Optimal Structural Diagnosis of Wiring Networks [1]

Weiping Shi

Department of Computer Science
University of North Texas
Denton, Texas 76203

Douglas B. West

Department of Mathematics
Univ. of Illinois at Urbana-Champaign
Urbana, Illinois 61801

Abstract

We consider the problem of full diagnosis all the shorts faults among a set of nets using the minimum number of parallel tests. We study adaptive and non-adaptive algorithms for structural diagnosis, using the information about the adjacency of the nets in the wiring network. We formulate the problem as a graph search problem and use graph theory to derive optimal and near-optimal adaptive and non-adaptive algorithms. Our results indicate that adaptive diagnosis often uses exponentially fewer tests than traditional non-adaptive diagnosis, and structural diagnosis may use exponentially fewer tests than functional diagnosis.

Keywords. Algorithm, testing, diagnosis, digital system, graph theory.

[1] W. Shi (wshi@cs.unt.edu) was supported by NSF grant MIP-9309120. D. B. West (west@math.uiuc.edu) was supported by NSA/MSP grant MDA904-93-H-3040. This paper has been accepted by FTCS'97.

1 Introduction

1.1 Problem Formulation

Interconnect diagnosis of wiring networks is an important problem with applications to design and testing of very large scale integration (VLSI), multi-chip module (MCM) and printed circuit board (PCB) systems [1, 4, 5, 6, 8, 9, 10, 11, 13, 19, 22, 23, 26]. A wiring network consists of a set of nets. Each net contains a set of interconnected drivers and receivers. The logic value of a good net is controlled by its drivers and observed by its receivers. When some nets are involved in a short fault, their receivers all receive the logical OR of the values of their drivers. (This is the *wired-OR model*. In the *wired-AND model*, they receive the logical AND of the driver values.) From the circuit layout information, we can find all potential places where a short fault may occur and represent the information as a graph $G = (V, \mathcal{E})$, which we call the adjacency graph. Each vertex $v \in V$ represents a net and each edge $v_i v_j \in \mathcal{E}$ represents a potential short fault between net v_i and net v_j. Note that although G records only possible faults between pairs of nets, we may have multiple-net short faults throughs sequences of faults involving two nets each.

The formulation is quite general, and any net extraction algorithm can be modified easily to construct G from the circuit layout [8, 18]. Figure 1 shows an example of four nets on a VLSI chip, with the adjacency graph G representing the potential short faults between the nets. In Figure 1, we assume the horizontal wires are routed on the first metal layer and the vertical wires are routed on the second metal layer, and short faults may occur between two wires at a cross point. For another example, consider a printed circuit board, where two nets share an edge in G if they are physically close enough to be possibly shorted [8, 22].

Figure 1: Routing of 4 nets and graph G representing all potential short faults between nets.

To diagnose a wiring network, a test engineer sends a parallel test vector of logical 0's and 1's from the drivers and observes the outputs from the receivers. The task of interconnect diagnosis is to apply test vectors to the nets to find the faults. In this paper, we study *structural diagnosis*, that is the detection and location of all single and multiple short faults between nets using an adjacency graph G. In contrast, *behavioral diagnosis* does not use any information regarding

the particular routing of the nets. Since no structural information is used, behavioral diagnosis assumes that all pairs of nets can be in short faults. Behavioral diagnosis is the special case of structural diagnosis in which $G = K_n$, where K_n is the complete graph on n vertices.

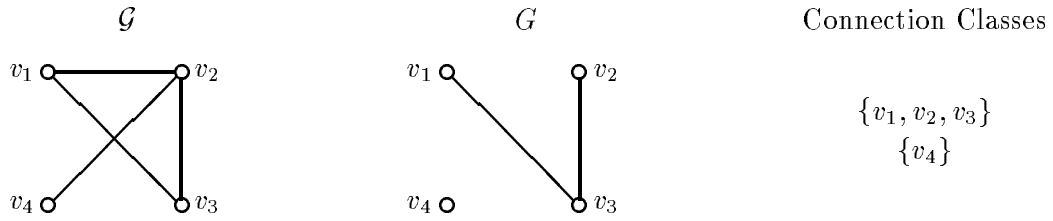

Figure 2: Adjacency graph G, fault graph F, and connection classes of F.

For any graph G, vertices x and y are *connected* in G if G contains a path with x and y as endpoints; such a path is an *x, y-path*. The components of a graph G are its maximal connected subgraphs, and the vertex sets of the components are the equivalence classes of the connection relation. We call these the *connection classes* of G. The problem of structural diagnosis of short faults can be described in graph theoretic terms as follows. Given an adjacency graph G, our goal is to find the connection classes of an unknown spanning subgraph F of G; *spanning* means that F contains all vertices of G. The subgraph F is the *fault graph*. Although F has the same vertex set as G, the edge set of F is not given. Each edge xy in F represents the occurrence of an actual direct short fault between nets x and y. Figure 2 shows an adjacency graph G, a fault subgraph F, and the connection classes of F. In general, it is not possible to determine the fault graph F by interconnect diagnosis; in the example of Figure 2, we cannot distinguish whether F has two edges or three edges among $\{v_1, v_2, v_3\}$.

The basic test operation is the *parallel test*: 0's and 1's are sent from the drivers and outputs are observed at the receivers. Formally, we obtain information about the fault graph F only through queries to an oracle. For any query set $S \subseteq V(G)$, the oracle tells us $Q(S)$, the set of vertices connected to vertices of S in the fault graph F; $Q(S)$ is the union of the connection classes that intersect S. In the example of Figure 2, $Q(\{v_1\}) = \{v_1, v_2, v_3\}$ and $Q(\{v_2, v_4\}) = \{v_1, v_2, v_3, v_4\}$. Our objective is to find the connection classes of F using the minimum number of queries. Diagnosing a wiring network is the same as finding the connection classes of the fault graph F, and applying a parallel test vector corresponds to querying the oracle.

Diagnosis algorithms may be adaptive or non-adaptive. In an *adaptive* algorithm, each query is computed using the responses to previous queries. In a *non-adaptive* algorithm, all query inputs are fixed before asking any questions. Non-adaptive diagnosis is used in built-in self test, where tests are hardwired. Adaptive algorithms can be used in so-called *boundary scan* [4, 9, 10, 11] where inputting tests and outputting results takes much longer than generating tests.

1.2 Previous Results

There are three levels of diagnostic resolution. The highest level is *full diagnosis*: finding all the connection classes [6, 23]. The second level is *faulty net identification* [5, 23]: identifying all the nets involved in short faults. The lowest level is *fault detection*, detecting whether there is any fault at all [13]. In this paper, we study full diagnosis.

Table 1 summarizes the previous best results for behavioral diagnosis. Table 2 summarizes the previous best results for structural diagnosis. In the table, lg denotes the base 2 logarithm, n is the number of nets, G is the adjacency graph, and $\chi(G)$ is the chromatic number of G.

Table 1: Best algorithms for behavioral diagnosis.

diagnostic resolution	**adaptive ?**	**best algorithm**	**# of tests**	**# of tests optimal?**	**time for test generation**
fault detection	no	counting seq [13]	$\lceil \lg n \rceil$	yes	$O(n \lg n)$
faulty net identification	no	max anti-chain [5]	$(1 + o(1)) \lg n$	almost	$O(n \lg n)$
full diagnosis	no	walking 1 skip 1 [23]	$n - 1$	yes	$O(n^2)$
	yes	divide conquer [23]	$\lceil \lg n \rceil$	yes	$O(n \lg n)$

Table 2: Best algorithms for structural diagnosis.

diagnostic resolution	**adaptive ?**	**best algorithm**	**# of tests**	**# of tests optimal?**	**time for test generation**
fault detection	no	graph coloring [5, 8]	$\lceil \lg \chi(G) \rceil$	yes	NP-hard
faulty net identification	no	max anti-chain [5]	$(1 + o(1)) \cdot \lg \chi(G)$	almost	NP-hard
full diagnosis	no	decomposition [6]	?	?	?
	yes	?	?	?	?

