instances with at most $\lceil \lg n \rceil$ further queries. Therefore the expected number of queries used by Algorithm 2 is at most

$$\lg\lg n + \lg(2k^4 \lg n) + \frac{1}{\lg n} \lg n = 2\lg\lg n + 4\lg k + 2.$$

□

Again, an algorithm using $O(\min\{k, \lg\lg n + \lg k\})$ queries can be obtained by alternating between the k-query algorithm and Algorithm 2.

4. Lower Bounds. In this section we obtain optimal lower bounds (within a constant multiplicative factor) on the number of queries for nondeterministic, deterministic, and randomized algorithms.

THEOREM 4.1. *For a graph with k components, every algorithm finding the connection classes uses at least $\lg k$ queries.*

Proof. Since the response to a query $Q(S)$ cuts each subset $U \subseteq V$ that is known to be a union of connection classes into at most two subsets $U \cap Q(S)$ and $U - Q(S)$ with no edges between them, we need at least $\lg k$ queries to separate the set of vertices into k classes. □

Theorem 4.1 provides a lower bound for every algorithm. A nondeterministic algorithm can guess the connection classes and use $\lceil \lg k \rceil + 1$ queries to verify them.

THEOREM 4.2. *Let A be a deterministic algorithm finding the connection classes of unknown graphs. Over graphs with n vertices and k components, in the worst case A uses at least $\min\{k, \lg n\}$ queries.*

Proof. Consider the following adversary. In response to our first query S, the adversary makes a graph in which S induces a single component if $|S| \leq n/2$, or a graph in which $V - S$ induces a single component if $|S| > n/2$. This leaves a subproblem with $k - 1$ components and at least $n/2$ vertices. □

The randomized lower bound is more involved. We first describe Yao's corollary [7] of von Neumann's minimax principle. By considering a matrix game in which the rows correspond to deterministic algorithms and the columns to input instances, Yao observed that the expected performance of the optimal randomized algorithm on the worst input instance for it equals the expected cost of the worst input distribution against the best deterministic algorithm for it. More precisely, let P denote a distribution over deterministic algorithms A, Q denote a distribution over input instances G, and $c(A, G)$ denote the cost of running algorithm A on input instance G, we have

$$\min_P \max_G E_P[c(A, G)] = \max_Q \min_A E_Q[c(A, G)].$$

Hence to provide a lower bound for a randomized algorithm, it suffices to prove a lower bound for the expectation of every deterministic algorithm against a particularly bad input distribution.

We will need several lemmas.

LEMMA 4.3. *(Randomized Reduction Lemma) For $i \in \{1, 2\}$, let $\mathcal{G}_i$ be a set of input instances, Π_i be a property for $\mathcal{G}_i$, and let A_i be a deterministic algorithm for testing $\mathcal{G}_i \in \Pi_i$. Let $B : \mathcal{G}_1 \to \mathcal{G}_2$ be a deterministic or randomized transformation such that $G_1 \in \Pi_1$ if and only if $B(G_1) \in \Pi_2$. Let Q_1 be a distribution over $\mathcal{G}_1$ of a random input G_1, and let Q_2 be the resulting distribution of $B(G_1)$. Under these conditions,*

$$\min_{A_1} E_{Q_1}[c(A_1, G_1)] \leq E_{Q_1}[c(B, G_1)] + \min_{A_2} E_{Q_2}[c(A_2, G_2)].$$

Proof. We may view B as an algorithm on the set $\mathcal{G}_1$. Our general cost function c uses an arbitrary cost measure; by $c(B, G_1)$ we mean the cost in this measure of performing the transformation. If B is deterministic, then combining B and A_2 gives a deterministic algorithm for recognizing Π_1. Thus the claim holds for deterministic transformations.

If B is randomized, then combining B and A_2 gives a randomized algorithm for Π_1. The performance of a randomized algorithm is a weighted average of the performance of deterministic algorithms, weighted by some distribution. The combination of B and A_2 costs at least as much as the best randomized algorithm; let P be the best distribution over deterministic algorithms. Since the expectation over P is a convex combination over deterministic algorithms, for each fixed input distribution Q_1 we have the inequality below, which completes the proof.

