An Exact Algorithm for the Statistical Shortest Path Problem *

Liang Deng
Dept. of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
ldeng@uiuc.edu

Martin D. F. Wong
Dept. of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
mdfwong@uiuc.edu

ABSTRACT

Graph algorithms are widely used in VLSI CAD. Traditional graph algorithms can handle graphs with deterministic edge weights. As VLSI technology continues to scale into nanometer designs, we need to use probability distributions for edge weights in order to model uncertainty due to parameter variations. In this paper, we consider the statistical shortest path (SSP) problem. Given a graph G, the edge weights of G are random variables. For each path P in G, let L_P be its length, which is the sum of all edge weights on P. Clearly L_P is a random variable and we let μ_P and σ_P^2 be its mean and variance, respectively. In the SSP problem, our goal is to find a path P connecting two given vertices to minimize the cost function $\mu_P + \Phi(\sigma_P^2)$ where Φ is an arbitrary function. (For example, if $\Phi(x) = 3\sqrt{x}$, the cost function is $\mu_P + 3\sigma_P$.) To minimize uncertainty in the final result, it is meaningful to look for paths with bounded variance, i.e., $\sigma_P^2 \leq B$ for a given fixed bound B. In this paper, we present an exact algorithm to solve the SSP problem in $O(B(V+E))$ time where V and E are the numbers of vertices and edges, respectively, in G. Our algorithm is superior to previous algorithms for SSP problem because we can handle: 1) *general graphs* (unlike previous works applicable only to directed acyclic graphs), 2) *arbitrary edge-weight distributions* (unlike previous algorithms designed only for specific distributions such as Gaussian), and 3) *general cost function* (none of the previous algorithms can even handle the cost function $\mu_P + 3\sigma_P$. Finally, we discuss applications of the SSP problem to maze routing, buffer insertions, and timing analysis under parameter variations.

1. INTRODUCTION

With continued technology scaling, parameter variations have become a major factor that affects circuit performance and could lead to excessive yield loss [1]. Variations in device and interconnect do not appear to be scaling at the same rate with the technology. For technology nodes of 65 nm or below, design methodologies that consider this important issue are needed to avoid over-pessimistic design or yield loss in manufacturing.

Graph algorithms are widely used in VLSI CAD and many CAD problems can be formulated as shortest path problems. Traditional shortest path algorithms can handle graphs only with deterministic edge weights. In order to model uncertainty due to parameter variations, we need to use probability distributions for edge weights. In this paper, we consider the statistical shortest path (SSP) problem. Given a graph G in which edge weights are random variables. For each path P in G, let L_P be its length which is the sum of all edge weights on P. Clearly L_P is a random variable. Let μ_P and σ_P^2 be the mean and variance, respectively, of L_P. In the SSP problem, our goal is to find a path P to minimize the cost function $\mu_P + \Phi(\sigma_P^2)$ where Φ is an arbitrary function. For example, if $\Phi(x) = 3\sqrt{x}$, the cost function is $\mu_P + 3\sigma_P$. This cost function is widely used to measure the performance or yield when the distributions are Gaussian. To minimize uncertainty in the final result, it is meaningful to look for paths with bounded variance, i.e., $\sigma_P^2 \leq B$ for a given fixed bound B.

In this paper, we present an exact algorithm to solve the SSP problem in $O(B(V+E))$ time where V and E are the numbers of vertices and edges, respectively, in G. Our algorithm assumes all edge-weight variances are integers. (For graphs with non-integer edge-weight variances, we can simply discretized the range of real numbers for variances with desirable precisions and apply the algorithm designed for integer variances.) The main idea of our algorithm is to expand G into a larger graph G' by splitting each node into a number of nodes. New edges are added and edge weights (real numbers) are assigned intelligently. The resulting graph G' is guaranteed to be directed acyclic and that the deterministic shortest path in G' gives an optimal path in G. There were some previous efforts on the SSP problem [2–5]. Our algorithm is superior to previous algorithms because we can handle

- general graphs
- arbitrary edge-weight distributions
- general cost function

Note that previous algorithms for the SSP problem were designed for directed acyclic graphs with specific edge-weight distributions such as Gaussian. Moreover, none of the previous algorithms can handle the general cost function $\mu + \Phi(\sigma^2)$. In fact, none of them can handle the important cost function $\mu + 3\sigma$ even for Gaussian distributions.