For behavioral diagnosis ($G = K_n$), Kautz [13] showed that $\lceil \lg n \rceil$ tests are necessary and sufficient to detect whether some short exists. He used the so-called *counting sequence* method, which is optimal for fault detection but does not provide faulty net identification. To identify all the nets involved in short faults, Cheng, Lewandowski, and Wu [5] proposed the *maximum independent set* method that uses approximately $\lg n + \frac{1}{2} \lg \lg n + O(1)$ tests. In this method,

a "maximum independent set" is the set of bit strings each containing an equal number of 0's and 1's (no string is coordinate-wise less than another; such families are called *antichains* in the theory of partially ordered sets, and the strings having an equal number of 0's and 1's correspond to a maximum antichain of subsets). They asked whether the method is optimal if information obtained from all nets may be used [5]. Shi and Fuchs [23] answered their question by proving that the maximum independent set method is not optimal but is within one test of optimality. For non-adaptive full diagnosis, Hassan, Rajski, and Agarwal [10] proposed the *walking ones* method. In the walking ones method, each query contains a single vertex. After n queries, we can look at each output $Q(\{v_i\})$ to tell which vertices are shorted to v_i. Shi and Fuchs [23] showed that the walking ones algorithm is not optimal, but that when we skip any one test it is optimal. For adaptive full diagnosis, Shi and Fuchs [23] presented a divide-and-conquer algorithm that uses $\lceil \lg n \rceil$ tests in the worst case, which is optimal.

For structural diagnosis, where the adjacency graph G is an arbitrary graph, several researchers [5, 8, 19, 22] observed that $\lceil \lg \chi(G) \rceil$ tests are needed (and suffice) for fault detection. where $\chi(G)$ is the chromatic number of G. (Because an adversary can can respond $Q(S) = S$ while potential edges remain, this result is equivalent to showing that the minimum number of bipartite subgraphs needed to cover G is $\lceil \lg \chi(G) \rceil$). Although this observation implies that finding the minimum set of tests is NP-hard, it relates the fault detection problem to a well-known graph theory problem where approximation algorithms are available. For full diagnosis, Lien and Breuer [19] defined neighborhood graphs, and McBean and Moore [22] defined pin-adjacency graphs. They proposed heuristics, which were improved by the graph decomposition algorithm of Feng, Huang, and Lombardi [6]. Several researchers [6, 11, 19] have refined non-adaptive algorithms to improve the bounds using structural information. For example, fault detection and walking ones can be combined to do full diagnosis in $\lceil \lg n \rceil + m$ tests, where m is the number of non-isolated vertices.

Other diagnosis problems have been studied. For diagnosis of networks with opens, shorts and stucks, see Goel and McMahon [9], Lien and Breuer [19], and Shi and Fuchs [23]. Boundary scan testing is studied in Chan [4] and in Angelotti *et al* [1]. Finding connection classes using an oracle for presence of single edges has been studied by Kavraki, Latombe, Motwani, and Raghavan [14].

1.3 New Results

In this paper, we present a variety of algorithms for interconnect diagnosis using graph theoretic techniques. In Section 2, we propose the first adaptive algorithm for structural diagnosis; Theorem 2.4 leads to a general method of approximating the minimum number of (adaptive) parallel tests for full diagnosis. In Section 3, we consider the non-adaptive problem, proving that deciding whether a set of parallel tests can perform full diagnosis is NP-hard, even for behavioral diagnosis.

Theorem 3.8 gives a general method of approximating the minimum number of (non-adaptive) parallel tests for full diagnosis. In Section 2 and Section 3, we present optimal or near-optimal adaptive and non-adaptive algorithms for full diagnosis when the adjacency graphs are complete graphs, complete bipartite graphs, paths, trees, planar graphs, circle graphs, or random graphs. Our results are summarized in Table 3. The results for the complete graph K_n were first given by Shi and Fuchs [23], but we give a different proof. The computation time for test generation is low-order polynomial for all algorithms in Table 3.

Table 3: Number of tests for full diagnosis of adjacency graphs in special classes.

algorithm type		K_n	$K_{m,n}$ $m \leq n$	path	tree	planar graph	circle graph	random graph
adaptive	lower bound	$\lceil \lg n \rceil$	$\lceil \lg(m+1) \rceil$	$\lg \lg n$	$\lg \lg n$	$\lg \lg n$	$\lg \Delta(G)$	$\lg n - \frac{1}{2} \lg \lg n$
	upper bound	$\lceil \lg n \rceil$	$\lceil \lg(m+1) \rceil$	$\lg \lg n$	$\lg \lg n + 3$	$\frac{1}{2} \lg n$	$\lg n$	$\lg n$
non-adaptive	lower bound	$n-1$	m	$\lceil \lg n \rceil$	$\lceil \lg n \rceil$	$\lceil \lg n \rceil$	$\lg n + \Delta(G)$	$n/\sqrt{\lg n}$
	upper bound	$n-1$	m	$\lceil \lg n \rceil$	$\lceil \lg n \rceil$	$O(\sqrt{n})$	$\lg n \cdot \Delta(G)$	$n-1$

In Table 3, $\Delta(G)$ is the maximum degree among all vertices in G. The "lower bound" for a class is the largest worst-case number of queries known to occur for graphs (of order n) in the class. In particular, among trees the path achieves the upper bound asymptotically and may be the worst case. Behavioral diagnosis corresponds to the case where the adjacency graph is the complete graph K_n. The case where G is a path was first studied by Yau and Jarwala [26]. Their algorithm uses $O(E + \lg n)$ tests, where E is a predetermined upper limit on the number of nets involved in short faults. Planar graphs arise in printed circuit board applications [8]. A graph is a circle graph if the vertices of the graph can be represented by chords of a circle so that two vertices are adjacent if and only if the corresponding chords intersect. In many switchbox routing or channel routing applications [18], the adjacency graph G is a circle graph.

Our results show that adaptive algorithms can reduce the number of queries exponentially compared with non-adaptive algorithms. This may have great impact for applications where on-line test generation can be computed quickly but application of tests is slow, such as in boundary scan [4, 9, 10, 11]. We also show that structural information can further reduce the number of queries drastically. In Table 3, there are exponential gaps between the number of tests required for non-adaptive algorithms and adaptive algorithms, and between the number of tests required for behavioral diagnosis and structural diagnosis of sparse graphs.

2 Adaptive Algorithms

In this section we study adaptive algorithms for finding the connection classes of an unknown subgraph F of an arbitrary graph G. An adaptive algorithm A implicitly defines a decision tree. At each node of the decision tree, the algorithm selects a set of vertices S and makes a query. Upon receiving the set $Q(S)$ from the oracle, the algorithm selects a subtree. Each leaf of the decision tree corresponds to a partition of the vertex set $V(G)$ into connection classes. The decision tree cannot be represented explicitly because many nodes have exponential degree in terms of $|V(G)|$, and because the number of leaves is at least the number of ways to partition $V(G)$.

Definition 2.1 Let $\mathbf{A}_G$ be the set of all adaptive algorithms that find the connection classes of an unknown subgraph of G. Let $q(A, F)$ be the number of queries used by algorithm A when the unknown subgraph is F. We define $q(G)$, the *adaptive query number* of a graph G, to be the minimum worst-case number of queries:

$$q(G) = \min_{A \in \mathbf{A}_G} \max_{F \subseteq G} q(A, F).$$

In other words, the query number $q(G)$ is the minimum number of parallel tests necessary and sufficient for adaptive full diagnosis when the graph of potential faults is G.

We use an important concept in topological graph theory: graph G_1 is a *minor* of graph G_2 if G_1 can be obtained from G_2 by a sequence of edge deletions and edge contractions. For example, Figure 3 shows that the complete graph K_4 is a minor of the complete bipartite graph $K_{3,3}$.

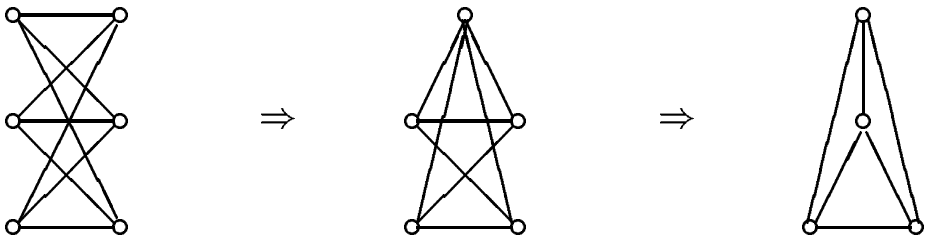

Figure 3: K_4 is a minor of $K_{3,3}$.

