Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

ON CONNECTIVITY VERIFICATION IN MULTI-CHIP MODULE SUBSTRATES

A. Kahng
G. Robins
E. Walkup

October 1991
CSD-910074

On Connectivity Verification in Multi-Chip Module Substrates*

Andrew B. Kahng, Gabriel Robins and Elizabeth A. Walkup†

Dept. of Computer Science, UCLA, Los Angeles, CA 90024-1596
†Dept. of Computer Science, Univ. of Washington, Seattle, WA 98195

Abstract

Multi-chip module packaging techniques present several new technical challenges, notably substrate testing. We formulate substrate testing as a problem of connectivity verification in trees via *k-probes*; this paper presents a linear-time algorithm which computes a minimum set of probes achieving complete open fault coverage. Since actual substrate testing also involves the scheduling of probe operations, we formulate efficient probe scheduling as a special type of metric traveling salesman optimization and give two effective heuristics. Empirical results using both random and industry benchmarks demonstrate reductions in testing costs of up to 21% over the best previous methods. The paper concludes with generalizations to alternate probe technologies and several open problems.

Key phrases: Multi-chip modules, VLSI testing, graph algorithms, open fault detection, interconnect testing and verification, circuit probing, integrated circuit reliability.

1 Introduction

Multi-chip module (MCM) technology has recently emerged as an economically viable means for packaging complex, high-performance systems [2] [7] [16] [20]. Traditionally, system performance is limited by interconnection delays at the upper levels of the hierarchy (e.g., printed circuit board or backplane), and may be improved by increasing circuit density and die size. However, as we approach wafer-scale integration, poor manufacturing yield and incompatibility with mixed technologies make such a monolithic system implementation unattractive. The MCM approach resolves this dilemma, allowing high circuit density and yield while decreasing interconnect delay.

MCMs eliminate individual integrated circuit (IC) packages, allowing die to be situated closer together. This shortens interconnect length and enables up to a three-fold increase in

*This research was supported in part by NSF grant MIP-9110696, by an IBM Graduate Fellowship (Robins), and by an NSF Graduate Fellowship (Walkup).

clock frequency, a seven-fold decrease in area, and a 30% decrease in power consumption over the best values achievable using high-density printed circuit boards (PCBs) [4]. A typical MCM (see Figure 1) consists of a substrate containing inter-chip wiring, upon which are mounted a number of bare die. The MCM substrate is made of silicon, alumina, or cofired ceramic, and usually consists of multiple layers (up to thirty or more wiring layers, as well as power and ground layers). The integrated circuits are bonded to pads of the "chip layer" of the substrate using solder bumps or tape-automated bonding (TAB) technology [17].

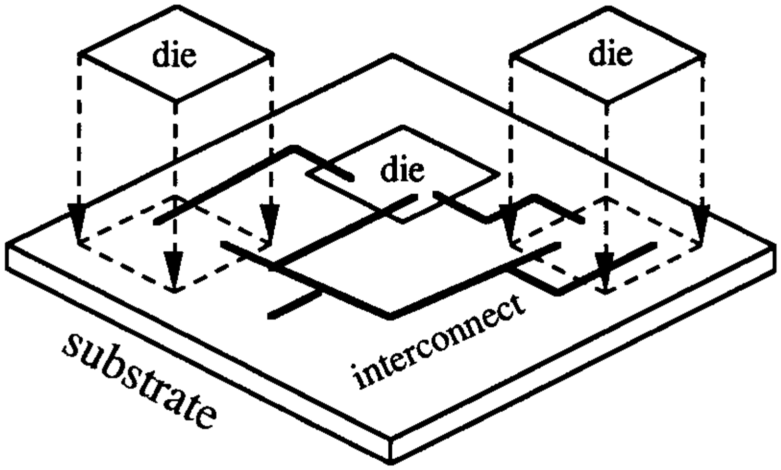

Figure 1: An example of a multi-chip module, showing the underlying substrate containing the interconnect, as well as several mounted die.

The increased use of multi-chip module packaging for large, high-performance systems has focused attention on several new and challenging CAD problems, especially those related to layout, thermal reliability, and testing [6] [16]. Testing in particular presents one of the most persistent challenges of the MCM approach [1] [7] [20]. It is desirable to discover defects in the MCM substrate as early as possible, since the cost of locating and fixing a system fault increases geometrically with each successive stage of the system manufacturing and marketing process. Certainly, the fully-assembled MCM package can be tested using combinatorial IC testing techniques. However, the pre-assembly MCM substrate simply contains a set of disjoint wiring connections with no active devices; thus, the substrate cannot be tested using conventional techniques. With this in mind, the present work addresses verification of electrical connectivity in MCM substrates.

We model the interconnect in the MCM substrate as follows. A *net* is a set of pins p_i that are to be electrically connected. Each signal net is routed on multiple routing layers using a tree topology, where we assume without loss of generality that each leaf is a net terminal, each edge is a wire segment on a single wiring layer, and each internal node is a *via* between two or more routing layers (Figure 2). We wish to verify that the routing topology of each net is properly implemented, with no faults.

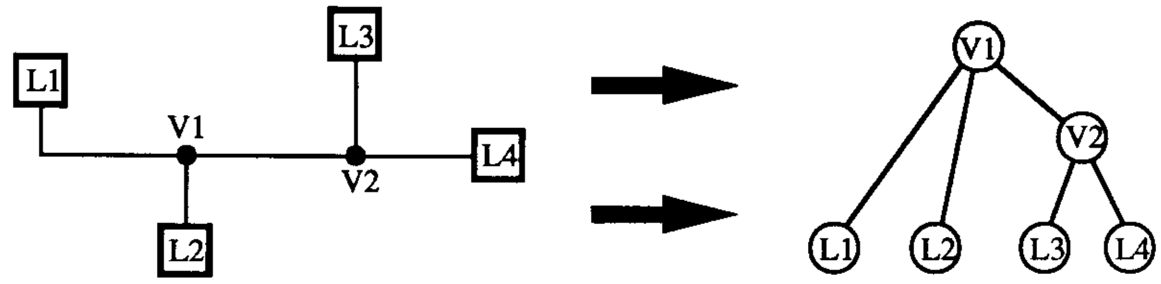

Figure 2: A sample net (left) and its corresponding tree representation (right); pins become leaf nodes while vias become internal nodes.

Two fault classes are of interest in MCM substrate testing: open faults, and short faults. An open fault is an electrical disconnection between two points that are to be connected. As will be discussed in Section 2 below, there are two types of open faults: wire opens, which correspond to edge failures in the tree topology, and cracked vias, which correspond to a physical form of node failure which arbitrarily disconnects subtrees and is not necessarily detected by tests designed to cover wire opens. A short fault is defined to be an electrical connection between two nets that are not intended to be connected.

Traditional methods for connectivity checking involve either parallel probing of the circuit under test, or combinatorial exercising of the logic, neither of which apply to MCM substrate testing [2]. In verifying connectivity for PCBs, a bed-of-nails tester will simultaneously access every grid point, yielding an efficient, parallel checking procedure. However, this idea cannot be applied to MCMs as feature sizes are too small to allow use of such a grid-based methodology. A combinatorial approach, e.g., the boundary-scan method for hierarchical design, requires system-specific, built-in test circuitry [9] [18]. In general, this method will apply only to a completely assembled MCM, but not to a substrate which contains isolated interconnect with no active circuit elements.

