On Theoretical and Practical Considerations of Path Selection For Delay Fault Testing

Jing-Jia Liou, Li-C. Wang, and Kwang-Ting Cheng
Department of Electrical and Computer Engineering,
University of California, Santa Barbara

ABSTRACT

In current industrial practice, critical path selection is an indispensable step for AC delay test and timing validation. Traditionally, this step relies on the construction of a set of worse-case paths based upon discrete timing models. The assumption of discrete timing models can be invalidated by delay effects in the deep submicron domain, where timing defects and process variation are statistical in nature. In this paper, we study the problem of optimizing critical path selection, under both fixed delay and statistical delay assumptions. With a novel problem formulation and new theoretical results, we prove that the problem in both cases are computationally intractable. We then discuss practical heuristics and their theoretical performance bounds, and demonstrate that among all heuristics under consideration, only one is theoretically feasible. Finally, we provide consistent experimental results based upon defect-injected simulation using an efficient statistical timing analysis framework.

1. INTRODUCTION

Process variations, manufacturing defects, and noise are major factors to affect timing characteristics of deep sub-micron (DSM) designs [1, 2]. The delay effect from these factors can often be continuous in nature [3] [4], to which the traditional assumptions of discrete timing and delay models become less applicable. These continuous factors should better be captured and simulated using statistical models and methods.

In today's industry, one commonly-adopted method is to select the k longest paths for testing, where k depends on the affordable number of test patterns. When selecting critical paths, the notion of being critical depends on the timing length of a path, which is often calculated using discrete delay models based upon nominal or worst-case timing scenarios.

If critical paths are selected for explicit testing, the definition of a path being critical will obviously affect the quality of the tests. Without a rigorous (and practical) definition, the optimization problem of selecting the k "best" paths is not well defined. Consequently, there is no way to analyze the feasibility of a path selection method. Often, the quality of a path selection method and its resulting path set can only be "guessed" via experiments.

In this paper, we formulate the critical path selection as an optimization problem based upon statistical delays and defect occurrences. Through theoretical analysis, we demonstrate that optimization for critical path selection consists of solving two intractable sub-problems. Then, we search for the best method in practice for solving the problem by analyzing various heuristics with respect to their theoretical performance. We conclude that only one heuristic (called "H-Opt") is theoretically feasible.

To demonstrate that our theoretical results are valid in reality, we developed a statistical timing analysis framework that is capable of performing defect-injected simulation. Consistent experimental results are then obtained to confirm our theoretical findings.

This paper is organized into three parts. In section 2, we give a brief introduction about prior work and background in statistical timing model. The second part consists of sections 3 to 7 which include all the theoretical analysis. In section 3, the problem of critical path selection is formally defined. Then, in section 4 we show that optimizing the path selection is to simultaneously optimize two different objectives (call them $\wp_{obj1}$ and $\wp_{obj2}$). In section 5, the problem of optimizing the first objective $\wp_{obj1}$ is analyzed in detail. Then, in section 6, the analysis for optimizing the second objective $\wp_{obj2}$ follows. In section 7, we combine these results to theoretically estimate the performance of different heuristics for critical path selection.

Experimental results are explained in section 8. These results validate the practical application of our theoretical work. The last section concludes the paper.

2. BACKGROUND

Historically, the definition of critical path is based upon the nominal or worst-case timing analysis [5, 6, 7, 8] (i.e., the delay of each cell/interconnect is of discrete timing values based upon either nominal or worst-case delays). In the industry, timing analysis often relies on cell characterization where the earliest, latest, and average signal arrival times are estimated for each pin-to-pin pairs of the cell [9]. With these discrete timing values, the delay of a path can be defined as the accumulated delay on the path. The set of critical paths can then be constructed by selecting either a fixed number of the longest paths, or all paths that fall into a pre-defined time range. If circuit segment coverage is considered, then the set of critical paths can include, for each signal segment, the timing longest path. Such a set of critical paths may also ensure a complete topological coverage of the circuit [5].

In deep sub-micron testing, delay variations resulted from manufacturing process, small defects, and/or signal noise can alter the discrete timing assumptions in the delay models. Consequently, the sets of critical paths in different chip instances can be significantly different. It is then questionable that testing a set of critical paths defined based upon the traditional discrete timing models would still be effective in the DSM domain.

From the above perspective, the definition of critical path can no longer be deterministic. Instead, the most critical path should be defined as the one which has the highest probability of being "critical" when a large number of the chip instances are produced [10]. This probabilistic perspective suggests that more sophisticated analysis and simulation methods are needed in order to identify the set of critical paths, and to accurately estimate the return of testing these paths [11].

The new definition of critical path above does not directly imply an optimal path selection strategy. If we simply select the k most critical paths based upon their *critical probabilities*, the resulting path set may not be the optimal set for delay testing. For example, consider two paths which topologically have a substantial overlap. The return of testing the second path after testing the first path should be reduced, and is not the same as that by testing of the second path alone. This type of path correlations should be included in the statistical analysis for path selection.

It is also important to note that for a defect that falls beyond the topological coverage of a selected path set, this defect has no chance of being detected. Due to this reason, it seems that the selection of critical path also needs to consider path coverage. Then, it is unclear what will be the best way to simultaneously incorporate the path selection objectives from both path correlation and path coverage and at the same time, not to sacrifice the original objective of selecting the statistically timing critical paths.

Without a formal definition of the problem, it is hard to formally capture the concepts of timing critical path, path correlation, and path coverage and hence, hard to effectively incorporate all these objectives into path selection. Therefore, in the following we start with a formal definition of the path selection problem.

3. PROBLEM FORMULATION

In this section, we define the path selection problem. A circuit C is a graph with 5-tupe (V,E,I,O,f), where V is a set of vertices, E is a set of arcs, I,O are two subsets of V with $I \cap O = \phi$, and f is a function on E where $\forall e_i \in E$, $f(e_i)$ is a random variable defined over $[0,+\infty]$.