Lemma 2.1 If $G_1 = (V_1, \mathcal{E}_1)$ is a minor of $G_2 = (V_2, \mathcal{E}_2)$, then $q(G_1) \leq q(G_2)$.

Proof. If G_1 is obtained from G_2 by deleting [contracting] an edge e, then the problem of finding connection classes for an unknown subgraph F_1 of G_1 is the problem of finding the connection

classes of an unknown subgraph F_2 of G_2 with the additional information that F_2 does not contain [does contain] edge e. With more information about F_2 within G_2, we cannot need more queries.

□

Lemma 2.2 If the vertex sets of graphs G_1 and G_2 are disjoint, and $G_1 + G_2$ denotes the disjoint union of G_1 and G_2, then $q(G_1 + G_2) = \max\{q(G_1), q(G_2)\}$.

Proof. Let V_1, V_2 be the vertex sets for G_1, G_2, and let A_1, A_2 be optimal algorithms for finding connection classes on these graphs. We define an algorithm on $G_1 + G_2$. Whenever A_1 wants to query $S_1 \subseteq V_1$ and A_2 wants to query $S_2 \subseteq V_2$, we query $S_1 \cup S_2$. Since $Q(S_1) = Q(S_1 \cup S_2) \cap V_1$ and $Q(S_2) = Q(S_1 \cup S_2) \cap V_2$, both A_1 and A_2 can proceed. □

Definition 2.2 For a graph $G = (V, \mathcal{E})$ and set $S \subseteq V$, the *S-connection graph* G_S of G is the graph with vertex set S and edge set consisting of all pairs $v_i v_j$, such that $v_i, v_j \in S$ and G has a v_i, v_j-path that intersects S only at its endpoints.

The concept of S-connection graph allows us to concentrate on a set of vertices S and simplify the rest of the graph to keep only the connection information. In Figure 4, G_2 is the S-connection graph of G_1, for $S = \{v_1, v_2, v_3, v_4, v_5\}$.

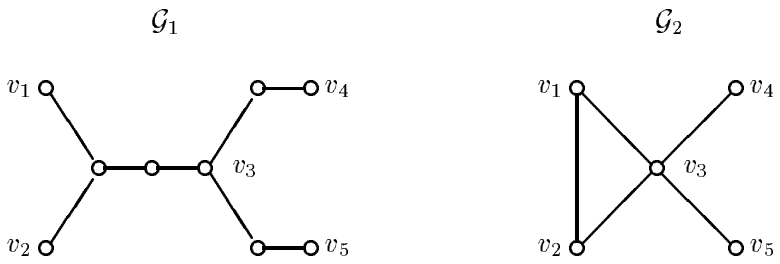

Figure 4: G_2 is an S-connection graph of G_1 for $S = \{v_1, v_2, v_3, v_4, v_5\}$.

Lemma 2.3 Given a graph G with spanning subgraph F and vertex subset S, there is a spanning subgraph F' of G_S such that the connection classes of F' are the intersections with S of the connection classes of F.

Proof. Let F' be the spanning subgraph of G_S consisting of the edges $v_i v_j$ such that F has a v_i, v_j-path intersecting S only at its endpoints. The graph F' incorporates the information of which pairs of vertices in F are in the same connection classes. □

For any graph G and set $S \subseteq V(G)$, the *induced subgraph* of G induced by vertex set S is the graph $G[S]$ whose edge set is $\{v_iv_j\colon\ v_iv_j \in E(G) \text{ and } v_i, v_j \in S\}$. The graph obtained by deleting the vertices in S is $G - S$; thus $G[V(G) - S] = G - S$.

Definition 2.3 Given a graph H and a set $S \subseteq V(H)$, the *restricted connection class problem* is the problem of finding the connection classes of an unknown spanning subgraph F of H, where F is restricted to lie among those spanning subgraphs whose components all intersect S. In other words, the possible unknown subgraphs are those for which $Q(S) = V(H)$. The number of queries required to solve this problem is

$$q'(H, S) = \min_A \max_F q^*(A, F),$$

where F belongs to the set described above, A has the knowledge that F is in this set, and $q^*(A, F)$ is the number of queries that A uses when the unknown subgraph is F.

Arguing as in the proof of Lemma 2.1, it follows that if H_1 is a minor of H_2 and both contain the vertex set S, then $q'(H_1, S) \leq q'(H_2, S)$.

Theorem 2.4 For a graph G with vertex set V,

$$q(G) \leq 1 + \min_{S \subseteq V} \max\{q(G_S), q(G - S)\}. \tag{1}$$

Proof. Let A be an algorithm that solves the connection class problem on G. Let S be the set of vertices chosen by A to be the first query. The response $Q(S)$ from the oracle produces disjoint subgraphs $G[Q(S)]$ and $G - Q(S)$, such that the unknown subgraph has no edge between $Q(S)$ and its complement. Therefore,

$$\begin{aligned} q(G) &\leq 1 + \min_{S \subseteq V} \max_{Q(S)} \{\max\{q'(G[Q(S)], S), q(G - Q(S))\}\} && (2)\\ &= 1 + \min_{S \subseteq V} \max\{\max_{Q(S)} q'(G[Q(S)], S), \max_{Q(S)} q(G - Q(S))\} && (3)\\ &\leq 1 + \min_{S \subseteq V} \max\{q'(G, S), q(G - S)\}. && (4) \end{aligned}$$

In the last step, we have used that each $G[Q(S)]$ is a minor of G, and that each $G - Q(S)$ is a minor of $G - S$.

To finish the proof, it remains to show that $q'(G, S) \leq q(G_S)$. Let A' be an optimal algorithm that solves the connection class problem for G_S. We use A' to solve the restricted connection class problem for (G, S). When A' wants to make a query T on G_S, we make the query T on G and use $Q(T) \cap S$ as a simulated response on G_S. By Lemma 2.3, this is the correct response for

an actual spanning subgraph F' of G_S whose connection classes are the intersections with S of the connection classes of the unknown subgraph F of G.

The algorithm A' uses the response $Q(T) \cap S$ to choose the next query to make on G_S. When A' completes its work, it declares the connection classes for F'. We claim that this partition of S permits us to determine the connection classes of F, without further queries. If this claim is true, then we have used at most $q(G_S)$ queries to solve the restricted connection class problem.

Let C be a connection class of F. Since $Q(S) = V(G)$, C contains a nonempty subset of S. Also, $C \cap S$ is a connection class in F'. Thus we know the correct distribution of S among connection classes of F. Given $v \in C - S$, let $U = C \cap S$, and let U' be another connection class of F'. Every response $Q(T) \cap S$ is a union of connection classes in F'. Since A' determines that U and U' are distinct classes, some response contains one of them but not the other. In either case, we learn that v is not connected to any vertex of U' in F. Since we learn this for each connection class of F' other than U, we know that v is in the connection class containing U. □

Theorem 2.5 If G is a graph such that G_S is a minor of G for all $S \subseteq V(G)$, then

$$q(G) = 1 + \min_{S \subseteq V} \max\{q(G_S), q(G - S)\}. \tag{5}$$

Proof. Let V be the vertex set of G. Every optimal algorithm begins by making a query on some set S, after which it must (in parallel) solve the restricted problem $(G[Q(S)], S)$ and the usual connection class problem on $G - Q(S)$. It chooses S to minimize the worst-case subsequent number of queries. This yields the first equality below.

$$\begin{aligned}
q(G) &= 1 + \min_{S \subseteq V} \max_{Q(S)} \{\max\{q'(G[Q(S)], S), q(G - Q(S))\}\} && (6)\\
&= 1 + \min_{S \subseteq V} \max\{\max_{Q(S)} q'(G[Q(S)], S), \max_{Q(S)} q(G - Q(S))\} && (7)\\
&\geq 1 + \min_{S \subseteq V} \max\{q'(G, S), q(G - S)\} && (8)\\
&\geq 1 + \min_{S \subseteq V} \max\{q'(G_S, S), q(G - S)\} && (9)\\
&= 1 + \min_{S \subseteq V} \max\{q(G_S), q(G - S)\}. && (10)
\end{aligned}$$

Within the maximization, we cannot increase the value by choosing a particular $Q(S)$. In the first term we choose $Q(S) = V(G)$; in the second we choose $Q(S) = S$. Then, since G_S is a minor of G, we can reduce to the restricted problem on G_S. Finally, we obtain $q'(G_S, S) = q(G_S)$ since $Q(S) = V(G_S)$ provides no information, and the restricted problem (G_S, S) is actually the unrestricted problem on G_S. The other direction of the inequality is Theorem 2.4. □

Theorem 2.4 can be used to design approximation algorithm for more general graphs, such as the one at the end of this section. Theorem 2.5 can be used to obtain exact expressions for the adaptive query number on some classes of graphs.