Several groups have recently proposed new methods for verifying circuit connectivity during MCM manufacturing. Each of these new methodologies relies on *sequential probing* of the

MCM substrate, in contrast to the standard approaches above which use parallel probing or combinatorial testing.

Golladay et al. [6] propose an electron-beam method to test MCM substrates for short/open faults by injecting charge into individual nets and then scanning them for faults. Unfortunately, electron-beam testers typically have a relatively small working window of access to the chip/substrate, so that probing a location outside that window requires physical motion of an apparatus. Compounding this drawback is the fact that an electron-beam may require a long time to charge up large nets, so that a testing methodology based on this process can be prohibitively slow [15].

All other sequential probing approaches involve variants of k-probe testing, where k "flying" probe heads simultaneously move around the circuit, measuring resistance and capacitance values to determine the existence of shorts between pairs of nets and opens between two pins of a single net. Formally, we define a ***k*-probe** to be a set of k distinct net terminals which are visited simultaneously by k movable probe heads.[1] A single k-probe simultaneously verifies all $\binom{k}{2}$ paths between pairs of terminals in the probe set by measuring resistance and capacitance values. For example, when $k = 2$ the unique path between the two terminals is checked. A 2-probe sequential testing approach developed by Crowell et al. [3] for bare-board testing has been adopted by certain MCM manufacturers [12]. The method of Crowell et al. uses only one probe for each net in the layout, placing the probe heads on the two pins of the net which are physically farthest apart. Unless the measured resistance deviates significantly from the value predicted for the correct circuit, one assumes that no open fault exists. Similarly, only when capacitance is far from the predicted value will a possible short fault between two nets be investigated carefully.

The algorithm of Crowell et al. [3] is efficient in that it uses just one probing operation per net. However, an unfortunate choice of probe locations may yield measured capacitance and resistance very similar to the predicted values, even in the presence of a fault. For example, an open fault caused by a disconnected pad will be detected only by directly probing a path through the pad itself; probing any other path will fail to notice the small deviation in capacitance and resistance values. Indeed, the number of pads in the net induces a lower bound on the number of probe operations needed for fault coverage.

[1]Current probe technology generally uses $k = 2$, but probe machines with higher values of k are currently under development [14].

As noted above, the incomplete fault coverage afforded by such methods as [3] is economically unacceptable. Thus, MCM manufacturers are now adopting substrate test methodologies which provide complete open fault coverage for all nets [14] [19]. With this in mind, Yao et al. [19] have recently proposed a quadratic-time algorithm that determines a set of 2-probes which will check for all possible open faults. It turns out that sufficient capacitance measurements are taken during the open fault checking process to determine whether two nets have been shorted together (i.e., we will encounter a capacitance value that is too high) [3] [19]. Thus, the remainder of this discussion is confined to the issue of complete open fault coverage. In this paper, we give a linear-time algorithm which for any $k \geq 2$ determines a k-probe set which accomplishes complete open fault coverage of each net. The number of probes used by our method is the minimum possible.

Once probes are found which adequately test the required classes of open faults, one must be schedule the probes for execution by a mechanical tester. Obtaining a good schedule is critical, especially with large production runs. Previous groups [3] [19] have used generic greedy or iterative traveling salesman heuristics to attack this problem. In this paper, we propose two effective heuristics for probe scheduling based on new observations concerning the metricity and allowable structure of the probe set.

The remainder of this paper is organized as follows. In Section 2, we formulate optimal open fault detection as a *tree testing* problem and present linear-time algorithms which find an optimal number of probes to cover all possible open faults. Section 3 shows that probe scheduling to minimize total travel time is a form of metric traveling salesman problem (TSP); we present two effective heuristics, one of which has small constant-factor error bound for scheduling any given set of probes. Section 4 gives experimental results on random and industry benchmark layouts, and Section 5 concludes with directions for future research.

2 Open Fault Detection

In this section we address the following:

Minimal Probe Generation (MPG) Problem: Given a routing topology for a signal net with l leaves (i.e., pins), determine a minimum set of k-probes needed to verify the net routing.

We consider two levels of open fault coverage: (i) coverage of all open faults on wire segments, and (ii) coverage of all open faults on wire segments and "cracked" vias (see below). This section presents optimal solutions for the two corresponding versions of the MPG problem. Due to the nature of current probing technology, the discussion assumes $k = 2$; extensions to arbitrary k are straightforward.

2.1 Optimal Detection of Wire Open Faults

In order to test the integrity of all wire segments, certainly every segment which is incident to a pin must be tested. Thus, the number of pins l (leaves in the routing topology) induces a lower bound of $\lceil \frac{l}{2} \rceil$ probes when $k = 2$. Our probe generation algorithm orders the pins of a net as $p_1, \ldots, p_l$ via an arbitrary in-order traversal of the routing tree. Choosing the $\lfloor \frac{l}{2} \rfloor$ probes $\{p_i, p_{i+\lfloor \frac{l}{2} \rfloor}\}$, $1 \le i \le \lfloor \frac{l}{2} \rfloor$, will cover all edges of the tree, as illustrated in Figure 3; if l is odd, an additional probe $\{p_1, p_l\}$ is generated. Figure 4 gives the formal algorithm statement.

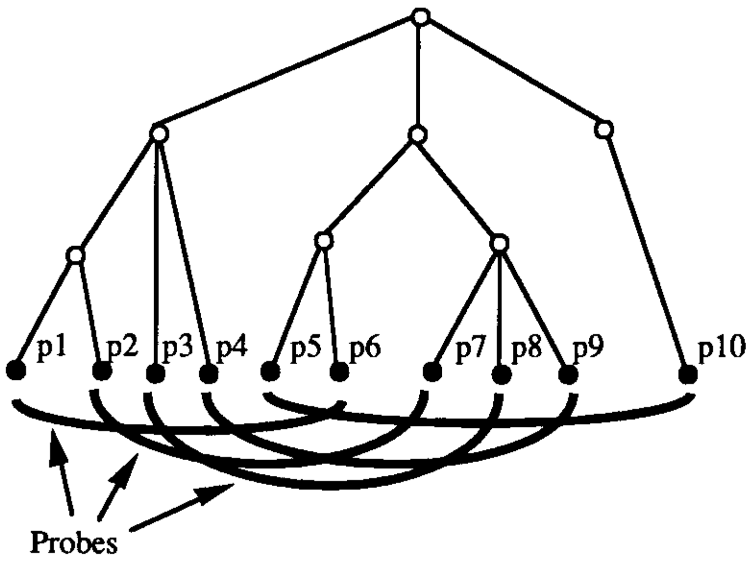

Figure 3: Selecting a minimal set of probes to detect the existence of any wire open faults. The probes $\{p_i, p_{i+\lfloor \frac{l}{2} \rfloor}\}$, $1 \le i \le \lfloor \frac{l}{2} \rfloor$, provide complete wire open fault coverage.

Theorem 1: Given a net whose routing tree topology has l leaves, $\lceil \frac{l}{2} \rceil$ 2-probes are sufficient for complete wire open fault testing.