A path p on C is defined as a path starting from a vertex in I and ending with a vertex in O. Let $p = \{e_1,\ldots,e_i\}$. The *timing length* of p, denoted as $TL(p)$ is a random variable characterized by the joint distribution $Sum = f(e_1)+\cdots+f(e_i)$. For each vertex $o_i \in O$, the *arrival time* denoted as $Ar(o_i)$ is a random variable characterized by the joint distribution $Max = \max\{TL(p_1),\ldots,TL(p_j)\}$ where each p_l, $1 \le l \le j$, ends at o_i. The *circuit delay* of C is defined as a random variable characterized by the distribution $\Delta(C) = \max\{Ar(o_1),\ldots,Ar(o_k)\}$, where $k = \mid O \mid$.

Given path set P, the induced circuit of P on C, denoted as $induced(P)$, is a sub-circuit C' where any edge segment not on a path in P is removed from C.

Let D be a *defect distribution function* defined on C. The path selection problem is defined as the following.

DEFINITION 1. *(Problem Definition) Given a circuit C, a defect distribution D, a clock period clk, and an integer k, find a set of k paths s.t. the following conditional probability is minimized:*

$$Defect\ Miss = \wp_{miss} =$$
$$Prob(\Delta(D(C)) > clk \mid \Delta(D(Induced(P)) \le clk)$$

where $D(circuit)$ produces a new circuit delay distribution in the way as described below.

3.1 Defect Distribution

The above optimization problem is not well-defined unless a specific D is given. Since D essentially alters the circuit delay of C, by the definition of circuit delay, there are two ways to define D: *segment oriented* ($D(e_i)$) and *path oriented* ($D(p_i)$). In this paper, we consider only the segment-oriented defect definition.

DEFINITION 2. *(Segment Oriented) D is a function defined on E, where $D(e_i) = (\delta_i,\gamma_i)$, γ_i is a random variable characterizing the probability of a defect occurrence on e_i, and δ_i is a random variable characterizing the delay defect size.*

Usually, we can assume that δ_i and γ_i are independent. For simplicity, we can further assume that δ_i and δ_j are independent for $i \neq j$. Similarly, we assume γ_i and γ_j are independent.

EXAMPLE 1. *With single-site uniform delay defect assumption, $Prob(\gamma_1 = 1) = \cdots = Prob(\gamma_m = 1) = \frac{1}{m}$ where $m = \mid E \mid$.*

4. OPTIMIZATION OBJECTIVES

Given the defect distribution in Definition 2, to minimize $\wp_{miss}$, we will re-formulate the problem slightly. In essence, to minimize $\wp_{miss}$, it is the same as to maximize $\wp_{capture}$ where $\wp_{capture}$ is defined as the following (Let A denote the event "defects fall on P," and B denote the event "$\geq$ 1 defect are captured by testing $Induced(P)$):"

$$\wp_{capture} = \wp_{obj2} * \wp_{obj1} \tag{1}$$

$$\wp_{obj1} = Prob(\mathrm{A}) \tag{2}$$

$$\wp_{obj2} = Prob(\mathrm{B}|\mathrm{A}) \tag{3}$$

Then, we can have the following theorem.

THEOREM 1. *$\wp_{capture} = 1 - \wp_{miss} - \wp_{no}$ where $\wp_{no}$ is the probability of all defects having no faulty effect on circuit timing.*

Proof Sketch. It suffices to show that the event spaces defined in the three probabilities $\wp_{capture}, \wp_{miss}, \wp_{no}$ are disjoint. Since the event spaces defined in the three probabilities are disjoint (and they form the total space), the theorem holds. □

Note that $\wp_{no}$ depends only on the circuit C and the defect function D, and is independent of P. Therefore, to minimize $\wp_{miss}$, by Theorem 1, is equivalent to maximize $\wp_{capture}$.

Given a defect distribution function D, by assuming that $\forall i, j$, δ_i is independent of γ_j, we can remove the conditional event in equation (3) and obtain a simpler equation for $\wp_{obj2}$.

$$\wp_{obj2} = Prob(B) \tag{4}$$

This is because defect occurrences and locations are independent of defect sizes. Hence, we can remove the conditional event A = "defects fall on P."

4.1 Maximizing $\wp_{obj1} * \wp_{obj2}$

Given a circuit C, a defect function D, and a path set P, $\wp_{obj1}$ and $\wp_{obj2}$ can be calculated independently if $\forall i, j$, δ_i is independent of γ_j. However, it is important to note that to maximize $\wp_{capture}$, it is not sufficient to maximize $\wp_{obj1}$ and $\wp_{obj2}$ independently. This is because these two objectives can be opposite to each other during the selection of P. In other words, the optimal P to maximize $\wp_{obj1}$ may not be the optimal P to maximize $\wp_{obj2}$. However, we also note that if this can be done with the same set of P, then obviously that particular P also maximizes $\wp_{capture}$ as well.

Without knowing that if a single optimal P set exists for both $\wp_{obj1}$ and $\wp_{obj2}$ and hence, for $\wp_{capture}$, we consider the following three questions.

1. Independently, how to optimize $\wp_{obj1}$?
2. Independently, how to optimize $\wp_{obj2}$?
3. Together, how can we combine the optimal algorithm for question 1 and the optimal algorithm for question 2 without losing much in each algorithm for what it tries to optimize individually?

5. OPTIMIZING $\wp_{OBJ1}$

Given a path set P and a segment oriented defect function D, let $\{e_1,\ldots,e_k\}$ be the set of segments covered by P. Then, we have

$$\wp_{obj1} = Prob(\text{defects fall on P}) = Prob(\gamma_1,\cdots,\gamma_k) \tag{5}$$

where $Prob(\gamma_1, \cdots, \gamma_k)$ is the joint probability distribution of all random variables $\gamma_1, \ldots, \gamma_k$. If $\gamma_1, \ldots, \gamma_k$ are mutually independent, then we have

$$\wp_{obj1} \propto \frac{\sum_{\forall e_i \in P} Prob(\gamma_i)}{\sum_{\forall e_i \in C} Prob(\gamma_i)} = \frac{\sum_{\forall e_i \in P} Prob(\gamma_i)}{\text{some constant}} \quad (6)$$

To maximize $\wp_{obj1}$ in equation (6), we will focus the discussion on the following optimization problem that is essentially the same.