Corollary 2.6 For the complete graph K_n on n vertices, $q(K_n) = \lceil \lg n \rceil$.

Proof. This was first proved by Shi and Fuchs [23]. Here we obtain it from Theorem 2.5. Each S-connection graph of K_n is the clique $K_{|S|}$, which is a minor of K_n. Also $K_n - S = K_{n-|S|}$. Therefore $q(K_n) = 1 + q(K_{\lceil n/2 \rceil}) = \lceil \lg n \rceil$. □

Corollary 2.7 Let G be a graph, with $S \subseteq V(G)$. If $G_1, G_2, \ldots, G_k$ are the components of $G - S$, then $q(G) \leq 1 + \max\{\lceil \lg |S| \rceil, q(G_1), \ldots, q(G_k)\}$.

Proof. Since every S-connection graph is a minor of $K_{|S|}$, $q(G_S) \leq q(K_{|S|}) = \lceil \lg |S| \rceil$. From Lemma 2.2, $q(G - S) = \max\{q(G_1), q(G_2), \ldots, q(G_k)\}$. □

Corollary 2.8 For the path P_n on n vertices, $q(P_n) = \lg \lg n + O(1)$.

Proof. Each S-connection graph is $P_{|S|}$, which is a minor of P_n. The graph $G - S$ has $n - |S|$ vertices. From the pigeonhole principle, at least one component of $G - S$ is a path of at least $(n - |S|)/(|S| + 1)$ vertices. On the other hand, if the vertices of S are evenly spaced among the n vertices, then every component in $G - S$ contains at most $(n - |S|)/(|S| + 1)$ vertices. Therefore

$$q(P_n) \quad = \quad 1 + \min_{S \subset V} \max\{q(P_{|S|}), q(P_{\lceil (n-|S|)/(|S|+1) \rceil})\}. \tag{11}$$

Solving the equation $|S| = (n - |S|)/(|S| + 1)$ gives $|S| = \sqrt{n+1} - 1$. Therefore,

$$q(P_n) \quad = \quad 1 + q(P_{\lceil \sqrt{n+1}-1 \rceil}) \tag{12}$$

which yields $q(P_n) = \lceil \lg \lg(n + 1) \rceil$. □

It is important to clarify that the argument of Theorem 2.4 gives a procedure for generating the first query for a query algorithm whose number of queries satisfies the resulting bound, but the argument does not directly yield a recursive algorithm for finding the connection classes. Although the first query is the set S that minimizes $\max\{q(G_S), q(G - S)\}$, the connection class algorithm for a given instance does not then find connection classes in $q(G_S)$ and $q(G - S)$. It receives the response $Q(S)$ from the oracle and procedes to find connection classes in $G[Q(S)]$

and each component of $G - Q(S)$, in parallel, knosing that the connection classes in $G[Q(S)]$ all intersect S. For $G[Q(S)]$, it takes at most $q(G_S)$ additional queries to solve the problem, but G_S does not provide the next query set.

For example, Figure 5 considers an instance for $G = P_{15}$, using Corollary 2.8 to generate the queries. The edges in the adjacency graph G are shown, and the edges in the fault graph F are marked with "$\times$". The dark vertices are vertices in the query set S. After the first query, we have five subgraphs containing potential components, including two in $G[Q(S)]$ and three in $G - Q(S)$. Each subgraph is a path. We use Corollary 2.8 to generate the next test for paths in $G - Q(S)$. Because the 7-vertex component in $G[Q(S)]$ was generated by two vertices of S, we know that all but one of its edges are faults, and one additional query at any of its vertices finishes the problem. After the second query, we have all the connection classes of F. A general adaptive algorithm for finding connection classes appears in Section 4.

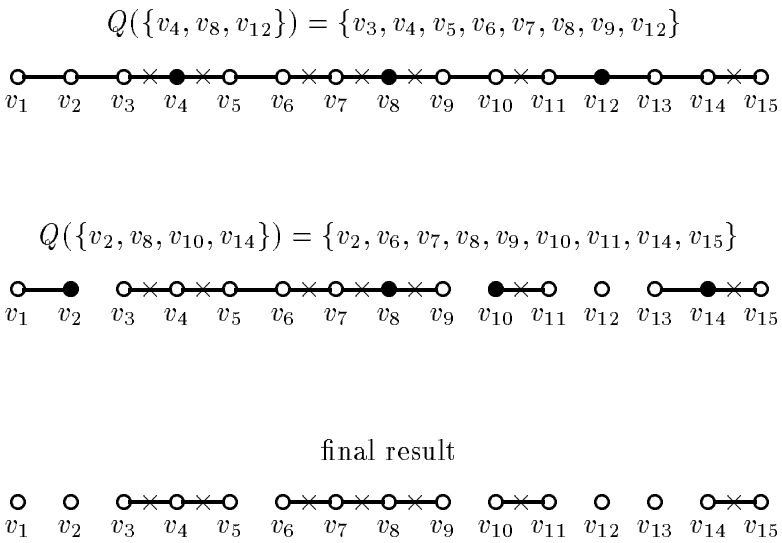

Figure 5: Find connection classes of $F \subset P_{15}$ in 2 queries.

Theorem 2.9 If G is a tree of n vertices, then $q(G) \leq \lg\lg n + 3$, and the queries can be constructed in polynomial time.

Proof. By removing a single vertex, an n-vertex tree can be partitioned into components having at most $n/2$ vertices each. (Given $v \in V(G)$, if some component in $G - \{v\}$ has more than $n/2$ vertices, move to the neighbor of v in that component, and repeat until all components have size at most $n/2$.)

We iteratively place such splitting vertices into S until each remaining component has at most $2\sqrt{n}$ vertices. This process is modeled by a decomposition tree T. The parents of leaves in T correspond to connected subgraphs of G with at least $2\sqrt{n}$ vertices, so there are at most $\sqrt{n}/2$ of them. When the leaves of T are deleted, we have a tree with at most $\sqrt{n}/2$ leaves and thus fewer than $\sqrt{n}$ vertices, each corresponding to a vertex of S.

If G_S at this point is not a tree, as in Figure 4, we add additional vertices of G to S. Let G' denote the subgraph of G that is the union of all paths in G joining vertices of G. We add to S all vertices that have degree at least 3 in G' (in Figure 4, one vertex is added). Since G' has fewer than $\sqrt{n}$ leaves, we add fewer than $\sqrt{n}$ vertices to S. The final set S has fewer than $2\sqrt{n}$ vertices. The graph G_S is the graph obtained from G' by contracting an edge incident to a vertex of degree 2 (unless both endpoints are in S) until no further such operations are available.

Let $f(n) = \max q(G)$, where the maximum is taken over all n-vertex trees. By Theorem 2.4, $f(n) \leq 1 + f(2\sqrt{n})$. This recurrence yields $f(n) \leq \lg\lg n + f(8) = \lg\lg n + 3$. □

Theorem 2.9 is essentially best possible, because the query number of the path is within three of this bound.

Theorem 2.10 If G is the complete bipartite graph $K_{m,n}$, then $q(G) = \lceil \lg(\min\{m,n\} + 1) \rceil$.