Proof: A graph is *bridge connected* if for every edge there exists some simple (i.e., vertex-disjoint) cycle containing that edge. Starting with the original tree, for each probe we add the

```
ALG1: Optimal probe set generator for wire open fault detection
Input: A routing tree topology with $l$ leaves
Output: A minimal set of probes for detecting wire open faults
Root the tree arbitrarily at an internal node
Induce an in-order labeling $p_1, \ldots, p_l$ of the leaves
Output the probes $\{p_i, p_{i+\lfloor\frac{l}{2}\rfloor}\}$, $1 \le i \le \lfloor\frac{l}{2}\rfloor$
If $l$ is odd Then Output the probe $\{p_1, p_l\}$
```

Figure 4: **ALG1:** Optimal detection of all wire open faults.

corresponding edge connecting the two leaves of the probe into the graph. A set of probes is sufficient to test for all wire opens if and only if it induces a bridge connected graph. In order to convert a tree into a bridge connected graph via the addition of a minimum number of new edges, it suffices to add the $\lceil\frac{l}{2}\rceil$ new edges $\{p_i, p_{i+\lfloor\frac{l}{2}\rfloor}\}$ for all $1 \le i \le \lfloor\frac{l}{2}\rfloor$, where $p_1, \ldots, p_l$ is the leaf sequence of the tree in any in-order traversal of the tree (when l is odd, the additional probe $\{p_1, p_l\}$ is used as well).

To see that every edge in the resulting graph $G = (V, E)$ lies on some simple cycle, observe that for every proper subtree in the original tree, there exists an edge in G connecting one of the leaves of that subtree to a leaf *not* in that subtree. Given an arbitrary edge $e = \{v_i, v_{i'}\}$ in the original tree (Figure 5), where v_i is the father of $v_{i'}$, one simple cycle that surely contains the edge e is $v_{i'}, \ldots, v_j, v_k, \ldots, v_m, \ldots, v_i, v_{i'}$ where v_j is any leaf in the subtree T' rooted at $v_{i'}$ such that v_j is connected (by a "probe edge") to a leaf v_k that is not in T', and v_m is the lowest common ancestor of both v_i and v_k. To see the existence of v_j and v_k, assume toward a contradiction that every leaf $v_{j'}$ in T' is connected by a probe to $v_{k'}$ which is also in T'. Because of the in-order labeling, leaves $v_{\lfloor\frac{l}{2}\rfloor}$ and $v_{\lfloor\frac{l}{2}\rfloor+1}$ are also in T'. Our assumption that all probes of leaves of T' are internal to T' then implies that v_1 must also be in T', along with v_l when l is even or v_{l-1} when l is odd. In the case where l is even, T' will contain all leaves of the input tree topology, contradicting the fact that T' is a proper subtree of the topology. For l odd, a probe which tests the sole leaf v_l not in T' must connect v_l to a leaf in T', contradicting the assumption that probes involving leaves in T' are internal to T'. □

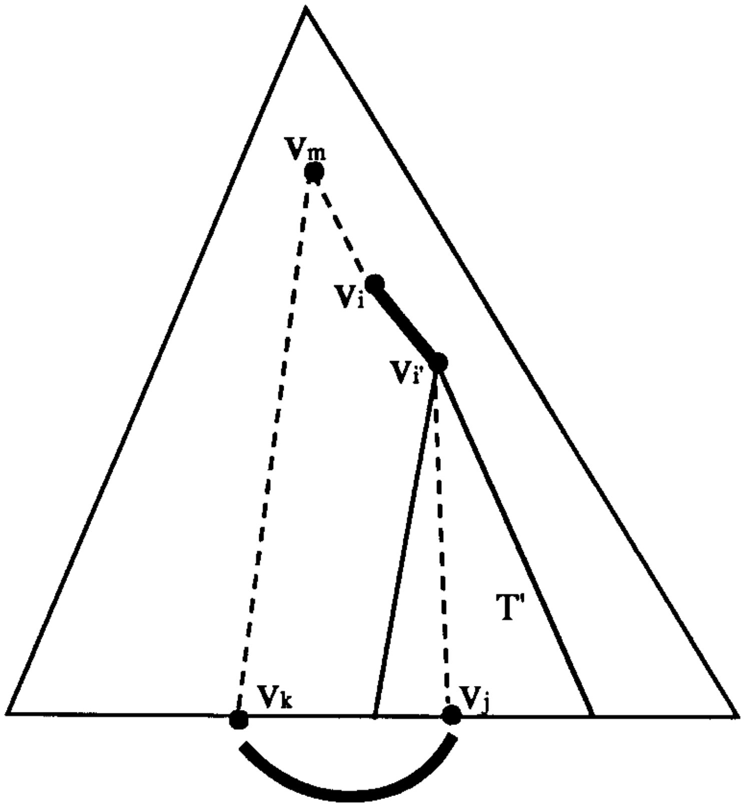

Figure 5: ALG1 checks each edge $e = \{v_i, v_{i'}\}$ for an open wire fault using a probe which forms a simple cycle containing e, as shown.

2.2 Optimal Detection of Cracked Via Faults

In manufacturing the MCM substrate, a via can physically "crack" due to such factors as misalignment in lithography or thermal stress. In other words, subtrees rooted at this internal node of the net can become electrically separated (see Figure 6) [19], so that certain sets of probes will detect this open fault, while other sets will fail to find the cracked via. This section gives a linear-time algorithm, which we call ALG2, that tests for *both* wire faults and cracked vias using the minimum possible number of probes.

ALG2 begins by rooting the tree topology at an internal node R of maximum degree d and then orienting all edges towards R. The algorithm then continues with each leaf node sending to its parent a message list containing its label. When a given node has received message lists from all of its children, it iteratively generates probes by pairing labels from distinct incoming

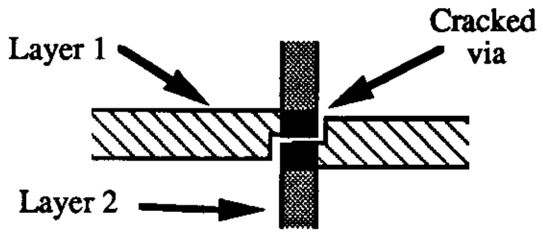

Figure 6: A cracked via in a routing. The two routing layers are depicted using different shadings, while the cracked via (depicted in black) disconnects the circuit as shown.

lists, at least one of which contains more than one node label; when the sum total of remaining labels at that node has been reduced to less than $d+1$, all remaining labels are concatenated and sent to the node's parent. This process is repeated at each node until only the root remains unprocessed, where a simple cleanup step is then performed. Figure 7 traces the execution of ALG2 on a small example, while Figure 8 gives the formal statement of ALG2.

Theorem 2: Given a net routing topology, ALG2 generates a set of probes sufficient to test for all wire open and cracked via faults.

