Path Criticality Computation in Parameterized Statistical Timing Analysis

Jaeyong Chung, Jinjun Xiong*,

Vladimir Zolotov*, and Jacob A. Abraham

Computer Engineering Research Center, University of Texas at Austin

**IBM T.J. Waston Research Center, Yorktown Heights*

Outline

- Introduction
- Path criticality formulation
  - $$P(\bigcap_{i=1}^{m} A_k < B_k)$$
- A novel method to compute a joint probability in SSTA
- Conclusions

Introduction

- In SSTA, criticality is a representative metric to gauge how important a given edge or path is in terms of timing
- The *criticality* of a path is defined as the probability that the path becomes the critical path
- The *criticality* of an edge is defined as the probability that the edge is on the critical path

Introduction

- Criticality is used in timing/yield optimization
  - Gate sizing, buffer insertion, Vth assignment
  - Transistor sizing
- Criticality is also very useful in testing
  - Timing critical paths (i.e., paths with high path criticality values) can be selected using SSTA
  - An ATPG tool takes these paths and generates test patterns sensitizing them
  - These patterns can be employed in performance testing, SDD testing, and speed-binning

Previous Work

- Run SSTA
- Obtain the circuit slack Sc
- Obtain the slack $S(p_1)$ of a given path p_1
- Compute $P(S(p_1) < S_c)$

$$S_c = \min\{S(p_1), S(p_2), S(p_3), \ldots\}$$

Xiong et al, Incremental criticality and yield gradients, DATE 2008

Previous Work

- The complement slack of a path is the minimum of all path slacks in the circuit excluding the path slack
- Obtain the complement slack $\overline{S}(p_1)$ from $S(p_1), S_c$
- Compute

$$P(S(p_1) < \overline{S}(p_1))$$

$$\overline{S}(p_1) = \min\underline{\{S(p_2), S(p_3), \ldots\}}$$

Lots of information is captured by a too simple linear form

Xiong et al, Incremental criticality and yield gradients, DATE 2008

General path criticality formulation

- Partition the set of all paths in the circuit into ***m*** groups
- Compute the minimum path slack of each group
- Path criticality of p_1 can be written as