Proof. Assume without loss of generality that $m \leq n$. If we pick any $\lceil m/2 \rceil$ vertices in the partite set of size m to use as S in Theorem 2.4, then

$$q(K_{m,n}) \leq 1 + \max\{q(K_{\lceil m/2 \rceil}), q(K_{\lfloor m/2 \rfloor,n})\} \leq 1 + q(K_{\lceil m/2 \rceil,n}).$$

With $q(K_{1,n}) = 1$, the recurrence yields $q(K_{m,n}) \leq \lceil \lg(m+1) \rceil$.

On the other hand contracting $m-1$ edges of a matching yields K_{m+1} as a minor K_{m+1}, as illustrated in Figure 2. From Lemma 2.1, $q(K_{m,n}) \geq q(K_{m+1}) = \lceil \lg(m+1) \rceil$. □

Next suppose that G is a planar graph. Planar graphs arise naturally when the routing is planar and short faults occur only between wires that are close together [8, 22].

Theorem 2.11 (Planar Separator Theorem, Lipton-Tarjan [21]) Let G be an n-vertex planar graph. In $O(n)$ time we can partition $V(G)$ into three sets A, B, C such that 1) no edge has endpoints in A and B, 2) $|A|, |B| \leq 2n/3$. and 3) $|C| \leq \sqrt{8n}$. C contains no more than $\sqrt{8n}$ vertices.

Corollary 2.12 Let G be an n-vertex planar graph. In $O(n)$ time we can find a set $S \subset V(G)$ such that $|S| \leq (12 + 6\sqrt{2})\sqrt{n}$ and the components of $G - S$ have order less than $n/4$.

Proof. From Theorem 2.11, $V(G)$ can be partitioned into A, B and C such that there is no edge between A and B, $|A|, |B| \leq \sqrt{8n}$, and $|C| \leq 2n/3$. We call such a set C a *separator*. The sizes of A and B are bounded by αn and $(1 - \alpha n)$ for some $1/3 \leq \alpha \leq 2/3$. Recursively find separators C_A for $G[A]$ and C_B for $G[B]$, we have $|C_A \cup C_B| \leq \sqrt{8\alpha n} + \sqrt{8(1-\alpha)n} < \sqrt{8n}\sqrt{2}$. We apply Theorem 2.11 recursively for 4 levels, reducing all components among the remaining vertices to order at most $(2/3)^4 n = 16n/81 < n/4$. Let S be the union of all the separators found in this tree of separations. We have

$$|S| \leq \sqrt{8n}(1 + 2^{1/2} + 2^{2/2} + 2^{3/2}) = \sqrt{n}(12 + 6\sqrt{2}).$$

Since each C can be found in time linear in the current number of vertices, the total time to find S is $O(n)$. □

Theorem 2.13 If G is a planar graph of n vertices, then $q(G) \leq \frac{1}{2}\lg n + O(1)$, and each query can be constructed in $O(n)$ time.

Proof. Let $f(n)$ be the maximum query number of an n-vertex planar graph. We prove by induction on n that $f(n) \leq \frac{1}{2}\lg n + c + 1$, where $c = \lg(12 + 6\sqrt{2})$.

When $n < 4$, the bound holds trivially. For the induction step, consider $n \geq 4$. By Corollaries 2.7 and 2.12,

$$\begin{aligned} f(n) &\leq 1 + \max\{\lg(\sqrt{n}(12 + 6\sqrt{2})), f(\lfloor n/4 \rfloor)\} && (13) \\ &\leq 1 + \max\{\frac{1}{2}\lg n + c, \frac{1}{2}\lg(n/4) + c + 1\} && (14) \\ &= \frac{1}{2}\lg n + c + 1. && (15) \end{aligned}$$

□

We do not know whether Theorem 2.13 is best possible. Since paths are planar, we have n-vertex planar graphs requiring $\lceil \lg\lg(n+1) \rceil$ queries, but these are our worst examples. Note that the S-connection graph of a planar graph need not be planar.

Random graphs generated by letting each edge occur with probability .5 tend to be fairly dense. We show next that when solving the connection class problem by adaptive algorithms on dense graphs, structural diagnosis is not much better than functional diagnosis.

Theorem 2.14 For almost every graph G, $q(G) \geq \lg n - \frac{1}{2}\lg\lg n + O(1)$. This bound also holds for every graph with about $n^2/4$ edges.

Proof. Bollobás, Catlin and Erdős [3] proved that almost every n-vertex graph has K_m as a minor, where $m = (1+o(1))n/\sqrt{\lg n}$. Lemma 2.1 then yields $q(\mathcal{G}) \geq \lg m = \lg n - \frac{1}{2}\lg\lg n + O(1)$ almost always. In fact, Kostochka [16] and Thomason [24] each proved that *every* graph of n vertices and about $n^2/4$ edges has K_m as a minor, where $m = (1+o(1))n/\sqrt{\lg n}$ (see Bollobás [2], page 279). □

In general, computing $q(G)$ appears to be NP-hard, but we have not proved this. Observe that the minimum number of queries to determine whether $F = \overline{K}_n$ equals $\lg \chi(G)$. Thus $q(G) \geq \lg \chi(G)$, but it is still possible that $q(G)$ is easier to compute. The problem belongs to a class of problems known as $\Sigma^P_{\log n}$, which is a superset of NP (see Garey and Johnson [7] for the definition).

3 Non-Adaptive Algorithms

In this section we study non-adaptive algorithms for solving the connection class problem. A non-adaptive algorithm T is a set of queries $S_1, S_2, \ldots, S_t$, and a subroutine that analyzes the set of responses $Q(S_1), Q(S_2), \ldots, Q(S_t)$ to derive the connection classes. The query sets $S_1, S_2, \ldots, S_t$ are fixed. We say that a non-adaptive algorithm T solves the connection class problem for G if, for every subgraph F of G, T finds the connection classes of F.

Definition 3.1 Let $\mathbf{T}_G$ be the set of all non-adaptive algorithms that solve the connection class problem for G. Let the worst-case number of queries used by an algorithm T be $t(T)$. The *non-adaptive test number* $t(G)$ of a graph G is the minimum number of non-adaptive queries that always suffices to solve the connection class problem of G:

$$t(G) = \min_{T \in \mathbf{T}_G} t(T).$$

A sequence of queries $S_1, S_2, \ldots, S_t$ defines a *sequential test vector (STV)* $X(v_i)$ for each vertex v_i. For each vertex v_i, $X(v_i) = x_{i1}x_{i2}\cdots x_{it}$ is the t-bit binary vector defined by $x_{ij} = 1$ if $v_i \in S_j$, and $x_{ij} = 0$ otherwise. Conversely, sequential test vectors $X(v_1), X(v_2), \ldots, X(v_n)$, where $X(v_i) = x_{i1}x_{i2}\cdots x_{it}$, define a set of queries $S_1, S_2, \ldots, S_t$ by $S_j = \{v_i : x_{ij} = 1\}$. We will also use sequential test vectors to describe the queries in this section, because the test vector language underscores the parallel aspects of non-adaptive queries.

For example, Table 4 shows for $n = 5$ the set of queries called the *walking ones sequence* and the corresponding sequential test vectors. Table 5 shows another set of queries.

It is immediate that the walking ones sequence suffices for full diagnosis. It is also known that a set of sequential test vectors can perform full diagnosis if it contains the walking ones sequence

Table 4: The walking ones sequence for five vertices.

	S_1	S_2	S_3	S_4	S_5
v_1	1	0	0	0	0
v_2	0	1	0	0	0
v_3	0	0	1	0	0
v_4	0	0	0	1	0
v_5	0	0	0	0	1

queries	STVs
$S_1 = \{v_1\}$	$X(v_1) = 10000$
$S_2 = \{v_2\}$	$X(v_2) = 01000$
$S_3 = \{v_3\}$	$X(v_3) = 00100$
$S_4 = \{v_4\}$	$X(v_4) = 00010$
$S_5 = \{v_5\}$	$X(v_5) = 00001$

Table 5: A set of four queries and sequential test vectors.

	S_1	S_2	S_3	S_4
v_1	0	0	1	1
v_2	1	0	0	1
v_3	0	1	1	0
v_4	0	0	0	0
v_5	1	0	1	0

queries	STVs
$S_1 = \{v_2, v_5\}$	$X(v_1) = 0011$
$S_2 = \{v_3\}$	$X(v_2) = 1001$
$S_3 = \{v_1, v_3, v_5\}$	$X(v_3) = 0110$
$S_4 = \{v_1, v_2\}$	$X(v_4) = 0000$
	$X(v_5) = 1010$

[19] or is diagonally independent [11]. It may be less obvious that the queries in Table 5 can also perform full diagnosis.