Proof: by induction on the number of leaves in the tree.
Basis: Any tree of depth one is fully tested by ALG2, since in that case we test one leaf with all others.
Induction: Assuming that our algorithm completely tests any tree with k leaves, we show that it also completely tests any tree with $k+2$ leaves. Let T be a tree with $k+2$ leaves and let l_1 and l_2 be the first leaves paired together to generate a probe when we apply ALG2 to T. Let T' be the tree with k leaves which results when we remove l_1 and l_2 from T. Let v be the internal node whose message lists we are examining when we generate the probe $\{l_1, l_2\}$. We claim that after generating the probe $\{l_1, l_2\}$, the message lists at v are precisely those that arrive at v' (the node in T' that corresponds to v in T), and so the rest of the execution of ALG2 on T is the same as the execution of ALG2 on T'. Since v is the first node at which we generate a probe, it must be the case that just before l_1 and l_2 are matched, the message lists at v contain the names of all leaves which are descendants of v. When l_1 and l_2 are removed, the new lists contain the names of all descendants of v' and so the algorithm proceeds as for the tree T' with k leaves. Clearly our probe sequence will test that l_1 and l_2 are connected; the continuation of

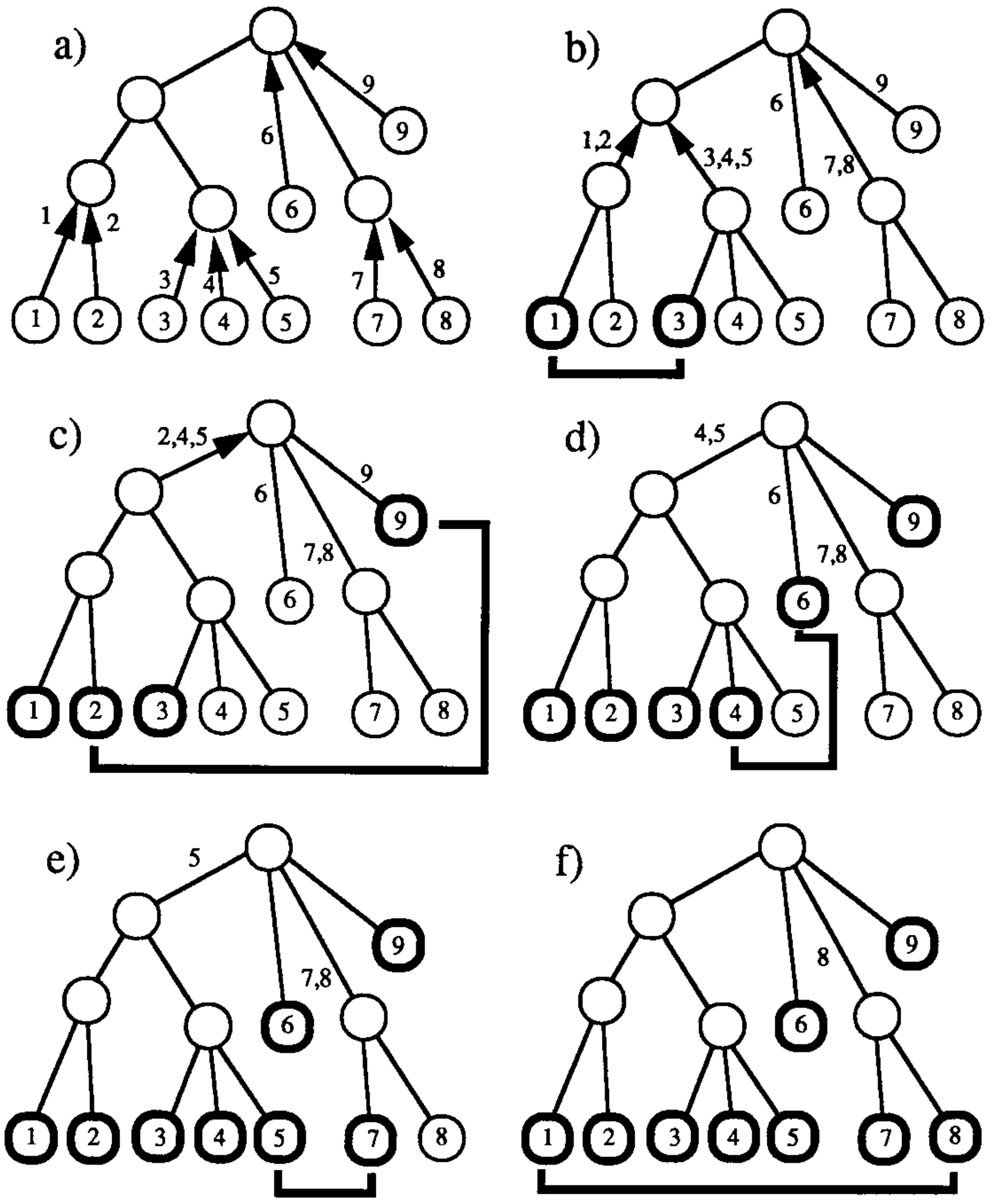

Figure 7: A sample run of ALG2 on a net topology containing 9 pins and 5 vias; a total of five probes are generated (thick arcs).

ALG2 on T' will generate a probe sequence which tests that T' is connected; and so we need now only show that the connectivity between l_1 and T' is also tested. If v is not the parent

```
ALG2: Optimal probe set generator for wire open and cracked via detection
Input: A routing tree topology $(V, E)$
Output: A minimal set of probes for detecting wire open faults and cracked vias
Root the tree at any via $R \in V$ of maximum degree $d$, and Direct all edges towards $R$
Each leaf node $v$ sends the message $\{v\}$ to $parent(v)$
While $\exists v \in V$, $v \neq R$ having received messages $M_1, \ldots, M_{deg(v)-1}$ from its $deg(v) - 1$ children
    While $\Sigma_{k=1}^{deg(v)-1} |M_k| > d$ and $\exists i$ such that $|M_i| \geq 1$
        Let $x \in M_i$
        Let $y \in M_j$ for some $j \neq i$, $|M_j| > 0$
        Generate probe $\{x, y\}$
        $M_i = M_i - \{x\}$
        $M_j = M_j - \{y\}$
    Let $L = M_1 \cup \ldots \cup M_{deg(v)-1}$
    Send $L$ to $parent(v)$
    $V = V - \{v\}$
When $V = \{R\}$ and $R$ has received messages $M_1, \ldots, M_d$ from its $d$ children
    While $\exists i, j$, $1 \leq i, j \leq d$, $i \neq j$ such that $|M_i| \geq 1$, $|M_j| \geq 2$
        Reorder messages $M_1, \ldots, M_d$ such that $|M_i| \leq |M_{i+1}|$ for all $1 \leq i < d$
        Let $k \leq d$ be as small as possible such that $|M_k| > 0$
        Let $x \in M_1$
        Let $y \in M_k$
        Generate probe $\{x, y\}$
        $M_1 = M_1 - \{x\}$
        $M_k = M_k - \{y\}$
    Let $L = M_1 \cup \ldots \cup M_d$
    If $|L| > 1$ Then generate probes $\{L_1, L_i\}$ $\forall$ $2 \leq i \leq |L|$ and terminate
                Else Choose any $v \in V$ such that $v$ and $L_1$ were not passed up by same child
                    Generate probe $\{L_1, v\}$ and terminate
```

Figure 8: **ALG2:** Optimal detection of both wire open and cracked via faults.

of both l_1 and l_2, then some edge on the path from l_1 to l_2 in T coincides with an edge of T'. Since the mutual connectivity of all edges on this path is tested, by transitivity the connection of all of these edges to T' is also tested. Conversely, suppose v is the parent of both l_1 and l_2. Both l_1 and l_2 pass to v message lists of length 1, but the only time we generate a probe from two singleton message lists is at the root when all message lists have length 1, a phenomenon which can occur only as part of the basis case.