The SSP problem has many applications in CAD for VLSI. We will briefly discuss its applications to timing analysis, maze routing and buffer insertion under parameter variations. For buffer insertion and timing analysis, the graphs are directed acyclic, but for maze

*This work was partially supported by the National Science Foundation under grant CCR-0306244

0-7803-9451-8/06/$20.00 ©2006 IEEE.

routing the graphs are of general forms. We have applied our SSP algorithm to some problems in these applications and the results are encouraging. For example, in statistical timing analysis, our algorithm can find not only the worst $\mu_P + \Phi(\sigma_P^2)$ delay bound, but also the longest delay path candidates, even the delay distributions at outputs are not Gaussian.

The rest of paper is organized as follows. In Section 2, we will formally present the SSP problem. In Section 3, an efficient algorithm is proposed to find the optimal solution for SSP problem. In Section 4, some techniques are discussed to further improve its efficiency. We will present some CAD applications for the SSP problem in Section 5 and conclude the paper in Section 6.

2. STATISTICAL SHORTEST PATH PROBLEM

Given a directed graph G, we are interested to find a path from a vertex s (called source) to a vertex t (called sink). Such a path is called an $s-t$ path. As stated in the last section, the goal of the statistical shortest path (SSP) problem is to find a path P from s to t in G such that the cost function $\mu_P + \Phi(\sigma_P^2)$ is minimized, where L_P is the length of the path P, μ_P is the mean value of L_P, σ_P^2 is the variance of L_P, and Φ is an arbitrary function.

For each edge e in G, let X_e be its edge-weight distribution, μ_e be the mean of X_e, and σ_e^2 be the variance of X_e. All edge-weight distributions are assumed to be mutually independent. It is well known that if X and Y are independent random variables and $Z = X + Y$, then the mean and variances of Z can be obtained by adding the means and variances of X and Y, respectively.

Consider an $s-t$ path P in G. From the fact that means and variances are additive, it follows that:

$$\mu_P = \sum_{e \in P} \mu_e, \quad \sigma_P^2 = \sum_{e \in P} \sigma_e^2$$

Therefore, if the cost function is of the form $\mu_P + k\sigma_P^2$, then the SSP problem can be easily solved by assigning a real-valued weight $\mu_e + k\sigma_e^2$ to each edge and solving the traditional deterministic shortest path problem. Unfortunately, this approach would not work for the general cost function $\mu_P + \Phi(\sigma_P^2)$, since

$$\Phi(\sigma_P^2) \neq \sum_{e \in P} \Phi(\sigma_e^2)$$

For example, if $\Phi(x) = \sqrt{x}$, we have $\Phi(\sigma_P^2) = \sigma_P$, which is the standard deviation. It is well known that standard deviations are not additive. As we will see later that an important cost function is $\mu_P + k\sigma_P$ (i.e., $\Phi(x) = k\sqrt{x}$), we need to find a new approach to solve the problem.

We now discuss the importance of the cost function $\mu_P + k\sigma_P$. Typically, one would like to design for the worst case. First, let us assume all edge-weight distributions are Gaussian. Since adding two Gaussian distributions results in a Gaussian distribution, all path-length distributions are Gaussian. It is well known that with a Gaussian distribution for L_P, we have

$$P(|L_P - \mu_P| \leq 3\sigma_P) > 0.99$$

Therefore, one can use the cost function $\mu_P + 3\sigma_P$ to minimize for the worst case. As for general edge-weight distributions, according to the Chebyshev's inequality,

$$P(|L_P - \mu_P| \leq k\sigma_P) > 1 - \frac{1}{k^2}$$

Clearly, the larger the k, the smaller is the "tail" probability. Therefore, we can fix a value for k to define what is considered to be the worst case. It follows that minimizing the cost function $\mu_P + k\sigma_P$ is for minimizing the statistical worst case path length.

As we mentioned in Section 1, we only consider $s-t$ paths P such that $\sigma_P^2 \leq B$ for a given fixed bound B. This is because we want to minimize uncertainty in the final result due to large variance σ_P^2. By bounding the variance of the path length of P, we have the following lemma.