$$\min_{A_1} E_{Q_1}[c(A_1, G_1)] \leq \min_P E_P[E_{Q_1}[c(A_1, G_1)]].$$

□

We will apply Yao's corollary to a particular distribution $R(n, k)$ over graphs with $n + o(n)$ vertices and k components, where $k \leq \frac{1}{2} \lg \lg n$. To select a graph according

to $R(n,k)$, we form cliques $C_1, C_2, \ldots, C_k$ with vertex sets of sizes $n^{\epsilon_1}, n^{\epsilon_2}, \ldots, n^{\epsilon_k}$, respectively. We let $\epsilon_1 = 0$, $\epsilon_2 = 1$ and $\epsilon_3 = 1/2$. For $i \geq 4$, ϵ_i is chosen with the following distribution:

$$\Pr\left[\epsilon_i = \epsilon_{i-1} + \left(\frac{1}{2}\right)^{i-2}\right] = \Pr\left[\epsilon_i = \epsilon_{i-1} - \left(\frac{1}{2}\right)^{i-2}\right] = \frac{1}{2}. \tag{4.1}$$

This generates $\sum_{i=1}^{k} n^{\epsilon_i} = n + o(n)$ vertices in total. Finally, apply a random permutation to the vertex labels to complete the generation of a graph from $R(n,k)$. Figure 4.1 illustrates a selection of the exponents.

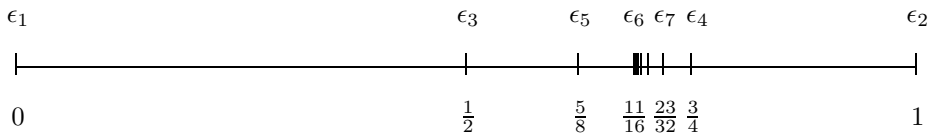

FIG. 4.1. *Example distribution of the exponents.*

In the above definition and the rest of this section, we omit the ceiling function $\lceil\ \rceil$ on the number of vertices for simplicity. To do so does not affect the claimed bound, since each time a ceiling is omitted, the number of vertices is affected by an additive term of at most 1, while the smallest component that might be affected is of size at least $\lg n$.

Given $\epsilon_1, \ldots, \epsilon_k$, let π be the permutation of $\{1, 2, \ldots, k\}$ such that $\epsilon_{\pi(1)} < \epsilon_{\pi(2)} < \cdots < \epsilon_{\pi(k)}$. We call each interval $(\epsilon_{\pi(i)}, \epsilon_{\pi(i+1)}]$ an *ϵ-interval*. The length of $(\epsilon_{\pi(i)}, \epsilon_{\pi(i+1)}]$ is $\epsilon_{\pi(i+1)} - \epsilon_{\pi(i)}$. The real numbers $\epsilon_{\pi(1)}, \ldots, \epsilon_{\pi(k)}$ partition $(0,1]$ into $k-1$ disjoint ϵ-intervals. Of the $k-1$ ϵ-intervals, there is one with length 2^{-i} for each $i = 1, 2, \ldots, k-3$, and there are two with length $2^{-(k-2)}$ that are consecutive.

LEMMA 4.4. *For any fixed integer $k \geq 3$ and fixed real number $x \in (0,1]$, under the distribution of $R(n,k)$, the probability that x is in an ϵ-interval of length 2^{-i} is 2^{-i} when $1 \leq i \leq k-3$, and it is $2^{-(i-1)}$ when $i = k-2$.*

Proof. When $k = 3$, every x is in an ϵ-interval of length $1/2$. When $k > 3$, x remains in the interval of length $1/2$ with probability $1/2$, and otherwise x falls into some interval determined by applying the process for $k-1$ steps to an interval of length $1/2$. Multiplying all lengths and probabilities by $1/2$ in that distribution yields the remainder of the specified distribution for k, so the claim holds by induction. □

LEMMA 4.5. *Given positive integers $x_1, x_2, \ldots, x_k$, let $P(x_1, \ldots, x_k)$ be the distribution over graphs consisting of disjoint cliques having $x_1, \ldots, x_k$ vertices that is obtained by assigning vertices to components at random. For any positive integers $n_1, \ldots, n_k, c$, the expected number of queries used by a deterministic algorithm that finds the connection classes against $P_2 = P(n_1 \cdot c, \ldots, n_k \cdot c)$ is at least the expected number of queries used by an optimal deterministic algorithm against $P_1 = P(n_1, \ldots, n_k)$.*

Proof. We exhibit a randomized algorithm B that transforms P_1 to a distribution P' without making any query. By Lemma 4.3, the expected number of queries against P' is at least the expected number against P_1. We then observe that the expected number of queries against P' is the same as the expected number used by the same algorithm against P_2.

Given G_1 drawn from P_1, for each vertex $v_i \in V_1$ we add a clique Q_i having $c-1$ vertices and an edge between v_i and Q_i. We then apply a random permutation σ to the resulting set of vertices to obtain a graph G_2. We have changed each component having n_i vertices in G_1 to a component having $c \cdot n_i$ vertices in G_2, and we did not add any components or make any queries.