DEFINITION 3. *(Maximum Path Cover (MPC)) Given a circuit graph $G = (V, E, I, O)$, a weight assignment function W defined on E such that $W(e_i) = w_i$, and an integer k, the problem is to find a path set of size k in order to maximize:*

$$(\textstyle\sum_{e_i \in P} w_i), \text{ where } \mid P \mid = k$$

In the following, we will show that the MPC problem is practically intractable. Then, we will show several heuristics to solve the problem and discuss their performance.

5.1 Intractability of The MPC Problem

One problem related to MPC is the Minimum Vertex Cover (Min-VC) problem discussed in [12]. Given an undirected graph $G = (V, E)$, the Min-VC is to find the minimum set of vertices that cover all edges. It is shown in the paper that Min-VC is a problem in the MAX-SNP class, where finding an $(1+\varepsilon)$ polynomial time approximation algorithm is NP-hard [12]. That is, if the optimal size of the vertex cover to the problem is OPT, it is NP-hard to guarantee a vertex cover with a size $\leq (1+\varepsilon)OPT$ for some $0 < \varepsilon \leq 1$.

There is a slightly different version of the Min-VC problem called the Maximum Vector Cover (Max-VC). The problem is that, given an integer k, find a set of k vertices that cover the maximum number of edges. Petrank [13] shows that it is also NP-hard to find a $(1-\varepsilon)$-approximation algorithm for the Max-VC problem.

The generalized version of the Max-VC problem is called the Maximum Coverage (Max-C) Problem. Given a set $I = \{1, \ldots, n\}$. Let J denote the indices of all non-empty subsets of I, and S_j denote the jth subset with index $j \in J$. Given a set $F = \{S_i | i \in J\}$, a non-negative weight w_i for each S_i, and a positive integer p, the problem is to find a subset $X \subseteq I$ with $\mid X \mid = p$ such that the total weight of all S_k which have nonempty overlap with X is maximized.

Let $d = \max\{\mid S_j \mid : j \in J\}$. The Max-C problem is a generalized version of the Max-VC problem because we can reduce the Max-C problem to the Max-VC problem by setting $d = 2$ and all $w_i = 1$. If we allow any weight assignments, but keep the constraint $d = 2$, the Max-C problem is reduced to the weighted version of the Max-VC problem (WMVC). For $d > 2$, it is the same as solving the WMVC problem on a hypergraph.

LEMMA 1. *MPC problem is intractable.*

Proof. To demonstrate that MPC is intractable, it suffices to develop a polynomial time reduction scheme from the WMVC problem to the MPC problem. Here we re-state the WMVC problem. Given an undirected graph $G = (V, E)$, a weight assignment $W(e_i) = w_i$ for all $e_i \in E$, and a positive integer p, find a vertex cover $X \subseteq V$, $\mid X \mid = p$, of which the total covered weight is the maximum. Given a problem instance in WMVC, we will reduce it into an MPC problem instance using the following polynomial time algorithm.

1. Create two nodes s and t.
2. Order all edges as $e_1, \ldots, e_m$ where $\mid E \mid = m$.
3. Pick a vertex $v_j \in V$, create a path p_j where p_j starts from s and ends at t, and contains all ordered edges $\{e_{j_1}, \ldots, e_{j_k}\}$ ending at v_j. The following "pseudo edges" with weight assignments equal to some fixed small number close to zero are added to connect $\{e_{j_1}, \ldots, e_{j_k}\}$ in order to form a path.
   - Add a pseudo edge from s to e_{j_1}.
   - Add a pseudo edge from e_{j_k} to t.
   - For any adjacent edges $e_{j_l}, e_{j_{l+1}}$, if $j_{l+1} - j_l > 1$, add a pseudo edge from e_{j_l} to $e_{j_{l+1}}$.
4. $V = V - \{v_j\}$, if V is empty, stop; Otherwise, goto step 3.

It is obvious that the above reduction is an $O(m|V|)$ algorithm. By keeping the weight assignment for each original edge, the MPC problem is to find p paths that cover the maximum total weight. If we ensure that the total weight given by all pseudo edges is far less than the minimum edge weight assigned in the original problem instance, then those pseudo edges will have no impact on the total weight calculation for the optimal solution.

Let $T(P)$ denote the total weight covered by a solution P in MPC and $T(X)$ denote the total weight covered by a solution X in WMVC. Then, for any two solutions P_1, P_2 in MPC, where $|P_1| = |P_2| = p$, there exists two corresponding solutions X_1, X_2 in WMVC (just map a path back to its corresponding vertex) such that $T(P_1) < T(P_2) \Leftrightarrow T(X_1) < T(X_2)$. The same ordering in the solution spaces of the WMVC and WPC instances implies that if the optimal solution is unique, it is the same in both instance. Moreover, given an ε, $0 < \varepsilon \leq 1$, if there exists a polynomial time $(1-\varepsilon)$-approximation algorithm for MPC, then it implies that there exists a polynomial time $(1-\varepsilon)$-approximation algorithm for WMVC. Hence, the MPC problem is intractable. □

5.2 Heuristics to Approximate MPC

In this section, we will discuss heuristics to solve the MPC problem. Most of these heuristics have been analyzed for the Max-C problem. Therefore, to facilitate the discussion, we will first show a polynomial time reduction from MPC to Max-C.