LEMMA 1. *Let P be an $s-t$ path in G with $\sigma_P^2 \leq B$. Let u be a vertex on P and let $Q \subseteq P$ be an $s-u$ path. We have $\sigma_Q^2 < B$.*

PROOF. Let P be $< v_0, v_1, v_2, ..., v_n >$ where $v_0 = s$ and $v_n = t$. Let P_i be the subpath of P from v_0 to v_i. Let e_i be the edge (v_{i-1}, v_i). Note that P_{i+1} is obtained by concatenating P_i with e_{i+1}. Since variances are additive, it follows that $\sigma_{P_{i+1}}^2 = \sigma_{P_i}^2 + \sigma_{e_{i+1}}^2 > \sigma_{P_i}^2$. Thus the path length variances monotonically increase along the path P. Since the maximum path length variance is B which is at $v_n = t$, all other path length variances on the path P are strictly less than B. The lemma follows since u is on P. □

Finally, we assume all edge-weight variances are integers. This assumption allows us to design an algorithm to exactly solve the SSP problem. Note that for graphs with non-integer edge weights, we can discretize the range of real numbers for variances within desirable precision and then apply our algorithm.

3. ALGORITHM

We note that only μ and σ^2 are additive, not $\mu + \Phi(\sigma^2)$ with arbitrary function of Φ. So it is no longer true that the optimal path must consist of optimal subpaths. Without this *optimal-substructure property* [6], algorithms for classical shortest path problem are not valid for SSP problem.

Our approach to solve the SSP problem is to reconstruct the graph so that the optimal-substructure property is satisfied. We will construct a new graph G' from the original G. G' is a graph with deterministic edge weight, and thus the existing algorithms for classical shortest path problem can be used to find the shortest path in G'. Furthermore, the shortest path in G' must correspond to the shortest path in G whose path-length distribution has the minimum $\mu + \Phi(\sigma^2)$ value.

Note that the edge-weight distribution can be captured by μ and σ^2, and both of them are additive. We modify the graph so that only one of them is stored on the edge. Thus the edge weight becomes a deterministic number. The problem is reduced to how to modify the original graph G into a new graph G' so all edge weights become deterministic without losing the random variable information.

We use the node-splitting technique to achieve this objective. Since the SSP problem is to find optimal $\mu + \Phi(\sigma^2)$, which is linear in μ but non-linear in variance. We will use μ as the new edge weight in G'. Expanded vertices are used to preserve the variance. We split node

$u \in G$ into a set of nodes $\{u_1, u_2, \cdots, u_i, \cdots, u_B\}$ in G'. Note that each $s-u$ path in G has a corresponding $s-u_i$ path in G'. The variance of the length of an $s-u$ path in G must be an integer between 1 and B according to Lemma 1. Node u_i in G' represents the end point of an $s-u$ path in G with path length variance i. For each u_i, we call i the variance-index or just var-index, and write var-index$(u_i) = i$. According to Lemma 1, we only need to consider paths with path-length variances bounded by B in G. So for a node $u \in G$, we only create B nodes in G'. Of course, it is not necessary to expand the source node s. We create a vertex s_0 node in G' to represent s.

After vertex splitting, we create the edges in the new graph G'. We have three different cases. First of all, consider the edges from source node s_0. As shown in Figure 1, for any edge from s to a in G, we will create a corresponding edge in G'. It points from s_0 to a_i. Assume the edge-weight of (s, a) has mean μ_e and variance σ_e^2. Then, $i = \sigma_e^2$ and $w(s_0, a_i) = \mu_e$, where $w(s_0, a_i)$ is the deterministic edge weight of (s_0, a_i).