Lemma 3.1 (Shi-Fuchs [23]) A necessary and sufficient condition for sequential test vectors $X(v_1), \ldots, X(v_n)$ to solve the connection class problem on K_n is that for any disjoint nonempty vertex sets U, V, $\vee_{u \in U} X(u) \neq \vee_{v \in V} X(v)$, where the operation $\vee$ is the bit-wise Boolean OR.

Lemma 3.1 does not give an efficient method to check whether the set of sequential test vectors can perform full diagnosis, because the number of such V_1 and V_2 subsets is exponential in terms of n. Next we prove that it is unlikely that there exists any polynomial time verifiable characterization.

Theorem 3.2 It is NP-hard to tell whether a set of sequential test vectors can solve the connection class problem for G, even if $G = K_n$.

Proof. By Lemma 3.1, it is sufficient to show the following problem is NP-hard.

Problem II: Given a set of t-bit binary vectors $T = \{X_1, X_2, \ldots, X_n\}$, are there disjoint non-empty subsets of indices I and J such that $\vee_{i \in I} X_i = \vee_{j \in J} X_j$?

We start the reduction from the *Not-All-Equal 3SAT* problem, known to be NP-complete [7]:

> Instance: Set $Y = \{y_1, y_2, \ldots, y_n\}$ of variables, collection $C = \{c_1, c_2, \ldots, c_t\}$ of clauses over Y such that each clause $c \in C$ contains at most three variables.
>
> Question: Is there a truth assignment for Y such that each clause in C has at least one true literal and at least one false literal?

First observe that Not-All-Equal 3SAT remains NP-complete even if all clauses contain only positive literals: Given an instance of Not-All-Equal 3SAT problem with some negated literals, replace each negated literal $\overline{y}_i$ by a new variable y'_i and add a new clause $(y_i + y'_i)$. Now all clauses contain only positive literals, and the newly added clauses force $y'_i = \overline{y}_i$, for all i.

From an instance of Not-All-Equal 3SAT in which all clauses contain only positive literals, we construct a set of t-bit binary vectors $T = \{X_1, \ldots, X_n\}$. For $1 \leq i \leq n$, we define $X_i = x_{i1}x_{i2}\cdots x_{it}$ by $x_{ij} = 1$ if $y_i \in c_j$ and $x_{ij} = 0$ otherwise.

Observe that any solution to the instance of the Not-All-Equal 3SAT defines sets of indices $I = \{i\colon\ y_i \text{ is true}\}$ and $J = \{j\colon\ y_j \text{ is false}\}$. These sets are disjoint, non-empty, and satisfy $\vee_{i\in I} X_i = \vee_{i\in J} X_i = 111\cdots1$. Conversely, a partition of the index set into sets I and J such that $\vee_{i\in I} X_i = \vee_{i\in J} X_i = 111\cdots1$. yields a solution to this instance of Not-All-Equal 3SAT. Therefore, the following problem is NP-hard:

> Problem Π': Given t-bit binary vectors $T = \{X_1, X_2, \ldots, X_n\}$, is there a partition I, J of $\{1, 2, \ldots, n\}$ such that $\bigvee_{i\in I} X_i = \bigvee_{j\in J} X_j$?

To finish the proof, we show that a polynomial time algorithm A for problem Π would yield a polynomial time algorithm for problem Π'. Given an instance T of problem Π', we use A to find disjoint nonempty index sets I and J, such that $\bigvee_{i\in I} X_i = \bigvee_{j\in J} X_j$. There are three cases.

1) No such I and J is found. Then instance T does not have a solution for problem Π'.

2) They are found, and $I \cup J = \{1, 2, \ldots, n\}$. Then instance T has a solution for problem Π'.

3) They are found but $I \cup J \neq \{1, 2, \ldots, n\}$. First expand I and J to be maximal sets solving Π. Let $Z = \vee_{i\in I} X_i$. Then instance T of problem Π' has a solution if and only if the following instance of problem Π' has a solution: $T' = \{X_i \wedge \overline{Z}\colon\ i \notin I \cup J\}$. □

Theorem 3.4 computes $t(K_n)$. Shi and Fuchs [23] proved it using general consequences of Lindström [20] and Tverberg [25]. Here we give a direct proof using Tverberg's argument. We need a set theory result as a lemma.

Lemma 3.3 (Radon's Theorem) Let $p_1, p_2, \ldots, p_m$ be points in $\mathbf{R}^d$. If $d < m-1$, then the set of points has disjoint subsets P and Q such that the intersection of the convex hull of P and the convex hull of Q is non-empty.

Theorem 3.4 When G is the complete graph K_n on n vertices, $t(G) = n-1$.

Proof. The $n-1$ queries $\{v_1\}, \{v_2\}, \ldots, \{v_{n-1}\}$ suffice; we show that no smaller set suffices.

Consider a set of t-bit sequential test vectors $X_1, \ldots, X_n$, where $X_i = x_{i1}x_{i2}\ldots x_{it}$. We prove that if $t < n-1$, then there are non-empty disjoint subsets of indices I and J such that $\vee_{i\in I} X_i = \vee_{j\in J} X_i$.

We may assume that $t = n-2$; otherwise we add extra queries. Since a set containing two zero vectors cannot test whether the corresponding vertices are isolated or form an isolated edge, we may assume that $X_1, \ldots, X_{n-1}$ are nonzero vectors. Let $s_i = X_i \cdot 1_t$. Normalize the vectors by defining $Y_i = \frac{1}{s_i} X_i$, so that each Y_i has sum 1. Thus each Y_i lies in a hyperplance in $\mathbf{R}^t$; this hyperplane is isomorphic to $\mathbf{R}^{t-1} = \mathbf{R}^{n-3}$. Although the hyperplane is $n-3$-dimensional, we have still expressed the points using t coordinates corresponding to the queries.

By Radon's Theorem, there are minimal nonempty disjoint subsets of these points having a common point Y' in their convex hull. We can write $Y' = \sum_{i\in I} \alpha_i Y_i = \sum_{j\in J} \beta_j Y_j$, with all coefficients being positive. Each coordinate in Y' is nonzero if and only if it is nonzero in Y_i and Y_j for some $i \in I$ and $j \in J$. Thus $\vee_{i\in I} X_i = \vee_{j\in J} X_i$, with this vector being nonzero in the same coordinates as Y'. □

We extend the Shi-Fuchs theorem to arbitrary graphs.

Theorem 3.5 A necessary and sufficient condition for sequential test vectors $\{X(v)\colon v \in V(G)\}$. to find all connection classes of G is that $\vee_{u\in U} X(u) \neq \vee_{v\in V} X(v)$ whenever U, V are disjoint nonempty subsets of $V(G)$ such that $G[U]$, $G[V]$, and $G[U \cup V]$ are connected graphs.

Proof. If the condition fails using sets $U, V \subseteq V(G)$, then these test vectors do not distinguish between inputs in which the only nontrivial connection class is $U \cup V$ and inputs in which the only nontrivial connection classes are U and V. (This has been called the *confounding syndrome* [11].)

Conversely, suppose that the condition holds. First partition the set of vertices into disjoint subsets according the response of each query as follows. Before the ith step, suppose that we have partitioned V into $V_1, \ldots, V_m$. The response of the next query $Q(S_i)$ further partitions each V_j

into $V_j \cap Q(S_i)$ and $V_j \cap (V - Q(S_i))$. When we finish all the queries, report each maximal vertex set inducing a connected subgraph of G that lies in a single V_i as one connection class of the fault graph F. We show that this algorithm works correctly.

If C is a connection class of F, then $C \subseteq Q(S)$ or $C \cap Q(S) = \emptyset$ for every $S \subseteq V(G)$. Thus all of C remains in the same Vk, and all of C will be reported to be in the same connection class.