Note that while our induction is on the number of leaves in the tree, every tree with more than one leaf can be built from the basis case. There is only one tree with two leaves: the tree with a root and two leaves at depth 1. There are two possible trees with three leaves: the tree with a root and three leaves at depth one; and the tree with a root which has one leaf child at depth one, and one internal node at depth one, which in turn has two children which are leaves at depth two. Of these three trees, only the last does not fit the basis case, but it will after one more probe is generated. Any other tree with two or three leaves will never be given as input to the algorithm since its root will not have degree d, and will not arise during the induction since that would violate Lemma 7 below. □

We now use a sequence of lemmas to prove that ALG2 uses the minimum possible number of probes.

Lemma 1: A node of degree d requires $d-1$ probes to test whether it is cracked.

Proof: Testing an internal node for "cracks" using probes is analogous to connecting a set of d vertices by edges until a single tree connects all the vertices. The result is immediate since a tree with d vertices contains exactly $d-1$ edges. □

Lemma 2: No node passes up more than d leaf names in its message list.

Proof: by induction on the maximum distance from a node to any one of its leaves.
Basis: If the distance is zero then the node is a leaf and passes up one name in its message list.
Induction: Suppose the node does send up a message list containing more than d leaf names. Since, by the induction hypothesis, each child of the node sent up at most d leaf names, the propagated leaf names must come from the message lists of two or more children. But under such conditions, the algorithm will generate probes using leaf names from different message lists until either each child's message list has length one, or the total message count is d or less. In the first case, since the node received messages from at most $d-1$ children, it can send up a message list of length at most $d-1$. In the second case, the node sends up a list of length at most d. □

Lemma 3: No internal node passes up fewer than $d-1$ leaf names in its message list, unless the list contains the names of all leaves that are descendants of that node.

Proof: by induction on the maximum distance from a node to any one of its leaves.
Basis: If the distance is zero then the node is a leaf and it passes up the message list containing its name.
Induction: Suppose a node passes up k leaf names where $k < d-1$ and that the node has t descendants where $t > k$. Since $k < d-1$, no probes could have been generated at that node, and k must be the sum of the lengths of all message lists sent by the node's children.

Each of these children must have sent lists of length no greater than k, and so by the induction hypothesis, each must have sent lists containing all their leaf descendants. The union of these lists is precisely the list of the descendants of the node, and so we have a contradiction. $\square$

Lemma 4: If any combination of $2(d-1)$ or more leaf names are present among d non-empty message lists at the root node and the difference in length between the longest two message lists is no more than $d-1$, then either all leaf names appear in the probe sequence exactly once, or else one leaf name appears twice and all others appear once.

Proof: by induction on d.

Basis: $d = 2$. There are two message lists. If they both have the same length, then they are completely matched with each other and no leaf names are repeated. If one has length one longer than the other, then one leaf will need to appear a second time in order to match the single leaf remaining after all the matches have been made.

Induction: Matches at the root are made by the algorithm until one of the incoming message lists has length one. At this point there are at least $2(d-1)$ leaves distributed among the message lists. The next match leaves $d-1$ non-empty message lists. The difference in lengths between the two longest message lists has decreased by one unless these lengths were already equal. The total number of leaves remaining is at least $2(d-1-1)$ among $d-1$ lists. We then invoke the induction hypothesis for $d-1$. $\square$

Lemma 5: If any combination of $2(d-1)-k$ leaf names is present among d non-empty message lists at the root node, then exactly one of the leaf names generated in the probe sequence will appear $k+1$ times in the generated probe sequence, and every other leaf name will appear exactly once.

Proof: by induction on d.

Basis: If $d = 2$ and $k = 0$ then we have two singleton lists, and we match them together without duplicating any leaves.

Induction: If there exists any message list with length greater than one, we match it with a list of length one and invoke the induction hypothesis. (There must be a list of length one, else we would have $2(d-1)$ or more leaf names present.) If no list has length greater than one, then $2(d-1)-k = d$, or $d = k+2$. We then must use one of the leaves $d-1 = k+1$ times. $\square$

Lemma 6: If $k > 0$ and $2(d-1)-k$ leaf names arrive at the root, then there are exactly $2(d-1)-k$ leaves in the tree.

Proof: Suppose that at least one child of the root sent up a message list of length $d-1$ or more. There are $d-1$ other children, each of which must have sent at least one leaf name. This would imply a total of at least $2(d-1)$ leaf names at the root, which is too many; hence, no child of the root sent up $d-1$ leaf names. Applying Lemma 3, we find that all leaf descendants of the root must appear in the message lists received by the root, and so there must be exactly $2(d-1)-k$ leaves in the tree. □

Lemma 7: No child of the root is left with a message list of length greater than one when all other lists have reached length zero.

Proof: Suppose that when we get to the step $L = M_1 \cup \ldots \cup M_d$ when $V = \{R\}$ that one of the M_i's has size $|M_i| > 1$. There must be only one such M_i, else we would have continued to generate probes. Consider the last $d-1$ probes generated: each must have taken one element from M_i, and M_i must always have been the maximum-length message list at the root since no other M_j has length within one of M_i. Hence, M_i must have started with length $d+1$, but Lemma 2 guarantees this will not happen, and so $|M_i| \leq 1$. □

Theorem 3: ALG2 generates the optimum number of probes for complete wire open and cracked via testing.

Proof: Let l be the number of leaves in the tree. If $2(d-1)$ or more leaf names arrive at the root, then by Lemma 4, we have either l or $l+1$ leaf names used in the sequence of probes, which is optimal since every leaf name must appear in at least one probe, or else not all wires have been tested. If $2(d-1)-k$ leaf names arrive at the root, then by Lemma 6 we have $l = 2(d-1)-k$, and by Lemma 5 the sequence of probes contains $l+k = 2(d-1)$ leaf names. There are then $d-1$ probes, which is optimal by Lemma 1. □

Except at the root, each probe generated by ALG2 will remove two distinct leaf node names from the messages (i.e., lists of leaf node names) being passed. At most d leaf names will remain to be processed at the root, requiring at most $d-1$ additional probes. Therefore, to test an l-pin net the optimal number of probes that our algorithm generates is bounded by $\frac{l-d}{2} + (d-1) = \frac{l}{2} + \frac{d}{2} - 1$; this bound is achieved in a star topology (such a topology is theoretically possible given the multi-layer interconnect). Assuming that d is a constant dependent on technology, each node v passes no more than d leaf names up to its parent, and thus each node will certainly receive fewer than d^2 leaf names from its children. Since each node is processed only once, and since the amount of processing at each node is a constant, the overall time complexity is linear in the size of the routing topology, which is clearly optimal.