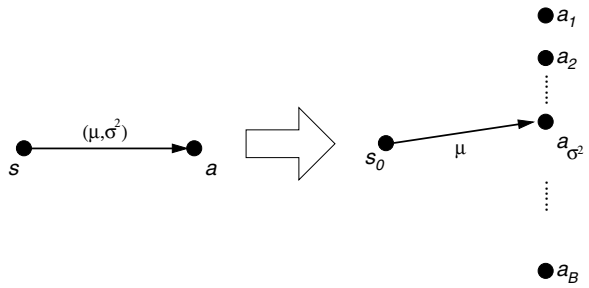


Figure 1: Illustration of node splitting for the source

Second, consider any edge $e = (u, v)$ in G where $u \neq s$ and $v \neq s$. μ_e and σ_e^2 are mean and variance of the edge weight of (u, v), respectively. As shown in Figure 2, u and v are divided into two sets of nodes in G'. Again, because of the additive property of variance, we will create the edge point from u_i to v_j, where i and j satisfy the following equation:

$$j = i + \sigma_{uv}^2$$

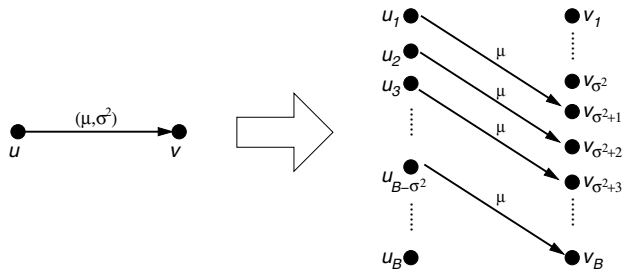


Figure 2: Illustration of node splitting

The edge weights are assigned as μ_e as illustrated. We also notice the created edges in G' are non-crossing. Thus, for each edge in the original graph G, there will be less than B edges created in G'

Finally, consider the sink node t in G. After vertex splitting, the sink node t in G is also divided into B nodes in G'. To find the shortest path from s to t, we have to create a new dummy node in G' to represent t in G. As shown in Figure 3, t' is created as the sink node in G'.

According to the vertex splitting and edge creation procedures we discussed, any path from s_0 to t_i corresponds to a path from s to t in G with path-length variance i. To capture the $\Phi(\sigma^2)$ term in the cost function, we assign $\Phi(i)$ as the edge weight from t_i to t'.

Thus the new deterministic graph G' is constructed. The psudocode is shown as BUILDGRAPH. It is obvious that any path P' from s_0 to t' in G' corresponds to a path P from s to t in G. Because for any edge or vertex in the path Q from s_0 to t_i, it is mapped to one edge or vertex in G. So Q corresponds to a distinct path P in G. In G', t_i is connected to t by one edge, which implies the P' also corresponds to a distinct Q. Thus P' must corresponds to one P in G.

We can also prove that distinct P in G maps to distinct P' in G'. Let P be a path in G. For any sub-path of P from s to u, the path-length variance is known and fixed. So u corresponds to a vertex u_i in G'. Assume (u, v) is an edge in P, and v corresponds to vertex v_j in G'. Since only one edge (u_i, v_j) exists in G', for any $(u, v) \in P$, it maps to distinct (u_i, v_j). Thus the one-to-one correspondence between P and P' is proved.

```
BUILDGRAPH:
    create source node s0 in G'
    s0 corresponds to s in G
    for each node u in G except s
        Create B vertices {u1, ··· , uB} in G'
    for each edge e = (u, v) in G
        for each node ui
            j = i + σe²
            if j ≤ B
                create edge (ui, vj) in G'
                w(ui, vj) = μe
    Create t' in G'
    for each ti
        Connect ti to t'
        w(ti, t') = Φi
```

It is trivial to prove that P' has a path-length $\mu_P + \Phi(\sigma_P^2)$. We have the following theorem:

THEOREM 1. *The shortest path in G' which is created by* BUILDGRAPH *from G corresponds to the optimal path P in G which has the minimum $\mu_P + \Phi(\sigma_P^2)$ for the path-length distribution.*

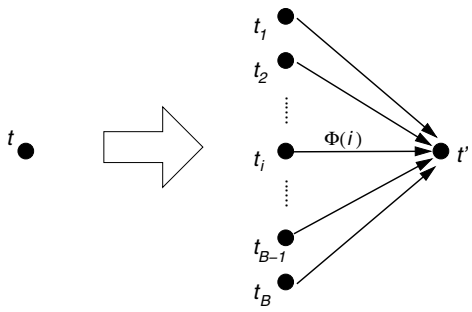


Figure 3: Illustration of node splitting for the sink

Theorem 2 states an important property of G'.