Permutation σ guarantees that vertices are assigned to components at random in G_2. Therefore, the probability of a particular partition of the vertices into connection classes depends only on the sizes of the connection classes, which are fixed. Thus the distribution over answers to the connection-class problem is the same for P' as for P_2. Furthermore, the two problems are solved by the same algorithms and with the same expected number of queries, because instances with the same probabilities can be paired up so that the responses to all queries are exactly the same. □

LEMMA 4.6. *Let k be a positive integer satisfying $k \le \frac{1}{2}\lg\lg n$. If F satisfies*

$$F(n,k) \ge 1 + \sum_{i=1}^{k-2} 2^{-i}\left(1 - \frac{1}{\lg n}\right) F(n^{2^{-i}}, k-i-2)$$

for $k \ge 2$, then $F(n,k) \ge k/5$ for all $k \ge 2$.

Proof. If $k \le \frac{1}{2}\lg\lg n$, then

$$k - i - 2 < k - \frac{i}{2} \le \frac{\lg\lg n}{2} - \frac{i}{2} = \frac{\lg\lg n^{2^{-i}}}{2}.$$

Thus we can apply induction on k. When $k = 2$, since empty sums equal 0, the initial condition is $F(n,k) \ge 1 > k/5$. When $k > 2$, the induction hypothesis yields $F(n,i) \ge i/5$ for $2 \le i < k$. Now

$$F(n,k) \ge 1 + \sum_{i=1}^{k-2} \frac{1}{2^i}\left(1 - \frac{1}{\lg n}\right)\frac{k-i-2}{5}.$$

To evaluate the sum, we first prove by induction on j that $\sum_{i=1}^{j-2} 2^{-i}(j-i) = j-2$. This is trivial for $j = 2$ or $j = 3$. For $j > 3$, we have

$$\sum_{i=1}^{j-2}\frac{j-i}{2^i} = \sum_{i=1}^{j-3}\frac{j-1-i}{2^i} + \sum_{i=1}^{j-3}\frac{1}{2^i} + \frac{2}{2^{j-2}} = j-3+\left(1-\frac{1}{2^{j-3}}\right)+\frac{1}{2^{j-3}} = j-2.$$

Applying also $\sum_{i=1}^{k-2} 2^{1-i} < 2$, we obtain

$$\begin{aligned} F(n,k) &> 1 + \left(1 - \frac{1}{\lg n}\right)\left(\frac{k-4}{5}\right) \\ &> \frac{k}{5} + \frac{1}{5} - \frac{k-4}{5\lg n} \\ &> \frac{k}{5} + \frac{1}{5} - \frac{\lg\lg n}{10\lg n} > \frac{k}{5}. \end{aligned}$$

□

LEMMA 4.7. *For $k \le \frac{1}{2}\lg\lg n$, the expected number of queries used by any deterministic algorithm A finding connection classes against distribution $R(n,k)$ is $\Omega(k)$.*

Proof. Let V be the set of vertices and $S \neq \emptyset$ be the set of vertices algorithm A picks to make the first query. We consider three cases concerning S, each of which leads to the same recurrence.

Define x by $|S| = n^{1-x}$. In Cases 1 and 2, we suppose that $|S| < n$, and thus $x > 0$. Since $x \in (0,1]$, there exists an ϵ-interval $(\epsilon_{\pi(i)}, \epsilon_{\pi(i+1)}]$ containing x. By Lemma 4.4, the probability is 2^{-j} that the ϵ-interval containing x has length 2^{-j}; consider this possibility.

Intervals of length 2^{-j} are created when ϵ_{j+2} is selected, and ϵ_{j+2} lies between two such intervals. Further choices are made in one of those intervals, so ϵ_{j+2} is the bottom or top of the ϵ-interval of length 2^{-j} that remains. Thus $j+2$ is $\pi(i)$ or $\pi(i+1)$. We prove that in either case, with probability at least $1/\lg n$ we still have $C_{j+3}, \ldots, C_k$ within a single set of our partition.