LEMMA 2. *Given that the total path population under consideration is of polynomial size in terms of $|E|$, MPC is polynomial-time reducible to Max-C such that if there exists a polynomial time $(1-\varepsilon)$-approximation algorithm for Max-C, the algorithm is a $(1-\varepsilon)$-approximation for MPC.*

Proof. The mapping from MPC to Max-C is natural. Let $P = \{p_1, \ldots, p_n\}$ be the path set. We simply let $I = \{1, \ldots, n\}$ in the Max-C. Each S_i in the Max-C problem corresponds to an edge e_i. Hence, the weight of each e_i (w_i) is also the weight for S_i. We also have $j \in S_i$ if p_j contains the edge e_i. Essentially, the MPC problem is the same as the WMVC problem on a hypergraph. □

The above reduction further implies that if there exists a heuristic that guarantees a lower bound approximation ratio for the solution to MAX-C, then the heuristic can also guarantee the same lower bound approximation ratio for the MPC problem.

5.2.1 *Linear Program Relaxation Heuristic*

Authors in [15] utilizes Linear Program Relaxation (LPR) to solve the Max-C problem. They demonstrate that LPR heuristic is a $[1-(1-\frac{1}{d})^d]$-approximation algorithm, where $d = \max\{\mid S_j \mid : j \in J\}$ as stated before. For WMVC, $d = 2$ and hence, LPR heuristic is a $\frac{3}{4}$-approximation algorithm. With Lemma 2, the LPR heuristic is also a $[1-(1-\frac{1}{l})^l]$-approximation algorithm for the MPC problem, where l is the maximum number of paths which share the same edge segment in the circuit.

The LPR heuristic requires solving LP problem for maximizing $\wp_{obj1}$ alone. If we adopt this heuristic, it is hard to see how to combine with any other heuristic(s) used to maximize $\wp_{obj2}$ later. Since our final goal is to maximize $\wp_{capture}$, not merely $\wp_{obj1}$, the LPR heuristic does not seem to be a suitable heuristic for us even though it is the best known approximation algorithm for the Max-C problem. For the reason just mentioned, in the following we will turn our attention to the simpler "greedy" heuristics.

5.2.2 First Greedy Heuristic

Our first greedy heuristic is a typical and widely-used one in many optimization applications.

HEURISTIC 1. *In each step, select the path that results in maximum additional weight coverage.*

THEOREM 2. *The greedy heuristic in Heuristic 1 is a $[1-(1-\frac{1}{k})^k]$-approximation algorithm for MPC problem, where k is the number of paths allowed in the problem.*

Proof. It is well known as shown in [14] that the same greedy heuristic is a $[1-(1-\frac{1}{p})^p]$-approximation algorithm for the Max-C problem, where p is the number of vertices allowed in the problem. Hence, by Lemma 2, the theorem holds. □

As k becomes large, the greedy heuristic approaches to $(1-\frac{1}{e})$-approximation, where e is the natural number.

5.2.3 Second Greedy Heuristic

HEURISTIC 2. *Sort all paths according to their total weights covered. Select the largest k paths.*

Let L be the number of total paths in MPC (vertices in WMVC). Authors in [16] shows that the above heuristic for WMVC problem is a $\frac{k}{L}$-approximation algorithm. However, the same argument used in [16] does not hold for WMVC on hypergraph. This is because an edge on a hypergraph can connect more than two vertices.

Actually, one can construct an MPC instance to make the performance of Heuristic 2 as bad as possible. In fact, let $p_1,\ldots,p_n$ as the sorted paths with total covered weights $t_1 \geq \cdots \geq t_n$. It is easy to construct an instance to "fool" the heuristic by making the first k paths $p_1,\ldots,p_k$ exactly the same except for the last edge segment. And for each last edge, we associate a very samll weight ε. On the other hand, for all $p_{k+1},\ldots,p_n$, we make them all independent with each total covered weight all equal to "the weight of $p_1 - \varepsilon$." It is easy to see that the only bound we can have by heuristic 2 is then, $\frac{1}{k}$. That is, the heuristic guarantees the selection of the first maximum-weight path, but nothing more.

THEOREM 3. *The Heuristic 2 is a $\frac{1}{k}$-approximation algorithm for MPC problem (In the practical sense, it is unbounded).*

The best know heuristic is the $[1-(1-\frac{1}{p})^p]$-approximation by LPR, and the simple heuristic above is in essence an unbounded algorithm. That is, as k becomes sufficiently large, this simple heuristic can perform poorly.

5.2.4 Third Greedy Heuristic

HEURISTIC 3. *Sort all edges according to their weights. In each step, select a path that covers an uncovered edge whose weight is the maximum.*

Let $m = \mid E \mid$. The following theorem is straightforward.

THEOREM 4. *The Heuristic 3 is a $\frac{k}{m}$-approximation algorithm for MPC problem.*

Proof. Let the sorted edge weights be $w_1 \geq \cdots \geq w_m$. Let *Sol* be the solution weight given by the heuristic. It is clear that $OPT \leq \sum_{1\leq i\leq m} w_i$. Also, $Sol \geq \frac{k}{m}(\sum_{1\leq i\leq m} w_i)$ because Sol contains the largest-weighted k edges. Hence, the theorem holds. □

Since usually, we expect that $m \ll L$, this heuristic provides a much better bound than the second heuristic.

6. OPTIMIZING $\wp_{OBJ2}$

In this section, we discuss the optimization of $\wp_{obj2}$ given in equation (4) before.

Given a circuit $G = (V,E,I,O,f)$, we first consider a simplified case where $f = f_{fixed}(e_i) = c_i$ for some fixed constant c_i. This intends to model the fixed-delay assumption commonly used in delay test and timing analysis.

LEMMA 3. *In the fixed-delay circuit model, the optimal solution path set P for maximizing $\wp_{obj2}$ is to select the k longest paths.*

Proof. It can be observed that if P consists of the k longest paths, then for any edge in the induced circuit $Induced(P)$, the longest path that covers the edge in G is always included in $Induced(P)$. Hence, $\wp_{obj2}$ is maximized for all the edges in $Induced(P)$. □

Next, we consider the case for $f = f_{random}(e_i) = a_i$ where a_i is a random variable characterizing the delay on edge e_i. Given a path $p_j = \{e_1,\ldots,e_i\}$. Let $A_j = a_1 + \cdots + a_i$. We further define the *critical probability* of p_j as $CRT(p_j) = crt_j = Prob(A_j > clk)$ for a given constant clk. Since crt_j is a real number between 0 and 1, we can use these numbers to rank all paths.