THEOREM 2. *The G' created by* BUILDGRAPH *from G is a directed acyclic graph (DAG). Moreover, G' has at most B levels.*

PROOF. Suppose there is a cycle $C =< v_0, v_1, ..., v_m >$ in G' where $v_0 = v_m$. According to Lemma 1,

$$\text{var-index}(v_0) < \text{var-index}(v_1) < \cdots < \text{var-index}(v_m)$$

This is a contradiction since $v_0 = v_m$. To see that G' has at most B levels, it suffices to show that every path P in G' has at most B edges. Note that the variance-index of the nodes on P is monotonically increasing (Lemma 1). If P has more than B edges, the var-index for the last node u on P has var-index$(u) > B$, contradicting that maximum var-index for every node is B. Therefore P has at most B edges. It then follows that G' has at most B levels. □

It is well known that the shortest path problem in a directed acyclic graph (DAG) can be solved in $O(V + E)$ time. Moreover, existing linear time shortest path algorithms for DAG can handle positive and negative edge weights. Thus $\Phi(\sigma_p^2)$ can be arbitrary function with either positive or negative function values. In G', $E' < B \cdot E$ and $V' < B \cdot V$. So the SSP problem can be solved in $O\big(B(V + E)\big)$ time.

4. IMPROVED IMPLEMENTATION

Since G' must be a DAG, we can create G' in a topological order. Thus, we don't have to create all B nodes for each vertex in G, which not only speeds up the runtime, but also saves the memory usage.

The first step is to create the s_0 vertex in G' and create those vertices adjacent to s_0 according to the structure of G. This step is similar to the procedure in BUILDGRAPH. For an edge (s, u) in G, we will create the u_i and connect the edge (s_0, u_i). We will also store some information in each created vertex u_i:

- Its parent $\pi(u_i) = s_0$.
- The path-length mean $m(u_i)$ from s to u_i
- The level of this vertex. It is used to represent the topological order. Now the level $l(u_i) = 1$. Here we assume the level of s_0 is 0 because any edge point to s_0 is redundant and can be ignored.

Then assume we already expand G' into level k. For each node u_i in level k, we will create v_j according to the edge $e = (u, v)$ in G, assuming $j = i + \sigma_e^2 \leq B$. If v_j doesn't exist in G', we create v_j, connect (u_i, v_j) and store the following in v_j:

- $\pi(v_j) = u_i$
- $m(v_j) = m(u_i) + \mu_e$
- $l(v_j) = k + 1$

If v_j already exists, we will compare $m(v_j)$ and $m(u_i) + \mu_e$. If $m(v_j) \leq m(u_i) + \mu_e$, it means the path from s through u to v is not a sub-path of optimal path from s to t. We don't do anything on v_j. Otherwise, the path from s through u to v is a better solution, so we update the information in v_j.

We will repeat this procedure to level n. If nodes in level n satisfy either one of the following conditions, we will terminate our algorithm:

- $n = B$. Now all vertices of level n must have the variance equals to B. According to Theorem 2, it cannot create any vertex in G' from these vertices.
- All vertices in level n corresponds to t in G.

We terminate this algorithm by connecting all t_i in G' to t' and calculate $m(t_i) + \Phi(i)$ to find the shortest path.

Figure 4 illustrates an example of our faster approach. Assume we use $B = 25$ to construct G' from G in Figure 4(a). Using BUILDGRAPH, 100 vertices need to be created by vertex splitting. By the improved implementation, the SSP problem can be solved very efficiently as shown in Figure 4(b). It only creates 10 vertices in G'.

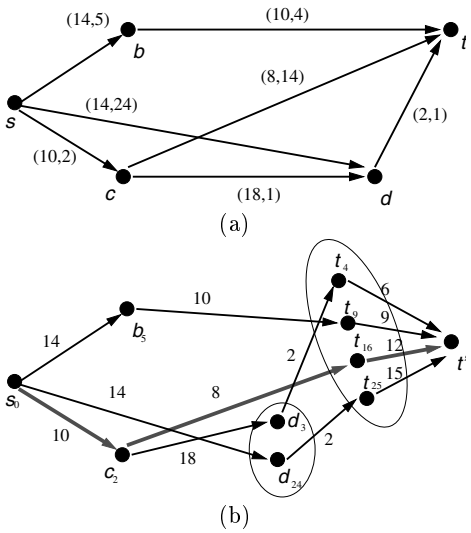


Figure 4: Improved implementation. (a) is the original graph G. The edges are annotated by (μ, σ^2) as the edge weights (b) is the expanded graph G'. The shortest path has a minimum $\mu + 3\sigma$ value.