3 Efficient Probe Scheduling

As noted above, efficient probe scheduling algorithms are necessary because testing cost is largely dependent on the total travel time of the probe heads. In mechanical probing, individual stepper motors will control the $x-$ and $y-$coordinates of each moving head. The distance $dist(A_i, B_i)$ traveled by the i^{th} probe head is given by

$$dist(A_i, B_i) = max[\ |A_{i_x} - B_{i_x}|\ ,\ |A_{i_y} - B_{i_y}|\ ].$$

This distance function (also known as the Chebyshev or L_∞ norm) reflects the fact that the maximum time interval for which any motor is engaged will determine the delay between consecutive probes; such a metric is typical in manufacturing applications and is quite accurate despite second-order effects such as acceleration and deceleration of the moving heads. For k-probes, when $k = 2$, the *cost* of moving the probe heads from a set of pin locations $A = \{A_1, A_2\}$ to another set of locations $B = \{B_1, B_2\}$ is given by

$$c(A,B) = \min\ \{\ \max[\ dist(A_1,B_1)\ ,\ dist(A_2,B_2)\ ]\ \ ,\ \ \max[\ dist(A_1,B_2)\ ,\ dist(A_2,B_1)\ ]\ \}.$$

For $k > 2$, the cost of moving the probe heads from $A = \{A_1, \ldots, A_k\}$ to $B = \{B_1, \ldots, B_k\}$ is given by

$$c(A,B) = \min_{\{\sigma\}}\ max[dist(A_1, B_{\sigma(1)})\ ,\ dist(A_2, B_{\sigma(2)})\ ,\ \ldots\ ,\ dist(A_k, B_{\sigma(k)})]$$

where $\{\sigma\}$ denotes the set of all permutations of the probe indices $\{1, \ldots, k\}$. In other words, we choose the mapping of A onto B in such a way that the maximum travel time of any probe head is minimized (see Figure 9).

In some technologies, each probe head may be carried by its own moving horizontal bar. If all such probe carriers are assumed to lie in the same plane due to the probe-machine construction, collisions between probes become a concern and no two such bars are allowed to cross each other's path. In other words, the y coordinates of the k probe heads must satisfy $y_1 \leq y_2 \leq \ldots \leq y_k$ at all times. Thus, the probe head coordinates are always sorted lexicographically [19]. This constraint clearly yields a metric which we call the *collision-free* metric; in contrast, the metric discussed above will be referred to as the *generalized* metric. The collision-free metric is more restrictive, since there is always a unique feasible permutation of the probe heads in traveling

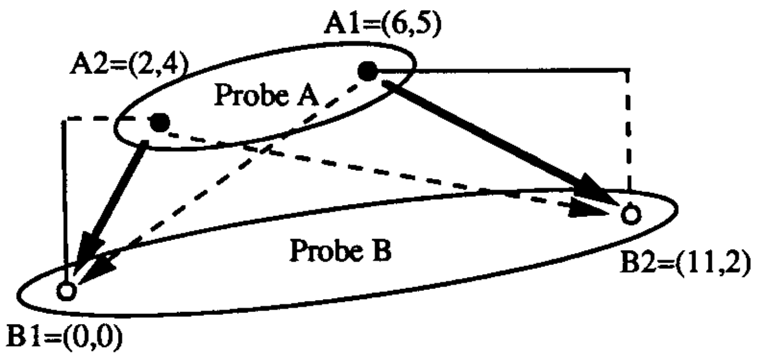

Figure 9: An example showing the distance between two probes $A = \{A1, A2\}$, $B = \{B1, B2\}$. We have $dist(A1, B1) = 6$, $dist(A2, B2) = 9$, $dist(A2, B1) = 4$, and $dist(A1, B2) = 5$; thus, the distance between the two probes A and B is $\min(\max(6,9), \max(4,5)) = \min(9,5) = 5$ (i.e., the best strategy will move one probe head from $A2$ to $B1$ while the other probe head moves from $A1$ to $B2$).

from one set of locations to another. In particular, for $k = 2$ the cost under the collision-free metric of moving the probe heads from $A = \{(x_1, y_1), (x_2, y_2)\}$ to $B = \{(x_3, y_3), (x_4, y_4)\}$, $y_1 < y_2$, $y_3 < y_4$ is given by $\max\{\ |x_1 - x_3|\ ,\ |y_1 - y_3|\ ,\ |x_2 - x_4|\ ,\ |y_2 - y_4|\}$.

The Minimal k-Probe Scheduling (k-MPS) Problem: Given a set of k-probes, minimize the total probe moving cost required in executing all probes.

A straightforward reduction from the geometric traveling salesman problem [5] yields:

Theorem 4: The k-MPS problem is NP-hard.

Proof: We can transform a geometric instance of TSP into an instance of MPS by introducing k copies of each site, then considering each set of k identical copies of a site as a single k-probe. Distances between probes will correspond to the original distances between the corresponding sites in the TSP instance. □

The probe scheduling problem seems quite unapproachable, both due to its theoretical intractability and because the distance and travel cost functions are not easily intuited. Thus, previous work relies on generic traveling salesman heuristics to optimize the probe schedule. For example, when $k = 2$ probe heads are available, Crowell et al. [3] use a bandsort algorithm to optimize the movement of *one* of the probe heads. Of course, the other probe head may be forced to travel very large distances between probes, and indeed the resulting schedule is often exceedingly inefficient. Yao et al. [19] use simulated annealing and the Kernighan-Lin 2-*opt* criterion [11] as the basis of an iterative interchange approach; their schedules save up to 83% of travel costs over the method of [3]. Note that all of the heuristics proposed in [3] and [19] have unbounded error.

In this section, we first show that the k-probe travel costs are actually metric (although clearly not geometric), i.e., distances between k-probes satisfy the triangle inequality for all values of $k \geq 1$. As a consequence, traveling salesman heuristics with constant-factor error bound apply [13]. Second, we exploit flexibility in the choice of probes to find probe sets which can co-exist in an efficient probe schedule.

3.1 Metricity of the k-MPS Problem

For the collision-free metric, the travel costs of the probe heads can be easily seen to satisfy the triangle inequality, since the probe head coordinates are always in lexicographic order. Thus, moving the probe heads from A to C via an intermediary B yields the same final probe permutation as would result by moving directly from A to C. Metricity follows from the metricity of the Chebyshev norm.

For arbitrary k-probes A, B and C, we may view the travel costs $c(A, B)$, $c(B, C)$ and $c(A, C)$ in the generalized metric as being respectively determined by the optimal permutations $\sigma_1 : A \rightarrow B$, $\sigma_2 : B \rightarrow C$ and $\sigma_3 : A \rightarrow C$. Comparing the composed permutation $\sigma_1 \circ \sigma_2 : A \rightarrow C$ with the permutation $\sigma_3 : A \rightarrow C$ yields the following:

Theorem 5: For any three k-probes A, B and C, the travel costs $c(A, B)$, $c(B, C)$ and $c(A, C)$ in the generalized metric satisfy the triangle inequality, i.e., $c(A, B) + c(B, C) \geq c(A, C)$.

Proof: Compare the set of edges of permutation $\sigma_3 : A \rightarrow C$ that defines $c(A, C)$, with the induced permutation $\sigma_1 \circ \sigma_2 : A \rightarrow C$ (see Figure 10). Define $\max(\sigma)$ to be the maximum distance traveled by any probe head according to the permutation σ. Clearly $c(A, C) \leq \max(\sigma_1 \circ \sigma_2)$, since $\sigma_1 \circ \sigma_2$ is not necessarily the minimum-cost permutation between A and C. On the other hand, $\max(\sigma_1 \circ \sigma_2) \leq c(A, B) + c(B, C)$ by the triangle inequality and the metricity of the Chebyshev norm. It follows that $c(A, C) \leq c(A, B) + c(B, C)$. □

Theorem 5 allows us to apply heuristics which achieve bounded error for *metric* TSP instances. In particular, Christofides' combination of a minimum spanning tree construction and matching [13] yields:

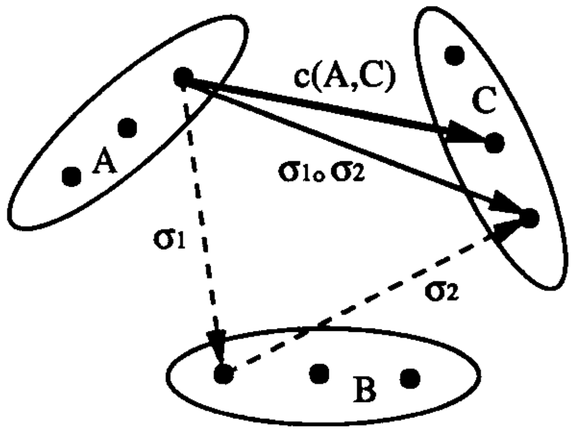

Figure 10: Metricity of the probe travel cost function.