Given two paths p_i, p_j with initial critical probabilities crt_i, crt_j, respectively, the $Prob(A_i > clk | A_j \leq clk)$ may not be the same as $Prob(A_i > clk)$. In fact, $Prob(A_i > clk|A_j \leq clk) \leq crt_i = crt_i - Cor(A_i,A_j)$, where $Cor(A_i,A_j)$ characterizes the *correlation factor* (or correlation probability) between paths p_i, p_j. If p_i and p_j are topologically overlapping, then the correlation factor is nonzero. We observe that the correlation factor is symmetric. In other words, $Cor(A_i,A_j)$=$Cor(A_j,A_i)$

Suppose that we rank all paths $p_1,\ldots,p_L$ according to their critical probabilities $crt_1 \geq \cdots \geq crt_L$. If we select $P_k = \{p_1,\ldots,p_k\}$, will it give us the optimal results for maximizing $\wp_{obj2}$ as that in the case of the fixed-delay model? The following lemma provides an answer.

LEMMA 4. *P_k can be unbounded for optimizing $\wp_{obj2}$.*

To see why this is true, we first define a much more complicated version of the WMVC problem.

DEFINITION 4. *Given an undirected graph $G = (V,E)$, the Dynamic Weighted Maximum Vertex Cover (D-WMVC) problem is an instance of a 5-tuple (V,E,W,Cor,CRT). CRT is a weight function associated with each vertex v_i and $CRT(v_i) = crt_i$ for $0 \leq crt_i \leq 1$. Cor is an update function on the weights. For any pair of vertices v_i, v_j, Cor is a function of (V',G,CRT) where V' is the set of vertices currently selected into the cover set. We have initially $Cor(v_i,v_j) = -c_{ij}$.*

The rule is that every time a vertex v_i is selected, we will count its weight as "covered," and at the same time all the weights associated with its adjacent vertices will be updated by the update function. After the update, the update function Cor itself will change accordingly (and suppose this change is polynomial time computable). Hence, the weight configuration is changed dynamically. Then, the D-WMVC problem is to select k vertices such that the total resulting weight is maximized.

It is easy to see that there is a natural mapping between the D-WMVC problem to the optimization problem for $\wp_{obj2}$. The weight function *CRT* in a D-WMVC instance corresponds to the critical probability function. The update function *Cov* corresponds to the correlation factor function. Since the update of *Cov* is defined dynamically based upon the current path selection, the D-WMVC problem is obviously a much harder problem than the original WMVC problem.

Given a D-WMVC problem instance, what is the P_k (the set of paths with the k largest critical probabilities) trying to accomplish? The answer is that P_k provides a greedy heuristic similar to Heuristic 2 described before for the MPC and Max-C problems. Therefore, it is not a good heuristic.

HEURISTIC 4. *(Greedy by Considering Path Correlation) Each time, select the path with the largest critical probability. After the selection, apply the correlation function Cov to update the critical probabilities for all unselected and correlated paths.*

It can be easily seen that Heuristic 4 is a version of Heuristic 1 in the context of the D-WMVC problem. Unfortunately, the $(1-\frac{1}{e})$ approximation bound cannot be guaranteed with Heuristic 4 for D-WMVC unless *Cov* becomes a static function. If this is the case, then we can reduce the D-WMVC problem into the WMVC problem.

What does a static *Cov* function mean? It means that in the circuit instance, no path is correlated to more than one other path. In this case, we only need to consider all pair-wise correlation factors and hence, the D-WMVC problem is the same as the WMVC problem.

THEOREM 5. *Given a circuit instance where no path is correlated with more than one other path, Heuristic 4 is a $(1-\frac{1}{e})$-approximation algorithm for the the $\wp_{obj2}$ optimization problem.*

Proof. To maximize $\wp_{obj2}$, it is the same as to maximize the total resulting weight (or total resulting critical probabilities) in the D-WMVC problem. Hence, the theorem holds. $\square$

Let $Cov_2(A_i, A_j)$ characterizes the critical probabilities shared by both A_i and A_j. Following a similar concept, we can define $Cov_l(A_{i_1}, \ldots, A_{i_l})$ as the critical probabilities shared by l random variables $A_{i_1}, \ldots, A_{i_l}$. Then, it is not hard to see that in the static definition of D-WMVC above, we are trying to use Cov_2 to capture all Cov_q for $2 < q \leq k$ where each Cov_q is defined on all possible q correlated paths, and k is the given path size in the problem.

DEFINITION 5. *(Residue Correlation Factor, RCF) Define $RCF = \sum_{3 \leq q \leq k, \forall crt_i} Cov_q[]$.*

RCF is the summation of all critical probabilities simultaneously shared by more than two paths.

DEFINITION 6. *(Static Instance of D-WMVC) Given a D-WMVC problem instance, define the static version of the instance as the one by replacing the dynamic update function Cov with the static function Cov_2.*

THEOREM 6. *Let Sol be the total critical probability output by using Heuristic 4 on the static version of the D-WMVC problem. Let OPT be the true optimal value. We have $(1-\frac{1}{e})(OPT - RCF) \leq Sol$.*

Proof. Let OPT' be the optimal value for the static version of the D-WMVC problem instance. We have $(1-\frac{1}{e})OPT' \leq Sol$ by Theorem 5. Observe that $OPT \leq OPT' + RCF$ because *RCF* is the upper bound of how much we may miss during the calculation of the critical probabilities. Hence, $OPT \leq (\frac{e}{e-1})Sol + RCF$ and the theorem holds. $\square$

7. HEURISTICS TO OPTIMIZE $\wp_{CAPTURE}$

Recall that $\wp_{capture} = \wp_{obj1} * \wp_{obj2}$. In the previous sections, we discuss heuristics to maximize $\wp_{obj1}$ and $\wp_{obj2}$ individually. Based upon those results, in this section we discuss three heuristics to maximize $\wp_{capture}$.