If connection classes $C_1, \ldots, C_k$ of F are reported as being a single connection class T, then $G[T]$ is connected, because each set reported as a connection class induces a connected subgraph of G. Also, at each iteration the algorithm leaves the entire set T in a single block V_k of the partition. Thus each $Q(S_i)$ contains all or none of T. Thus S_i contains a vertex of some C_j if and only if it contains a vertex of each C_j. In particular, $\vee_{v \in C_j} X(v)$ is the same for all j. Letting $U = C_1$ and $V = C_2 \cup \ldots \cup C_k$ yields a violation of the condition. □

Ideas like those of Section 2 allow us to compute non-adaptive test numbers of some graphs.

Lemma 3.6 If G_1 is a minor of G_2, then $t(G_1) \leq t(G_2)$.

Proof. Similar to Lemma 2.1. □

Lemma 3.7 If of graphs G_1 and G_2 are disjoint, then $t(G_1 + G_2) = \max\{t(G_1), t(G_2)\}$.

Proof. Similar to Lemma 2.2. □

Theorem 3.8 For a graph G with vertex set V,

$$t(G) \leq 1 + \min_{S \subseteq V}\{t(G_S) + t(G - S)\}. \tag{16}$$

Proof. For the set S achieving the minimum, we make the queries consisting of S, a minimum set of queries for G_S, and a minimum set of queries for $G - S$. Each resulting sequential test vector $X(v)$ is the concatenation of one truth bit for $v \in S$, the sequential test vector for v in the query set for G_S, and the sequential test vector for v in the query set for $G - S$.

Let U, V be disjoint nonempty subsets of $V(G)$ such that $G[U]$, $G[V]$, and $G[U \cup V]$ are connected. It suffices to show that $\vee_{u \in U} X(u) \neq \vee_{v \in V} X(v)$.

If U, V both lie outside S, then U, V and $U \cup V$ induce connected subgraphs in $G - S$. Thus the last $t(G - S)$ bits of the test vectors yield the desired non-equality.

If exactly one of U, V intersects S, then query S yields the desired non-equality, since $\vee_{u \in U} X(u)$ and $\vee_{v \in V} X(v)$ differ in the first bit.

Finally, suppose that both U and V intersect S. Let $U' = U \cap C$ and $V' = V \cap C$. By construction, $U', V', U' \cup V'$ all induce connected subgraphs of G_S. Thus the $t(G_S)$ bits of the test vectors corresponding to G_S yield the desired non-equality. □

Corollary 3.9 Let G be a graph, with $S \subseteq V(G)$. If $G_1, G_2, \ldots, G_k$ are the components of $G - S$, then $t(G) \leq |S| + \max_{1 \leq i \leq k}\{t(G_i)\}$.

Proof. Since G_S is a minor of $K_{|S|}$, $1 + t(G_S) \leq 1 + t(K_{|S|}) = |S|$. Lemma 3.7 and Theorem 3.8 complete the proof. □

Corollary 3.9 is similar to the algorithm of Feng, Huang and Lombardi [6].

Theorem 3.10 If G is the complete bipartite graph $K_{m,n}$, then $t(G) = \min\{m, n\}$.

Proof. Similar to Theorem 2.10. If S is the smaller partite set, then $G - S$ has no edges. Corollary 3.9 then yields $t(K_{m,n}) \leq \min\{m, n\}$. On the other hand, $K_{m,n}$ has $K_{\min\{m,n\}+1}$ as a minor. Therefore $t(K_{m,n}) \geq t(K_{1+\min\{m,n\}}) = \min\{m, n\}$. □

Theorem 3.11 For the n-vertex path P_n, $t(P_n) = \lceil \log n \rceil$.

Proof. For the upper bound, pick the center vertex as S in Theorem 3.8. Then $t(P_n) \leq 1 + t(P_{\lceil n/2 \rceil})$. Solving the recurrence relation with $t(P_1) = 0$ gives $t(P_n) \leq \lceil \lg n \rceil$.

For the lower bound. let U be the set of $\lceil n/2 \rceil$ vertices closest to one end of the path, and let V be the remaining vertices. Note that $U, V, U \cup V$ all induce connected subgraphs. For sequential test vectors solving the connection class problem, we must have $\vee_{u \in U} X(u) \neq \vee_{v \in V} X(v)$. Let j be a coordinate where they differ. The elements of one of $\{U, V\}$ have test vectors that all are 0 in coordinate j. We may assume that it is V (when n is odd, we may shift the central vertex from one set to the other if necessary to make the smaller set all 0 in coordinate j. That means we can solve the connection class problem on V_2 without using query S_j. Therefore $t(P_{\lceil n/2 \rceil}) \leq t(P_n) - 1$ and $t(P_1) = 0$. Solving the recurrence relation yields $t(P_n) \geq \lceil \lg n \rceil$. □

Let $\chi(G)$ be the chromatic number of G. Since $\lg \chi(G)$ queries are necessary to tell whether $F = \overline{K}_n$, at least this many queries are needed in the worst case to find all the connection classes.

Therefore $t(G) \geq \lg \chi(G)$. If the famous conjecture of Hadwiger [3] is true, then $t(G) \geq \chi(G) - 1$. Hadwiger's conjecture states that each graph G has $K_{\chi(G)}$ as a minor.

Theorem 3.12 If G is a tree of n vertices, then $t(G) \leq \lg n$, and the queries can be constructed in polynomial time.

Proof. As in Theorem 2.9, each n-vertex tree can be divided into subtrees of order at most $n/2$ by removing one vertex. Let $f(n) = \max t(G)$, where the maximum is taken over all n-vertex trees. From Theorem 3.8, $f(n) \leq 1 + f(n/2)$. Solving the recurrence relation with $f(1) = 0$ yields $f(n) \leq \lceil \lg n \rceil$. □

The upper bound in Theorem 3.12 holds with equality for paths, by Theorem 3.11.

Theorem 3.13 If G is an n-vertex planar graph, then $t(G) = O(\sqrt{n})$, and the set of queries can be computed in polynomial time.

Proof. Let $f(n) = \max t(G)$, where the maximum is taken over all n-vertex planar graphs. From Theorem 2.11 and Corollary 3.9,

$$f(n) \leq \sqrt{8n} + f(2n/3) \leq \sqrt{8n}\left(1 + \sqrt{\frac{2}{3}} + \frac{2}{3} + \left(\frac{2}{3}\right)^{3/2} + \cdots\right) = O(\sqrt{n}). \tag{17}$$

The time to find each separator is $O(n)$. The time to find all separators is $O(n \log n)$ since the depth of the recursion is $O(\log n)$. Therefore the total time is bounded by $O(n^2 \log n)$. □

Theorem 3.14 If G is a circle graph, then $t(G) \leq \Delta(G) \lg n + O(1)$.

Proof. If G is a circle graph, then G has a separator of size $\Delta(G)$. By Corollary 3.9, $t(G) \leq \Delta(G) \lg n + O(1)$. □

Theorem 3.15 For almost every graph G, $t(G) \geq (1 + o(1))n/\sqrt{\lg n}$.

Proof. As in Theorem 2.14, it suffices to observe that almost every graph has K_m as a minor, where $m = (1 + o(1))n/\sqrt{\lg n}$. □

4 Algorithms for General Graphs

In this section, we discuss how to solve the connection class problem on a general graph G, adaptively or non-adaptively. In order to use Theorem 2.4 to design adaptive algorithms, or to use Theorem 3.8 to design non-adaptive algorithms, the key is to find small vertex separators S whose deletion leaves only small components.

Unfortunately, there are few practical algorithms for finding good vertex separators in general graphs. Kanevsky's algorithm [12], used in [6], does not find balanced partitions and has running time $O(2^k n^3)$, where k is the connectivity of the graph. Every dense graph G has K_m as a minor, where $m = (1 + o(1))n/\sqrt{\lg n}$ [3]. Therefore G contains a subgraph of connectivity m, on which Kanevsky's algorithm needs exponential time. In contrast, there are extensive results on finding edge separators for general graphs. An *edge separator* of a graph G is an edge set whose removal disconnect G. Leighton and Rao [17] presented provably good approxmation algorithms for edge separators that partition graphs into subgraphs of equal sizes. There are also many good heuristics such as the Kerninghan-Lin algorithm [15]. It is easy to transform edge separators into vertex separators, where by *vertex separator* we mean that we are imposing a bound on the order of components left by deleting the vertex set.