5. APPLICATIONS

Many problems in CAD for VLSI can be formulated to SSP problem when variations become a concern. We will here briefly discuss its applications to maze routing, timing analysis and buffer insertion under parameter variations.

5.1 Maze Routing

Maze routing is to find the shortest path with minimum length in a grid routing problem. Traditionally, edge weights are assigned as real-valued number by cost function. However, parameter variations make it necessary to model the edge weights as random variables. The cost functions are often related to those parameters with variations. Considering the parameter variations, the maze routing problem can be formulated as a SSP problem.

Assume we want to find the optimal routing solution for a critical net. So we use the wire delay as the cost function. Considering the parameter variations, delay is not only the function of wire length. The geometric

process variations come from various sources. For example, the 3σ value of wire width variation could reach 25% of nominal wire width [7]. Thus the delay value changes substantially. Because of the systemic variations, the nominal values of wire resistance and capacitance per unit length are no longer uniform within one die. Furthermore, temperature variation can also impact the performance.

Without considering the variation, the cost function is $D = f(\mathbf{P}, T)$, where $\mathbf{P}$ is a set of geometric parameters related to wire delay, and T is the temperature. Now, process variations become significant. $P_i \in \mathbf{P}$ are random variables. All these variations are assumed to have a distribution in Gaussian. We use first order approximation [8] to calculate the delay distribution:

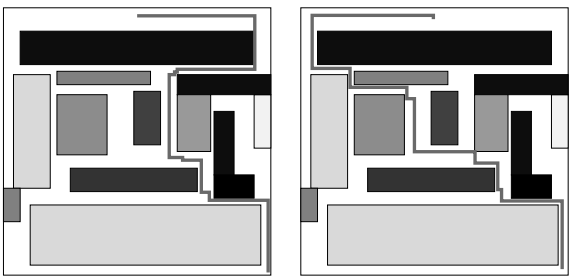
$$D' = f(\mathbf{P_0}, T) + \sum_{P_i \in \mathbf{P}} \frac{\partial f}{\partial P_i} \Delta P_i \qquad (1)$$

where P_0 is a set of the nominal value of parameters. ΔP_i are parameter variables with zero mean. Now the new cost function D' will assign a Gaussian distribution to edge weight. Consider the systemic variation, we use the following equation to calculate the mean value of edge weight:

$$\mu = g(x, y, \mathbf{P_0}, T)$$

The variance can be calculated directly from Equation 1.

Since the path length is now obviously Gaussian, we use $\mu + 3\sigma$ to find the optimal path length. Now the maze routing problem is formulated as a SSP problem. It can be solved by our proposed algorithm.



(a) (b)

Figure 5: Compare the maze routing with or without consider the variations. (a) is the shortest path found by classical shortest path problem and (b) is the result by solving SSP problem.

We have applied our SSP algorithm to solve the maze routing problem. Physical parameters for 65nm technology from ITRS2001 [9] are used. 3σ values of these parameters are set to be 30% of their nominal values. Elmore's delay model is used to calculate the delay these physical parameters. Figure 5 shows one of the comparisons between the traditional approach and our method. The darker region in Figure 5 means higher temperature. To make the routing wire clear, the temperature profiles are only shown in the blocks. Figure 5(a) shows the results without considering variation. To make the comparison fair, we use temperature profile to calculate the mean delay value. Systemic variations are also considered. In Figure 5(b), all variations are considered and the shortest path is found by solving the SSP problem. It is clear that the path in Figure 5(b) intelligently avoids the hot spots.