Case 1. If $j+2 = \pi(i)$, then

$$\epsilon_{\pi(i)} - 2^{-j} < \epsilon_{j+3}, \epsilon_{j+4}, \ldots, \epsilon_k \leq \epsilon_{\pi(i-1)} < \epsilon_{\pi(i)} < x \leq \epsilon_{\pi(i+1)}.$$

Consider the probability that S contains no vertex from components $C_{j+3}, C_{j+4}, \ldots, C_k$. With $C = C_{j+3} \cup \cdots \cup C_k$, we have

$$\Pr[S \cap C = \emptyset] = \frac{\binom{|V|-|C|}{|S|}}{\binom{|V|}{|S|}} > \left(\frac{|V|-|C|-|S|+1}{|V|-|S|+1}\right)^{|S|} = \left(1 - \frac{|C|}{|V|-|S|+1}\right)^{|S|}.$$

Since $x > \epsilon_{\pi(i)} \geq 2^{-k+2} \geq 4/\sqrt{\lg n} > 1/\lg n$ and $n^{-1/\lg n} = 1/2$, we have $|S| < n/2$. Using $|V| - |S| + 1 > n/2$, $|C| < 2n^{\epsilon_{\pi(i-1)}}$, and $|S| = n^{1-x}$, we have

$$\Pr[S \cap C = \emptyset] > \left(1 - \frac{4n^{\epsilon_{\pi(i-1)}}}{n}\right)^{n^{1-x}} > 1 - \frac{4}{n^{x-\epsilon_{\pi(i-1)}}} > 1 - \frac{4}{n^{\epsilon_{\pi(i)}-\epsilon_{\pi(i-1)}}}.$$

The middle inequality uses $(1-y)^z > 1 - yz$ for $z > 1$ and $0 < yz < 1$, which is verified by taking natural logarithms and expansions of both sides.

Since $\epsilon_{\pi(i)} - \epsilon_{\pi(i-1)} \geq 2^{-k+2} \geq \frac{4}{\sqrt{\lg n}}$, we have

$$\Pr[S \cap C = \emptyset] > 1 - \frac{1}{n^{4/\sqrt{\lg n}}} = 1 - \frac{1}{2^{4\sqrt{\lg n}}} > 1 - \frac{1}{2^{\lg \lg n}} = 1 - \frac{1}{\lg n}.$$

Thus $1 - 1/\lg n$ is a lower bound on the probability that all of $C_{j+3}, \ldots, C_k$ lie in $V - Q(S)$.

Case 2. If $j+2 = \pi(i+1)$, then the various definitions yield

$$\epsilon_{\pi(i)} < x \leq \epsilon_{\pi(i+1)} < \epsilon_{\pi(i+2)} \leq \epsilon_{j+3}, \epsilon_{j+4}, \ldots, \epsilon_k < \epsilon_{\pi(i+1)} + 2^{-j}.$$

Let p be the probability that S intersects each of $C_{j+3}, \ldots, C_k$. The set S misses C_l with probability

$$\Pr[S \cap C_l = \emptyset] = \frac{\binom{|V|-|C_l|}{|S|}}{\binom{|V|}{|S|}} < \left(\frac{|V|-|C_l|}{|V|}\right)^{|S|} = \left(1 - \frac{n^{\epsilon_l}}{n + o(n)}\right)^{n^{1-x}}$$
$$< \left(1 - \frac{n^{\epsilon_l}}{2n}\right)^{n^{1-x}} \leq \exp\left(-\frac{n^{\epsilon_l - x}}{2}\right).$$

The last inequality uses $1 - z \leq e^{-z}$ for $0 \leq z \leq 1$. Avoiding each of the bad events separately yields

$$p \geq 1 - \sum_{l=j+3}^{k} \Pr[S \cap C_l = \emptyset] > 1 - k \Pr[S \cap C_{j+3} = \emptyset]$$
$$> 1 - k \exp\left(-\frac{n^{\epsilon_{j+3}-x}}{2}\right) \geq 1 - k \exp\left(-\frac{n^{\epsilon_{\pi(i+2)}-\epsilon_{\pi(i+1)}}}{2}\right).$$

Let $y = n^{\epsilon_{\pi(i+2)}-\epsilon_{\pi(i+1)}}$. Since $\epsilon_{\pi(i+2)} - \epsilon_{\pi(i+1)} \geq 2^{-k+2}$ and $k \leq \frac{1}{2}\lg\lg n$, we have $y > k$ and thus $ky < y^2 < e^{y/2}$ when y is sufficiently large, which yields $ke^{-y/2} < 1/y$. Therefore, $p > 1 - 1/n^{\epsilon_{\pi(i+2)}-\epsilon_{\pi(i+1)}}$. The same computation as in Case 1 now yields $p > 1 - 1/\lg n$.

Case 3. The remaining case is $|S| \geq n$; that is, $x \leq 0$. In Case 2, we proved that when $0 < x < 4/\sqrt{\lg n}$, the probability is at least $1 - 1/\lg n$ that S hits all of $C_{j+3}, \ldots, C_k$. When S becomes even larger, the probability of hitting all these components cannot decrease.