Theorem 4.1 If G is a graph $C \subseteq E(G)$, then G contains a vertex set S of size at most $|C|$ such that the components of $G - S$ are subgraphs of the components of $G - C$.

Proof. For each edge $v_i v_j \in C$, arbitrarily pick an endpoint and include it in S. We have $|S| \leq |C|$, and $G - S \subset G - C$. Thus the components of $G - S$ are subgraphs of the components of $G - C$. □

We can improve on this simple procedure. Let G' be the subgraph of G with edge set C and vertex set consisting of all endpoints of edges in C. As S we use a vertex cover of G'. There are simple approximation algorithms to find minimum vertex covers within a factor of 2 (see [7], page 134).

The basic idea of our heuristic adaptive algorithm is suggested by Theorem 2.4: find a vertex set S to minimize $\max\{q(G_S), q(G-S)\}$. In general, $q(G-S)$ is small when S is large. Therefore, we start with $|S| = \sqrt{n}$ and increase or decrease the size of S until we balance $q(G_S)$ and $q(G-S)$. We use a constant $\epsilon \in (0,1)$ to specify how balanced we want $A_1(G_S)$ and $A_1(G-S)$ to be. To find a separator in Step 8, we use Theorem 4.1 and graph partition algorithms for edge separator. Step 9 can be computed using any all-pair shortest path algorithm. Step 8 to 15 will be repeated at most $\lg\lg n$ times. Therefore the running time of Algorithm A_1 is mainly the time to find separators.

Note that improving on non-adaptive testing requires that our algorithm make use of the response $Q(S)$. Thus we do not simply recurse on G_S and $G - S$ after the first query; Theorem 2.4 provides only a bound in terms of $q(G_S)$ and $q(G - S)$, but we still must take $Q(S)$ into account to solve a particular instance.

The algorithm iteratively maintains a *component structure* $P = \{(G_j, R_j) : j = 1, 2, \ldots, t\}$, where $\{G_1, G_2, \ldots, G_t\}$ is a collection of adjacency graphs that form a partition of V, and $\{R_1, R_2, \ldots, R_t\}$ is a collection of "chosen" subsets of vertices such that $R_j \subseteq V(G_j)$ and $Q(R_j) = V(G_j)$. This property of the chosen subsets implies that each G_j is a union of components. Algorithm 1 initializes with the component structure $P = \{(G, V(G))\}$, and then makes calls to Procedure $\mathcal{D}$ to refine the partition, reducing the size of the chosen sets at each step. When the algorithm terminates, every R_j is reduced to a single vertex, and therefore every $V(G_j)$ is one component.

Algorithm 1.
Input: Adjacency graph G.
Output: All connection classes.
1: $P \leftarrow \{(G, V(G))\}$.
2: **While** there exists $|R_j| > 1$ **do**
3: $P \leftarrow \mathcal{D}(P)$.
5: Report each $V(G_j)$ as one component.

Procedure $\mathcal{D}(P)$.
Input: A component structure $P = \{(G_j, R_j) : j = 1, 2, \ldots, t\}$.
Output: A refined component structure $P' = \{(G'_j, R'_j) : j = 1, 2, \ldots, t'\}$.
1: **For** $j \leftarrow 1$ **to** t **do**
2: Find $R'_j \subseteq R_j$.
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: **For** each component T of G_j **do**
7: $P' \leftarrow P' \cup \{(T, R'_j \cap V(T))\}$.
9: **Return** P'.

Similarly, we can use Theorem 3.8 to design a non-adaptive algorithm. This is simpler than

the adaptive situation because, as shown in the proof of Theorem 3.8, we can use the actual tests for G_S and $G - S$ to build the test vectors for G. Because the algorithm is nonadaptive, we need more queries but can generate them more simply. The detail of the algorithm is omitted due to the page limitation.

References

[1] F. W. Angelotti, W. A. Briston, K. T. Kaliszewski, and S. M. Douskey. System level interconnect test in a tristate environment. *Proc. 1993 International Testing Conference*, pp. 45–53.

[2] B. Bollobás. *Random Graphs.* Academic Press, New York 1985.

[3] B. Bollobás, P. Catlin and P. Erdős. Hadwiger's conjecture is true for almost every graph. *Europ. J. Combinatorics*, Vol. 1, 1980, 195-199.

[4] J. C. Chan. Boundary walking test: An accelerated scan method for greater system reliability. *IEEE Trans. on Reliability,* Vol. 41, No. 4, pp. 496–503, Dec. 1992.

[5] W.-T. Cheng, J. L. Lewandowski and E. Wu. Optimal diagnostic methods for wiring interconnects. *IEEE Trans. on Computer-Aided Design* Vol. 11, No. 9, pp. 1161–1166, Sept. 1992.

[6] C. Feng, W. Hang and F. Lombardi. A new diagnosis approach for short faults in interconnects, *Proc. 1995 Fault Tolerant Computing Symposium*, pp. 331–339.

[7] M. R. Garey and D. S. Johnson, *Computers and Intractability – A Guide to the Theory of NP-Completeness,* Freeman, New York, 1979.

[8] M. Garey, D. Johnson, and H. So. An application of graph coloring to printed circuit testing. *IEEE Trans. on Circuits and Systems*, Vol. CAS-23, No. 10, pp. 591–599, Oct. 1976.

[9] P. Goel and M. T. McMahon. Electronic chip-in-place test. *Proc. 1982 International Test Conference*, pp. 83-90.

[10] A. Hassan, J. Rajski and V. K. Agarwal. Testing and diagnosis of interconnects using boundary scan architecture. *Proc. 1988 International Test Conference*, pp. 126–137.

[11] N. Jarwala and C. W. Yau. A new framework for analyzing test generation and diagnosis algorithms for wiring interconnect. *Proc. 1989 International Test Conference*, pp. 63–70.

[12] A. Kanevsky. Finding all minimum size separating vertex sets in a graph. *Networks*, Vol. 23, No. 6, pp. 533–541, Sept. 1993.

[13] W. H. Kautz. Testing for faults in wiring networks. *IEEE Trans. on Computers* Vol. C-23, No. 4, pp. 358–363, April 1973.

[14] L. Kavraki, J.-C. Latombe, R. Motwani and P. Raghavan. Randomized query processing in robot motion planning. *Proc. 1995 ACM Symposium on Theory of Computing.*

[15] B. W. Kerninghan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Systems Technical Journal*, Vol. 49, pp. 291–307, 1970.

[16] A. V. Kostochka. The minimum Hadwiger number for graphs with a given mean degree of vertices. *Metody Diskret. Analiz.* Vol. 38 pp. 37–58, 1982 (in Russian).

[17] F. T. Leighton and S. Rao. An approximation max-flow min-cut theorem for uniform multicommodity flow problems with application to approximation algorithms. *Proc. 1988 IEEE Symposium on Foundations of Computer Science*, pp. 422–431.

[18] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, New York, Wiley, 1990.

[19] J.-C. Lien and M. A. Breuer. Maximal diagnosis for wiring networks. *Proc. 1991 International Test Conference*, pp. 96–105.

[20] B. Lindström. A theorem on families of sets. *J. Combinatorial Theory (A)*, Vol. 13, pp. 274–277, 1972.

[21] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM J. Computing*, Vol. 36, No. 2, pp. 177–189, April 1979.

[22] D. McBean and W. R. Moore. Testing interconnects: A pin adjacency approach. *Proc. 1993 European Test Conference,* pp. 484–490.

[23] W. Shi and W. K. Fuchs. Optimal interconnect diagnosis of wiring networks. *IEEE Trans. on VLSI Systems*, Vol. 3, No. 3, pp. 430–436, Sept. 1995.

[24] A. Thomason. An extremal function for contractions of graphs. *Math. Proc. Cambridge Philos. Soc.* Vol. 95, no. 2, pp. 261–265, 1984.

[25] H. Tverberg. On equal unions of sets. *Studies in Pure Mathematics,* edited by L. Mirsky, Academic Press, pp. 249–250, 1971.

[26] C. W. Yau and N. Jarwala. A unified theory for designing optimal test generation and diagnosis algorithms for board interconnects. *Proc. 1989 International Test Conference,* pp. 71–77.