H-Timing Traditionally, the most natural way is to select the k longest paths. Under a fixed-delay model, this heuristic optimize $\wp_{obj2}$ (Lemma 4) but has little guarantee for $\wp_{obj1}$. With a probabilistic delay model, this heuristic (select the largest k critical probabilities) is similar to Heuristic 2. Hence, it offers little guarantee for optimizing either $\wp_{obj1}$ or $\wp_{obj2}$. From this perspective, H-Timing is not a good heuristic.

H-Segment In this heuristic, optimizing $\wp_{obj1}$ has a higher priority than $\wp_{obj2}$. Given a circuit instance $G = (V, E, I, O, f)$ and a defect function $D(e_i) = (\delta_i, \gamma_i)$, at each step we select a path to maximize the total uncovered probability from $Prob(\gamma_i = 1)$. If there are multiple such paths, we then select the one with the longest timing length (or the largest critical probability).

H-Segment follows the Heuristic 1 above and hence, is an $(1-\frac{1}{e})$-approximation algorithm for maximizing $\wp_{obj1}$. However, it has no guarantee for optimizing $\wp_{obj2}$. Therefore, the performance can be unsatisfactory.

H-Opt Let $w_1 \geq \cdots \geq w_m$ correspond to the defect probabilities $Prob(\gamma_1 = 1), \ldots, Prob(\gamma_m = 1)$, respectively, At each step, we select a minimal j such that w_j is not yet covered. Then, use Heuristic 4 to select the largest critical probability of an unselected path p_i such that $e_j \in p_i$.

The H-Opt uses Heuristic 3 for maximizing $\wp_{obj1}$ and hence, is a $\frac{k}{m}$-approximation algorithm for optimizing $\wp_{obj1}$. Unfortunately, ensuring the coverage of the largest uncovered w_j may prevent H-Opt to behave exactly the same as Heuristic 4 above. However, if we consider that all edge segments have an almost equal probability of receiving a defect. Then, H-Opt will behave like Heuristic 4 with only one potential exception: Heuristic 4 may select a path whose edges are already covered by at least one path selected before. We discuss this issue below assuming that defect occurrence probabilities are uniform.

LEMMA 5. *Let $P = \{p_1, \ldots, p_i\}$ as the ordered paths selected by Heuristic 4. Let $C' = Induced(P)$. For any $p' \in C'$ and $p' \notin P$, the $Prob(TL(p') > clk | \forall p \in P, TL(p) \leq clk) = 0$.*

Proof. This is because if we make sure that all the long paths are shorter than the *clk*, it is impossible to have a short path whose timing is greater than *clk*. If $TL(p') > clk$ after testing all paths in P, then there exists a $j, 1 < j < i$ such that after testing $\{p_1, \ldots, p_j\}$ (testing is in that order), the conditional $Prob(TL(p') > clk)$ is greater than the conditional $Prob(TL(p_{j+1}) > clk)$. However, this implies that p' should be selected into P by the Heuristic 4 (instead of p_{j+1}) and hence, is not possible. $\square$

COROLLARY 1. *Let $P = \{p_1, \ldots, p_i\}$ as the ordered paths selected by Heuristic 4 after step i. Then, There exists a segment edge e such that $e \in p_i$ and $e \notin Induced(P - \{p_i\})$.*

The above corollary is implied by the Lemma 5. This corollary says that by using Heuristic 4 for optimizing $\wp_{obj2}$, it can also ensure a result for $\wp_{obj1}$, which is no worse than that given by applying Heuristic 3 to maximize $\wp_{obj1}$. Hence, Heuristic H-Opt is the only one that can simultaneously try to optimize both $\wp_{obj1}$ and $\wp_{obj2}$. With this corollary, we state the main theorem in our paper.

THEOREM 7. **(Main Theorem)** *Suppose that the optimal value of $\wp_{capture} = OPT1 * OPT2$. H-Opt computes a solution value Sol for maximizing $\wp_{capture}$. Then, we have $(\frac{k}{m})OPT1(1-\frac{1}{e})(OPT2 - RCF) \leq Sol$, given that defect occurrence distribution is uniform.*

To validate the theoretical results discussed in the previous three sections, in the following we describe a framework for conducting practical experiments under the statistical delay and defect occurrence assumption.

7.1 Compute Correlation Factor

The statistical method described in [11] provides a practical approach to calculate the critical probability for a given path. In order to implement Heuristic H-Opt, we also need a method to compute the correlation probabilities.

The overall scheme in Heuristic H-Opt consists of two steps: 1) Select the statistically longest path based upon the current delay distributions and ensure that it covers one additional edge segment, and 2) Re-construct delay distributions to reflect path correlation resulted from the selection.

For the re-construction of delay distribution after i paths are selected, $\forall i, 1 \leq i \leq k$, a cut-off period T is assumed. We use a Monte

Carlo sampling approach as described below. Suppose path A is selected, and consists of a sequence of signal segments whose delays are characterized by random variables $s_1 \ldots s_n$. The path delay of A can be characterized as the joint pdf $J(s_1 \ldots s_n)$. After the selection of path A, we re-construct all pdf's of $s_1 \ldots s_n$ based upon sampled circuit instances whose delays on path A are all $\leq T$. Now suppose another path B overlaps with A by consisting of $s_i \ldots s_j$. Since the distributions of $s_i \ldots s_j$ have changed, the joint pdf distribution of B will be re-calculated accordingly.

7.2 Universal Path Candidate Set

One key assumption during the discussion is that the number of paths being considered during the path selection is $O(m)$, where $m = |E|$ in the circuit graph. Without pre-processing, this is an unrealistic assumption because a circuit can easily have an exponential number of paths. In this section, we discuss a simple path selection scheme as a pre-processing step in the path selection optimization process. During this pre-processing step, the goal is to quickly cut down the size of total path population.