In each of Cases 1-3, we obtain the same recursive lower bound. Given the occurrence of one of these cases, with probability 2^{-j} the probability is at least $1 - 1/\lg n$ that the first query leaves us with components $C_{j+3}, C_{j+4}, \ldots, C_k$ all in $V - Q(S)$ or all in $Q(S)$. The numbers of vertices in $C_{j+3}, C_{j+4}, \ldots, C_k$ are $n^{\epsilon_{j+3}}, n^{\epsilon_{j+4}}, \ldots, n^{\epsilon_k}$, respectively. With $\delta = \epsilon_{j+2} - 2^{-j}$, the components of sizes $n^{\epsilon_{j+3}-\delta}$, $n^{\epsilon_{j+4}-\delta}$, …, $n^{\epsilon_k-\delta}$ have the distribution $R(n^{2^{-j}}, k-j-2)$. According to Lemma 4.5, solving the problem for the graph induced by $V - Q(S)$ or $Q(S)$ expects to use at least as many queries as solving it against $R(n^{2^{-j}}, k-j-2)$.

Let $F(n,k)$ be the expected number of queries used by A against $R(n,k)$. We have the following recurrence relation:

$$F(n,k) \geq 1 + \sum_{j=1}^{k-2} \frac{1}{2^j} \cdot \left(1 - \frac{1}{\lg n}\right) \cdot F(n^{2^{-j}}, k-j-2))$$

and $F(n,2) \geq 1$. From Lemma 4.6, $F(n,k) = \Omega(k)$.
□

THEOREM 4.8. *For every randomized algorithm finding connection classes of graphs with n vertices and k components, there is a graph in that class for which the expected number of queries used by the algorithm is $\Omega(\min\{k, \lg k + \lg\lg n\})$.*

Proof. If $k \geq \lg n$, then Theorem 4.1 yields $\lg k \geq \lg\lg n$ as a lower bound.

If $\lg n > k > \frac{1}{2}\lg\lg\frac{n}{2}$, consider the input distribution P formed by starting with a sample from $R(\frac{n}{2}, \frac{1}{2}\lg\lg\frac{n}{2})$ and then adding $k - \frac{1}{2}\lg\lg\frac{n}{2}$ arbitrary components so that the total number of vertices is n. Now we have a distribution over graphs with k components and n vertices. By Lemma 4.3, the expected number of queries used against distribution P is at least the expected number of queries used against distribution $R(\frac{n}{2}, \frac{1}{2}\lg\lg\frac{n}{2})$. By Lemma 4.7, this is $\Omega(\lg\lg n)$.

If $k \leq \frac{1}{2}\lg\lg\frac{n}{2}$, we start with $R(n/2, k-1)$ and add a single component to reach a total of n vertices. Lemma 4.3 yields expected cost at least the expected cost against $R(n/2, k-1)$. By Lemma 4.7, this is $\Omega(k)$.

By Theorem 4.1, $\lg k$ is always a lower bound. This completes the proof. □

Acknowledgments. The authors thank Edward Reingold and Steve Tate for helpful discussions and the referees for improving the presentation.

REFERENCES

[1] C. C. Chen and F. Hwang, *Detecting and locating electrical shorts using group testing,* IEEE Trans. on Circuits and Systems, 36 (1989), pp. 1113-1116.
[2] W.-T. Cheng, J. L. Lewandowski and E. Wu, *Optimal diagnostic methods for wiring interconnects*, IEEE Trans. on Computer-Aided Design, 11 (1992), pp. 1161-1166.
[3] M. Garey, D. Johnson, and H. So, *An application of graph coloring to printed circuit testing*, IEEE Trans. on Circuits and Systems, 23 (1976), pp. 591–599.
[4] N. Jarwala and C. W. Yau, *A new framework for analyzing test generation and diagnosis algorithms for wiring interconnect*, In Proc. International Testing Conference, 1989, pp. 63-70.
[5] W. H. Kautz, *Testing for faults in wiring networks*, IEEE Trans. on Computers, 23 (1973), pp. 358–363.
[6] L. Kavraki, J.-C. Latombe, R. Motwani and P. Raghavan, *Randomized query processing in robot motion planning*, in Proc. 27th Ann. ACM Sympos. on Theory of Computing, 1995, pp. 353-362.
[7] R. Motwani and P. Raghavan. *Randomized Algorithms*, Cambridge University Press, 1995.
[8] W. Shi and W. K. Fuchs, *Optimal interconnect diagnosis of wiring networks*, IEEE Trans. on VLSI Systems, 3 (1995), pp. 430–436.
[9] W. Shi and D. B. West, *Optimal structural diagnosis of wiring networks*, In Proc. 27th International Symposium on Fault Tolerant Computing, 1997, pp. 162-169.