5.2 Time Analysis

It is always a concern to predict the circuit performance accurately. Precise timing information is needed for circuit optimization to meet the yield or to avoid over design. Timing analysis considering variations is extensively studied recently. Path-based or block-based algorithms have been proposed to find the statistical longest paths and the delay distributions. Path-based approaches usually depend on static timer to find out a set of longest path candidates, and then the statistical approaches can be performed [10, 11]. Block based algorithms can get the delay distributions by propagating the random variables. However, they need path-based analysis to find out the longest paths [12, 13].

Our algorithm can be modified to find the earliest or latest arrival time. To perform timing analysis, a circuit is modeled as a DAG. The delay distributions for cells and interconnects are assigned as the weights of edges. Then our algorithm can be used to find the longest path with maximum $\mu + k\sigma$ value, or the shortest path with minimum $\mu - k\sigma$ value. Since this timing analysis is to find the extreme case of delays, the bound B could be large. With the pruning technologies similar to [14], our algorithm can find the longest or shortest path very efficiently, even if we set B to be infinity.

Another note on our proposed method is that the delay distribution for the longest path is not the delay distribution for the corresponding output. To find the true distribution of an output, we need to take MAX operation on delays of different paths to this output [10]. And even if the path delays are Gaussian, the delay distribution at the output is not necessarily Gaussian.However, we can use the $D_{MAX} = \mu + k\sigma$ value of the longest path as a good bound for delay distribution at output.

Our algorithm stated in Section 3 can also be modified to find longest path candidates. We will check all the edges to the sink node t'. If the delay distribution is not stochastically smaller than the longest path, we will treat it as a candidate of the statistical longest paths.

Table 5.2 shows some experimental results on ISCAS benchmark circuits. Our algorithm is performed to find the longest paths as well as the bound D_{MAX} ($k = 3$). The number of longest path candidates is labeled N_P in the table. Monte Carlo method is used for comparison. We perform 10000 runs of Monte Carlo analysis for each circuit. And the N_U shows the cases in Monte Carlo analysis which delays exceed the bound D_{MAX} (out of 10,000). We also show the mean μ_{mc} and variance σ^2_{mc} of Monte Carlo analysis results. Since the delay distribution at an output is not Gaussian, $\mu_{mc} + 3\sigma_{mc}$ could lead to a worse bound for timing analysis. Column T shows the runtime for different testbench circuits.

bench	D_{MAX}	T	N_P	μ_{mc}	σ_{mc}	N_U
C432	75.0546	0.05s	95	73.73	0.47	19
C499	45.2776	0.02s	186	43.91	0.49	13
C880	72.6244	0.03s	13	71.26	0.48	20
C1355	61.6536	0.05s	93	60.46	0.43	9
C1908	109.891	0.77s	134	108.27	0.54	4
C2670	103.08	6.07s	2959	102.20	0.62	12
C3540	132.986	3.35s	585	131.05	0.64	13
C5315	121.185	0.71s	19	119.28	0.64	15
C6288	267.311	22.91s	33	265.22	0.775	21
C7552	103.156	0.64	4	101.83	0.445	8

Table 1: Timing analysis on ISCAS benchmark

5.3 Buffer Insertion

Buffer insertion is a widely used interconnection optimization method. It can also be formulated as a shortest path algorithm [15]. Figure 6 illustrates a simple example with three possible buffer locations. The input driver resistance is R_d and the output load capacitance is C_L.

We construct corresponding graph as shown in Figure 6(b), where s is the source node which represents the driver, t is the sink node of the graph which corresponds to the load of the wire. The remaining vertices a, b and c represent three possible buffer locations. Each edge corresponds to a wire segment between two possible buffers. We treat driver and load as buffers for convenience.

Edges are always directed along the signal transmission direction, The edge weight is defined as the delay t_d from one buffer input to the next buffer input. So $t_d = t_g + t_w$, where t_g is the delay of the buffer and t_w is the delay of the wire.

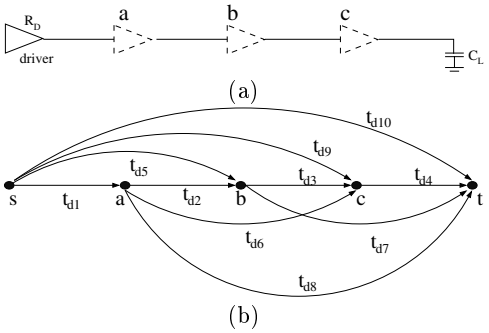


Figure 6: Formulate buffer insertion into shortest path problem. (a) is a small buffer insertion example with 3 possible locations. (b) Graph is built from (a).