In our methodology, we will construct the *universal path candidate set* (U). The size of U is much smaller than the number of all paths and hence, coverage of U can be calculated much faster. We further ensure that by covering U, the actual circuit performance can be guaranteed with a very high probability. Then, the U set will serve as the base point for later path selection optimization.

If in our statistical framework a path has a very low probability of being a "long path" then in reality it is unlikely that a small delay defect or variation on the path will cause a timing problem. With this idea in mind, construction of U are based on two given parameters: a test clock C and a cutoff period T where $T \leq C$. The U consists of every path whose probability of being a path longer than T is non-zero. In other words, if all paths in U are covered, then with a very high probability, any faulty behavior resulted from delay defect and variation of a delay size smaller than $\Delta = C - T$ will be captured [11]. After an initial U set is established, we can further prune the size of U by removing those functionally unsensitizable paths using the new methodology developed in [11].

8. EXPERIMENTAL RESULTS

8.1 Experimental setup

Our experimental flow consists of three major phases, timing analysis, path selection, and evaluation as described below.

I. Timing Analysis Phase

An efficient cell-based false-path-aware statistical timing analysis framework was developed in [11]. It requires pre-characterization of cells, i.e., building libraries of pin-pin cell delays and output transition times (as random variables). In our experiments, we utilizes a Monte-Carlo-based SPICE (ELDO) [18] to extract the statistical delays of cells for a 0.25μm, 2.5V CMOS technology.

II. Path Selection Phase

The first step in path selection is to produce the universal path candidate set U ([11]). Then, we apply each of the three heuristics (**H-Timing, H-Segment, and H-Opt** described in section 7) to derive an optimal path set S where $|S| = k$.

III. Evaluation Phase

In our study, we estimate the quality of selected paths in terms of the miss probabilities defined in Definition 1 at the beginning of section 3. This estimation is calculated based upon paths alone, instead of the quality of tests generated based upon those paths [17]. Hence, our metric involves only static analysis and is pattern independent. Most importantly, our metric is based upon the statistical delay evaluation framework which utilizes a Monte-Carlo-based approach to actually simulate a large sample of a given design. In our experiments, 10,000 circuit instances were analyzed.

We illustrate the complete procedure of the evaluation scheme as the following. In each Monte Carlo sampling run, first a circuit instance is generated according to the cell/interconnect delay distributions characterized through Monte Carlo SPICE. Also random defects can be injected for each circuit instance (on any locations). This instance will then be evaluated by two analysis steps: "statistical analysis of S" and "statistical analysis of U-S". The "statistical analysis of S" is to check if there is any path in S (on the given instance) longer than the testing clock C. If there is, then this instance is said to be faulty and covered by S (*Covered*). The "statistical analysis of U-S" performs a similar analysis on the set of $U - S$ and reports the number of faulty instances not covered by S (*Noncovered*). At the end, our scheme will calculate the probability of a faulty path captured by S based upon all the instances statistically produced. This *conditional missing probability* is defined as

$$\wp_{miss} = \frac{Noncovered}{Covered + Noncovered}$$

In other words, the conditional missing probability $\wp_{miss}$ is the probability that a delay defect is not covered by S given that the delay defect will affect the circuit performance.

Defect Distribution In the experiments, the evaluations are based on the assumption of a defect size distribution: $\lambda e^{-\lambda x}$ where x is the defect size and λ is a constant We use λ=0.1 and 0.04 in the experiments. This exponential distribution for defect size (given that defects occur) has been studied in many publications [19, 20] and is a practical assumption to be used. Note that it is also possible to adopt other distributions. However, using other distributions in general does not invalidate the trends observed in our work.

8.2 Results

we will focus on the results from circuit s5378 for detailed discussion. Other results are available but due to space limitation, they are not included. We note that all results we have so far are consistent with the theoretical findings.

The benchmark s5378 has an important characteristic: the path delay profile for s5378 indicates that the performance of the circuit is not dominated by a few paths (more equally distributed). Figure 1 demonstrates the path profile of the path universe U, where $|U| = 1328$.

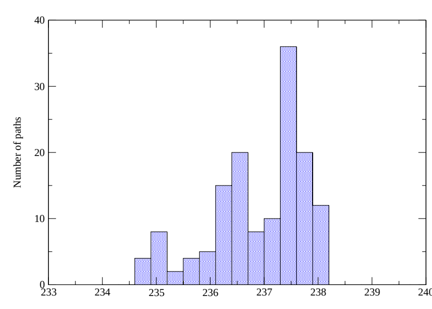

Figure 1: The profile of path delays for s5378.

The following plots show the evaluation results for different heuristics. These plots demonstrate the trends of missing probabilities versus the number paths. Results in Figure 2 are based on the defect distribution of $e^{-0.1x}$. For comparison, we also derive results for the defect distribution of $e^{-0.04x}$ in Figure 3. Random delay samples from $e^{-0.1x}$ range roughly from 0 to 40, while those from $e^{-0.04x}$ will extend to 100. By applying these two different defect models, we can show how larger defects affect the results of different heuristics.

As we observe in these results, **H-Opt** consistently outperforms all other heuristics as predicted by the theoretical analysis (with smaller missing probabilities). More interesting observations can be made when the model of $e^{-0.04x}$ is used. Since the range of defect size spreads out, more edges have to be covered to maintain a low missing probability for a fixed number of k paths. As shown in the figures, the **H-Opt** still converges quickly as the number of paths increases. As stated before, **H-Timing** provides no guarantee at all and hence, clearly performs even worse in Figure 3. For large-size defects, **H-Segment**, which is optimized for covering more segments, can have a similar level of coverage as **H-Opt** (while the number of selected paths is small).

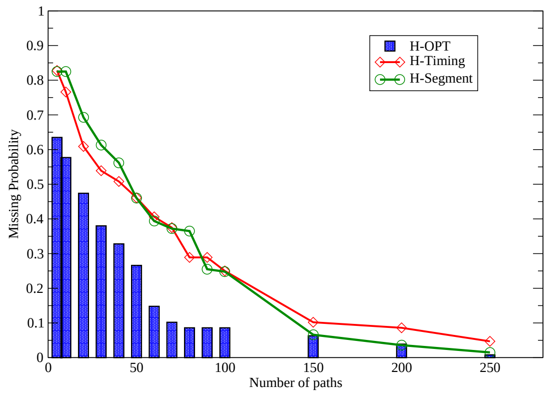

Figure 2: Comparing heuristics in statistical domain using a defect model of $e^{-0.1x}$.