Corollary 1: Given a set of n k-probes, for any fixed $k \geq 1$, a heuristic probe schedule with cost at most $\frac{3}{2}$ times optimal can be found in $O(n^3)$ time. □

3.2 Varying the Probe Set

A further optimization of the tour schedule is possible because *the set of probes is itself variable.* Figure 11 depicts an instance where a "smarter" choice of probes reduces the optimal tour cost by one-quarter. Most tree topologies can be tested with the minimum number of probes in many distinct ways. For example, each three-pin net in Figure 11 can be tested by a minimal set of 2-probes in three distinct ways (i.e., any two probes can be used); in fact, the 2-pin net is the only connection topology with a unique minimum probe set. For special nets such as V_{dd} and ground, the usual MCM architecture allows even more freedom: such nets can be viewed as being implemented by vias to dedicated routing planes, so that *any* decomposition of the terminals into sets of cardinality k will cover all open faults.

We thus obtain a new type of *compatibility TSP* problem, where sets of k-probes are selected to cover every net such that the optimal tour cost for the *union* of all probe sets is minimized. Such a formulation, where there is a synergy between the choice of probes and the optimal tour cost, seems to be new in the literature and is of independent interest.

The Minimal Probe Generation/Scheduling (MPG/S) Problem: Given a routing topology for a signal net, determine and schedule a set of probes so that the total probe moving cost is minimized.

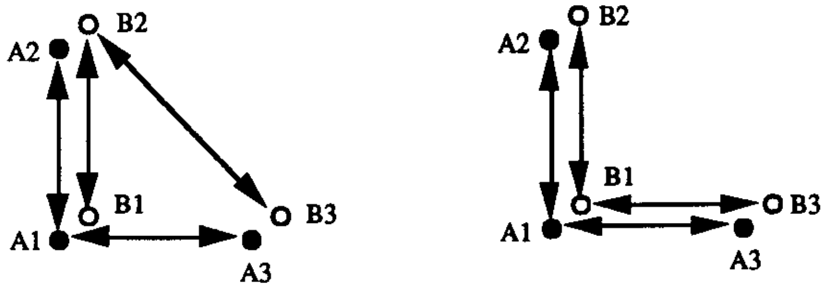

Figure 11: An example of how selecting probes carefully can reduce the total tour length by as much as one-quarter: four probes are required for complete wire open fault coverage over the two 3-pin nets $\{A_1 = (0,0),\ A_2 = (0,1),\ A_3 = (1,0)\}$ and $\{B_1 = (\epsilon,\epsilon),\ B_2 = (\epsilon, 1+\epsilon),\ B_3 = (1+\epsilon,\epsilon)\}$. Assuming that the probe tour must start and end at the origin, the probe set on the left will be optimally ordered as $\{(A_1,A_2)\ ,\ (B_1,B_2)\ ,\ (B_2,B_3)\ ,\ (A_1,A_3)\}$, requiring about four units of travel time. The probe set on the right may be ordered as $\{(A_1,A_2)\ ,\ (B_1,B_2)\ ,\ (B_1,B_3)\ ,\ (A_1,A_3)\}$, requiring only about three units of travel time.

In order to hybridize the probe-generation phase with the tour-scheduling phase, and to take advantage of the non-determinism inherent in the probe selection, we propose the heuristic ALG3 (Figure 12). ALG3 is based on a minimum-cost insertion strategy, i.e., it schedules all probes for a small subset S of nets, then iteratively adds the probe which has lowest insertion cost in the tour while still allowing a minimum probe set.[2] In executing ALG3, we typically choose S to be all nets other than power, ground, and 3-pin nets. That is, we specifically exploit the high degree of freedom in choosing probes for power, ground and 3-pin nets. As seen in the following section, ALG3 yields significantly shorter schedules than existing methods.

4 Experimental Results

We tested our algorithms on an MCM benchmark design obtained from Hughes Aircraft Co., containing 44 components and 199 nets. This is the same benchmark used by Yao et al. in [19]. We also used two randomized versions of the Hughes benchmark, where the same net topologies were retained, but with pin coordinates reassigned randomly from a uniform distribution in the layout region. ALG2 was used to generate minimal probe sets which cover all possible wire

[2]Note that a probe set which allows us to minimize travel cost may have more than the minimum possible number of probes. However, the heuristics discussed in this section require that the number of probes is minimum; the more general optimization is open.

```
ALG3: Insertion-based method for probe selection
Input: A collection $N$ of nets and their routing tree topologies
Output: An efficient heuristic probe schedule
Compute a minimal set of probes $P$ which verifies a subset $S \subset N$ of the nets
Compute a heuristic schedule (tour) $P_1, \ldots, P_m, P_1$ of $P$
While $\exists$ a net not having complete fault coverage
    Find a probe $P^*$ for any net $N_i$ such that
      (i) $N_i$ is still coverable by a minimal number of probes after $P^*$ is added, and
      (ii) the probe's minimum insertion cost between consecutive probes is
      minimized, i.e., $\min_{feasible\ P^*} \min_i \{c(P_i, P^*) + c(P^*, P_{i+1}) - c(P_i, P_{i+1})\}$
    Insert $P^*$ into the tour between probes $P_i$ and $P_{i+1}$,
      where $i$ was the tour index where $P^*$ had minimum insertion cost
```

Figure 12: **ALG3:** An insertion-based heuristic for probe selection.

open and cracked via faults. The schedules for these probe sets were optimized using the 2-opt TSP heuristic, as well as by 2-opt followed by 3-opt (in a separate run).

We also tested a variant of ALG3 on the same benchmark, as described in Section 3.2 above. We first generated a minimal set of probes for all nets other than the power, ground, and nets with 3 or less pins, then computed a heuristic tour for these probes, using the 2-opt TSP heuristic (again, in a separate run, we used 2-opt followed by 3-opt). Finally, we iteratively added additional probes for the remaining nets which (i) could be inserted into the current tour with minimum cost, and (ii) were compatible with previously chosen probes in some minimum probe set. In all cases, a total of 634 probes were generated by our algorithm, the same number as that generated by the algorithm of [19]. With each of the ALG3 experiments, 226 probes were initially chosen to cover the nets which had > 3 pins and which were neither power nor ground; the remaining 408 probes were added incrementally. In the ALG3 experiments, we optionally ran 2-opt improvement after every 10 probes added, and optionally ran 3-opt improvement after every 50 probes added. All of the above benchmarks were run with the collision-free distance function, as well as with the generalized distance function. These results are summarized in Table 1.