Now consider the parameter variations. Delays of buffers and interconnects are modeled as random variables. All t_i in Figure 6(b) are now distributions instead of deterministic numbers. The buffer insertion becomes a SSP problem.

Similar to the maze routing problem, we use Elmore delay model. The parameter data are from ITRS. Table 5.3 shows some runtime results for our method. The wire length is set to be 10mm. 45nm technology parameters are used. And we set $B = 200$. It runs on a Linux box with 1GHz Pentium III CPU and 512MB memory. LMAX is the possible buffer locations uniformly distributed on the wire. BUF# is the number of inserted buffer. and μ_{delay} and σ_{delay} are mean and standard deviation of shortest path, respectively.

LMAX	BUF#	μ_{delay}	σ_{delay}	runtime
50	11	536	18.8	<0.01s
100	10	537	18.7	2.1s
150	11	536	18.8	17s

Table 2: Buffer Insertion Results

6. CONCLUSION

In this paper, we proposed a new algorithm to exactly solve the statistical shortest path problem. It can find the optimal solution in $O(B(V + E))$ time. Techniques are also proposed for improvement. This algorithm can be used in various applications in nanometer designs when the parameter variations become a concern.

7. REFERENCES

[1] Shekhar Borkar, Tanay Karnik, Siva Narendra, Jim Tschanz, Ali Keshavarzi, and Vivek De. Parameter variations and impact on circuits and microarchitecture. In *Proc. of the 40th Design Automation Conference*, pages 338–342. ACM Press, 2003.

[2] H. Frank. Shortest path in probabilistic graphs. *Oper. Res.*, 17:583–599, 1969.

[3] C. Elliott Sigal, A. Alan B. Pritsker, and James J. Solberg. The stochastic shortest route problem. *Opers. Res.*, 28:1122–1128, 1980.

[4] R. P. Loui. Optimal path in graphs with stochastic or multidimensional weights. *Comm. of ACM*, 26:670–676, 1983.

[5] Ishwar Murthy. Stochastic shortest path problems with piecewise-linear concave utility functions. *Management Science*, 44:125–136, 11 1998.

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2001.

[7] Duane Boning and Sani Nassif. *Design of High-Performance Microprocessor Circuits*. 2002.

[8] Kanak Agarwal, Dennis Sylvester, David Blaauw, Frank Liu, Sani Nassif, and Sarma Vrudhula. Variational delay metrics for interconnect timing analysis. In *DAC 2004*, pages 381–384. ACM Press, 2004.

[9] Semiconductor Industry Association. *International Technology Roadmap for Semiconductors*, 2001.

[10] Michael Orshansky and Kurt Keutzer. A general probabilistic framework for worst case timing analysis. In *Proc. of the 39th Design Automation Conference*, pages 556–561. ACM Press, 2002.

[11] Hongliang Chang and S.S. Sapatnekar. Statistical timing analysis considering spatial correlations using a single pert-like traversal. In *Proc. International Conference on Computer Aided Design*, pages 621–625, 2003.

[12] A. Davgan and C. Kashyap. Block-based static timing analysis with uncertainty. In *Proc. of international conference on Computer Aided Design 2003*, pages 607– 614, 2003.

[13] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan. First-order incremental block-based statistical timing analysis. In *Proc. of the 41st Design Automation Conference*, pages 331–336, New York, NY, USA, 2004. ACM Press.

[14] Chirayu S. Amin, Noel Menezes, Kip Killpack, Florentin Dartu, Yehea Ismail, Umakanta Choudhury, and Nagib Hakim. Statistical static timing analysis: How simple can we get? In *Proc. of the 42nd Design Automation Conference*, pages 652–657, New York, NY, USA, 2005. ACM Press.

[15] Y. Gao and D. Wong. A graph based algorithm for optimal buffer insertion under accurate delay models. In *Proc. of the conference on Design, automation and test in Europe*, pages 535–539. IEEE Press, 2001.