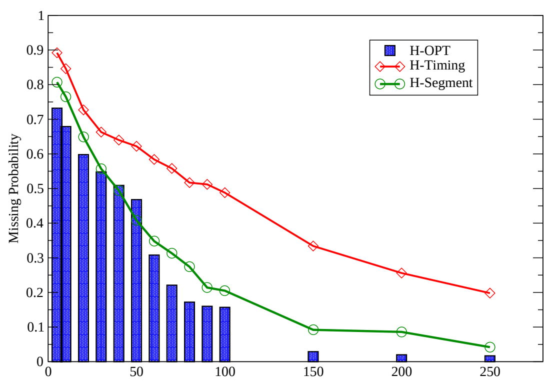

Figure 3: Comparing heuristics in statistical domain using a defect model of $e^{-0.04x}$.

9. CONCLUSION

In this paper, we formalize the problem of critical path selection as a new optimization problem that consists of two theoretically intractable sub-problems. We provide theoretical analysis for various heuristics used to solve each sub-problem individually. Then, we prove that the H-Opt heuristic is theoretical feasible and practical. We show that a seemingly intuitive heuristic H-timing can actually be the worst. To validate our findings, we develop an experimental scheme based upon statistical timing analysis framework and defect-injected simulation. Our experimental results confirm that H-Opt is indeed the best heuristic among all we studied. Our formulation of the path selection can lead to many interesting theoretical developments in the area of delay testing. The statistical timing evaluation framework can provide a general approach to validate and compare future DSM delay fault testing methods.

10. REFERENCES

[1] K. Baker, G. Gronthoud, M. Lousberg, I. Schanstra, , and C. Hawkins. Defect-Based Delay Testing of Resistive Vias-Contacts, A Critical Evaluation. *Proceedings of IEEE International Test Conference*, pages 467–476, September 1999.

[2] M. A. Breuer, C. Gleason, and S. Gupta. New Validation and Test Problems for High Performance Deep Sub-Micron VLSI Circuits. *Tutorial Notes, IEEE VLSI Test Symposium*, April 1997.

[3] Robert C. Aitken, "Nanometer Technology Effects on Fault Models for IC Testing", *IEEE Computer*, November 1999, pp. 46-51

[4] K-T Cheng, S. Dey, M. Rodgers, and K. Roy, *Test Challenges for Deep Sub-Micron Technologies*, ACM/IEEE Design Automation Conference 2000.

[5] W.-N. Li, S. M. Reddy, and S. K. Sahni. On Path Selection in Combinational Logic Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 8(1):56–63, January 1989.

[6] S. Tani, M. Teramoto, T. Fukazawa, and K. Matsuhiro. Efficient Path Selection for Delay Testing Based on Partial Path Evaluation. *Proceedings of IEEE VLSI Test Symposium*, pages 188–193, May 1998.

[7] C. P. Ravikumar, N. Agrawal, and P. Agrawal. Hierarchical Delay Test Generation. *Journal of Electronic Testing: Theory and Applications*, 10(3):188–193, June 1997.

[8] K. Antreich, A. Ganz, and P. Tafertshofer. Statistical Analysis of Delay Faults-Theory and Efficient Computation. *AEU-International Journal of Electronics and Communications*, 51(3):117–130, May 1997.

[9] R. Levy, D. Blaauw, G. Braca, A. Dasgupta, A. Grinshpon, C. Oh, B. Orshav, S. Sirichotiyakul, and V. Zolotov. ClariNet: a noise analysis tool for deep submicron design. *Proceedings of Design Automation Conference*, pages 235–248, June 2000.

[10] J.-J. Liou, K.-T. Cheng, and D. Mukherjee. Path Selection for Delay Testing of Deep Sub-Micron Devices Using Statistical Performance Sensitivity Analysis. *Proceedings of IEEE VLSI Test Symposium*, pages 97–104, April 2000.

[11] J.-J. Liou, A. Krstic, L.-C. Wang, and K.-T. Cheng. False-Path-Aware Statistical Timing Analysis and Efficient Path Selection for Delay Testing and Timing Validation. *Proceedings of Design Automation Conference*, June 2002.

[12] C. H. Papadimitriou, and M. Yannakakis, "Optimization, Approximation, and Complexity Classes," *Journal of Computer and System Sciences*, 43, 1991, pp. 425-440.

[13] E. Petrank, "The hardness of approximation: gap location," *Computational Complexity*, 4, 1994, 133-157.

[14] G. Cornuejols, M. Fisher, and G. L. Nemhauser, "Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms," *Management Science*, Vol 23, No 8, April, 1977, pp. 789-810.

[15] A. Ageev, and M. Sviridenko, "Approximation algorithms for maximum coverage and max cut with given sizes of parts," *IPCO*, G. Cornuejols, R. Burkard, G. Woeginger (Eds), 1999, 17-30.

[16] Q. Han, Y. Ye, H. Zhang, and J. Zhang, "On Approximation of Max-Vertex-Cover," in *European Journal of Operation Research*, 2001.

[17] M. Sivaraman and A. Strojwas. Path Delay Fault Diagnosis and CoverageA Metric and an Estimation Technique. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(3):440–457, March 2001.

[18] Anacad. Eldo v4.4.x User's Manual. 1996.

[19] N. N. Tendolkar. Analysis of Timing Failures Due to Random AC defects in VLSI moduels. *Proceedings of Design Automation Conference*, pages 709–714, June 1985.

[20] J. P. de Gyvez. *Integrated Circuits Defect-Sensitivity: Theory and Computational Models*. Kluwer Academic Publishers, Boston, MA, 1993.