As expected, the ALG3 variants, being able to carefully choose probes while constructing the heuristic tour, outperformed ALG2 by a considerable margin. Results are somewhat better when 3-opt is incorporated, also as expected. For the benchmark design, the best tour obtained in [19] using simulated annealing had cost 150,525,000; in comparison, our ALG3 variants

MCM	metric	ALG2 + 2-Opt	ALG2 + 2-Opt + 3-Opt	ALG3 + 2-Opt	ALG3 + 2-Opt + 3-Opt
Hughes	generalized	160,435,000	153,185,000	126,210,000	118,497,000
	collision-free	163,202,000	157,600,000	131,010,000	126,637,500
Random1	generalized	294,164,000	286,679,000	265,276,000	257,838,000
	collision-free	302,684,000	289,843,000	269,346,000	260,897,000
Random2	generalized	295,956,000	285,379,000	271,869,000	260,150,000
	collision-free	304,885,000	294,421,000	270,767,000	263,113,000

Table 1: Performance of ALG2 and ALG3 variants on the industry benchmark and on random examples. Note that the best probe schedule cost obtained by Yao et al. for the industry benchmark, using simulated annealing, was 150,525,000 units. The tour obtained by ALG3 + 2-opt + 3-opt gives savings of up to 21% over this value. Each benchmark was run with the collision-free distance function, as well as with the generalized distance function.

obtain up to 21% improvement over the results of [19]. Since simulated annealing usually gives solutions quite close to optimal [8], our results indeed confirm that careful choice of compatible probes is an important issue.

5 Future Work

Substrate testing for open faults is a critical phase in the production of multi-chip module packages. We have formulated MCM substrate testing as a problem of connectivity verification for trees using k-probes, and presented linear-time algorithms for optimal probe generation. Our algorithms yield minimum probe sets for covering all possible wire open and cracked via faults. Since the associated probe scheduling problem is metric, a bounded-error scheduling heuristic can be obtained. Furthermore, we present an insertion-based heuristic which exploits the special structure of 3-pin and the power/ground nets in the MCM substrate. This heuristic significantly improves probing costs over previous methods.

There are a number of interesting open problems. The fact that many different probe sets can cover a given net yields an interesting TSP variant, as noted above. It is possible that a "prize-collecting salesman" formulation (e.g., at least two of the three possible probes must be "collected" for each three-pin net) can be solved with constant-factor error via an LP-relaxation scheme. This would be quite useful, as the bounded error heuristic of Corollary 1 in Section

3 applies only when all of the probes have been fixed. Analyzing the maximum error inherent in arbitrarily fixing the probes is also of interest. Developments in probe technology will soon allow $k > 2$ probe heads to move simultaneously, affording even greater freedom in choosing the probe sets. Thus, the synergy between choice of probes and the resulting optimal schedule cost will continue to be of significance. More sophisticated strategies for the efficient insertion of probes into a partial tour are possible; for example, we may look for the best combination of added and deleted probes, iterating this tour improvement until no further cost reduction is possible. Finally, the concept of verifying connectivity by checking paths, rather than edges, is quite novel, as is the "physical" node failure mode (via cracking), and can be applied to both trees and arbitrary graphs arising in other fields of study.

6 Acknowledgements

We thank Dr. Brian Tien (now at Cadence Design Systems) and Mr. Ed Shi of Hughes Aircraft Company for access to the benchmark reported in Section 4. We are also grateful to Professor C. K. Cheng, Mr. Nan-Chi Chou and Mr. David Yao of the UCSD CSE Dept., as well as Mr. Tom Russell of ALCOA, for many interesting discussions.

References

[1] R. W. Bassett, P. S. Gillis and J. J. Shushereba, "Testing and Diagnosis of High-Density CMOS Multichip Modules", *Proc. IEEE Workshop on Multichip Modules*, Santa Cruz, March 1991, pp. 108-113.

[2] R. H. Bruce, W. P. Meuli and J. Ho, "Multi Chip Modules", *Proc. Design Automation Conf.*, June 1989, pp. 389-393.

[3] J. C. Crowell, R.J. Keogh, and J.A. Conti, "Moving Probe Bare Board Tester Offers Unlimited Testing Flexibility", *Industrial Electronics Equipment Design*, McGraw-Hill, Sept. 1984.

[4] W. W. Dai, "Performance Driven Layout of Thin-film Substrates for Multichip Modules", *Proc. IEEE Workshop on Multichip Modules*, Santa Cruz, March 1991, pp. 114-121.

[5] M. Garey and D. S. Johnson, "The Rectilinear Steiner Problem is NP-Complete", *SIAM J. of Applied Math.* 32(4) (1977), pp. 826-834.

[6] S.D. Golladay, N.A. Wagner, J.R. Rudert and R.N. Schmidt, "Electron-Beam Technology for Open/Short Testing of Multi-Chip Substrates", *IBM J. Res. Develop.* 34(2/3), March/May 1990, pp. 250-259.

[7] D. Herrell, "Multichip Module Technology at MCC", *Proc. IEEE Intl. Symp. on Circuits and Systems*, June 1990, pp. 2099-2103.

[8] D. S. Johnson, C. R. Aragon, L. A. McGeogh and C. Schevon, "Optimization by Simulated Annealing: An Experimental Evaluation (part 1)", *Operations Research* 37(6) (1989), pp. 865-892.

[9] Joint Test Action Group, "JTAG Boundary-Scan Architecture Standard Proposal", Version 2.0, March 30, 1988.

[10] A. B. Kahng, G. Robins and E. A. Walkup, "On Connectivity Verification in Multi-Chip Module Substrates", *Technical Report* CSD-TR-910074, October 1991.

[11] S. Lin, "Computer Solutions of the Traveling Salesman Problem", *Bell System Technical Journal* 44 (1965), pp. 2245-2269.

[12] B. McWilliams, nChip Inc., *private communication* (invited talk at CANDE meeting), San Marcos, CA, April 1991.

[13] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, 1982.

[14] T. Russell, ALCOA Corp., *private communication*, August 1991.

[15] R. G. Sartore, N. Shastry, U. Brahme, K. Jefferson and R. Halaviati, "Tutorial for Computer Aided Diagnostic E-Beam Testing of ASICs", *Proc. 4th IEEE Intl. ASIC Conf.*, Rochester, Sep. 1991, pp. T8:1.1 - T8:1.7.

[16] K. P. Shambrook, "An Overview of Multichip Module Technologies", *Proc. IEEE Workshop on Multichip Modules*, Santa Cruz, March 1991, pp. 1-6.

[17] M. Taylor and W. W. Dai, "TinyMCM", *Proc. IEEE Workshop on Multichip Modules*, Santa Cruz, March 1991, pp. 143-147.

[18] L.T. Wang, M. Marhoefer and E.J. McCluskey, "A Self-Test and Self-Diagnosis Architecture for Boards Using Boundary Scans", *Proc., First European Test Conf.*, Paris, April 1989, pp. 119-126.

[19] S.-Z. Yao, N.-C. Chou, C.-K. Cheng and T. C. Hu, "A Multi-Chip Module Substrate Testing Algorithm", *Proc. 4th IEEE Intl. ASIC Conf.*, Rochester, Sept. 1991, pp. P9:4.1 - P9:4.4.

[20] S. Weber, "For VLSI, Multichip Modules May Become the Packages of Choice", *Electronics*, April 1989, pp. 106-112.