$$P\left(\bigcap_{i=1}^{m} S(p_1) < S_i\right)$$

1. Various ways to formulate path criticality
2. Smart formulation considering non-idealities of SSTA can reduce errors

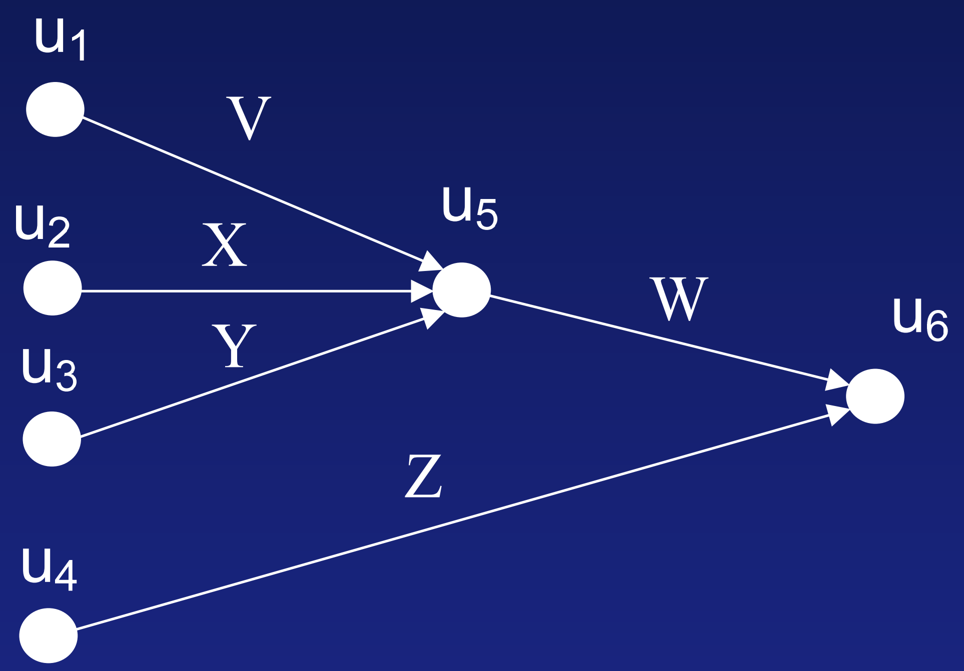
Partitioning

group 1 — $p_1 : u_1 \rightarrow u_5 \rightarrow u_6$

group 2 — $p_2 : u_2 \rightarrow u_5 \rightarrow u_6$

$p_3 : u_3 \rightarrow u_5 \rightarrow u_6$

group 3 — $p_4 : u_4 \rightarrow u_6$



$$P((V+W>\max\{X+W,Y+W\})\bigcap(V+W>Z))$$

$$=P((V+W>\max\{X,Y\}+W)\bigcap(V+W>Z))$$ **distributivity**

$$=P((V>\max\{X,Y\})\bigcap(V+W>Z))$$

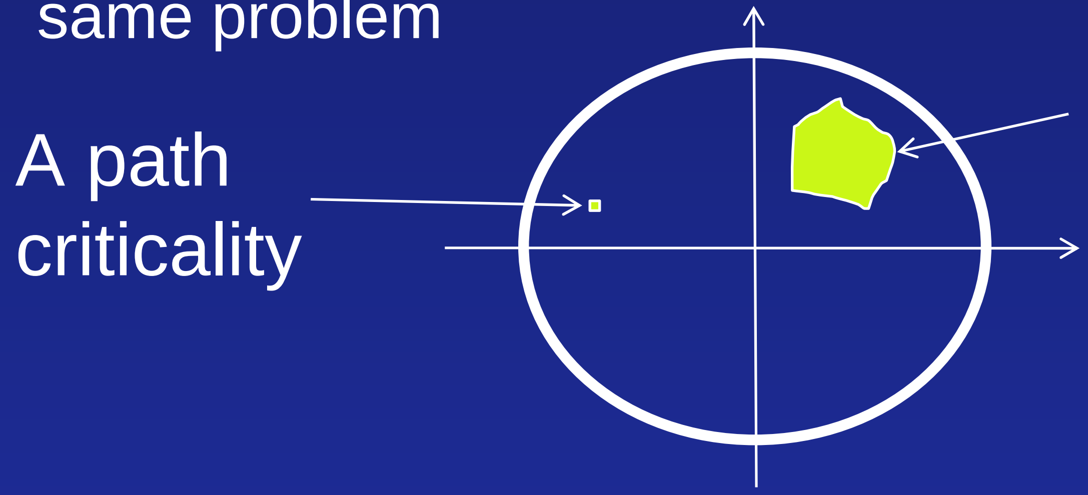
Computing a joint probability

- Path criticality computation is reduced to evaluating the multivariate normal CDF

$$P\left(\bigcap_{i=1}^{m} A_k < B_k\right) = P\left(\bigcap_{i=1}^{m} A_k - B_k < 0\right)$$

$$= \Phi(x_1, x_2, \ldots, x_m)$$

- Edge criticality computation is also reduced to the same problem



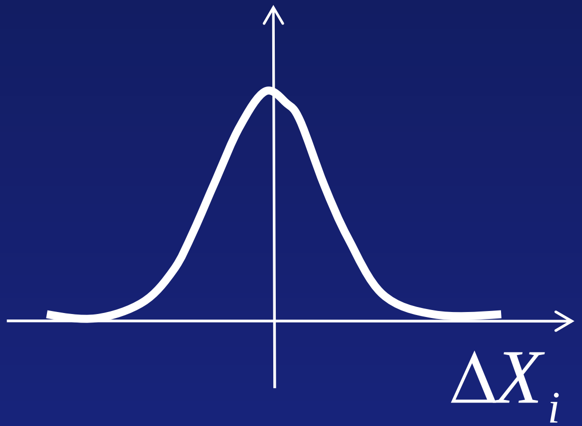
A new way to evaluate the CDF

- Previous approaches to evaluate the CDF
  - Numerical Integration (e.g.,mvncdf in matlab)
    - Accurate but extremely slow
  - Monte Carlo sampling (slow)
  - Using the max operation provided by SSTA
    - Very fast but poor accuracy
- We propose a novel, analytic conditioning operation
  - 1000x faster than Monte Carlo sampling at the same accuracy
  - 2~3x accurate at the cost of 3~4x runtime compared to max operation

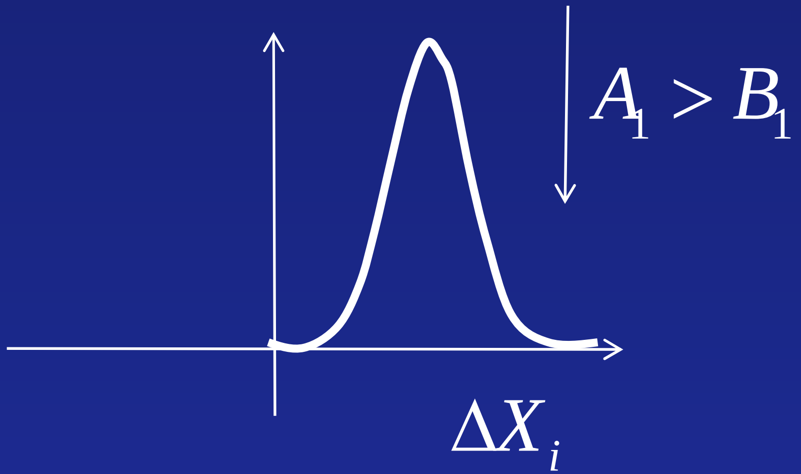
Conditioning operation

$$P\left(\bigcap_{i=1}^{2} A_k < B_k\right) = P(A_1 < B_1)\,P(A_2 < B_2 \mid A_1 < B_1)$$

$$A_2 = a_0 + \sum_{i=1}^{n} a_i \Delta X_i + a_{n+1} \Delta R_a$$



$$A_2 \big|_{A_1 > B_1} = a_0 + \sum_{i=1}^{n} a_i \Delta X_i + a_{n+1} \Delta R_a$$



Conditioning operation

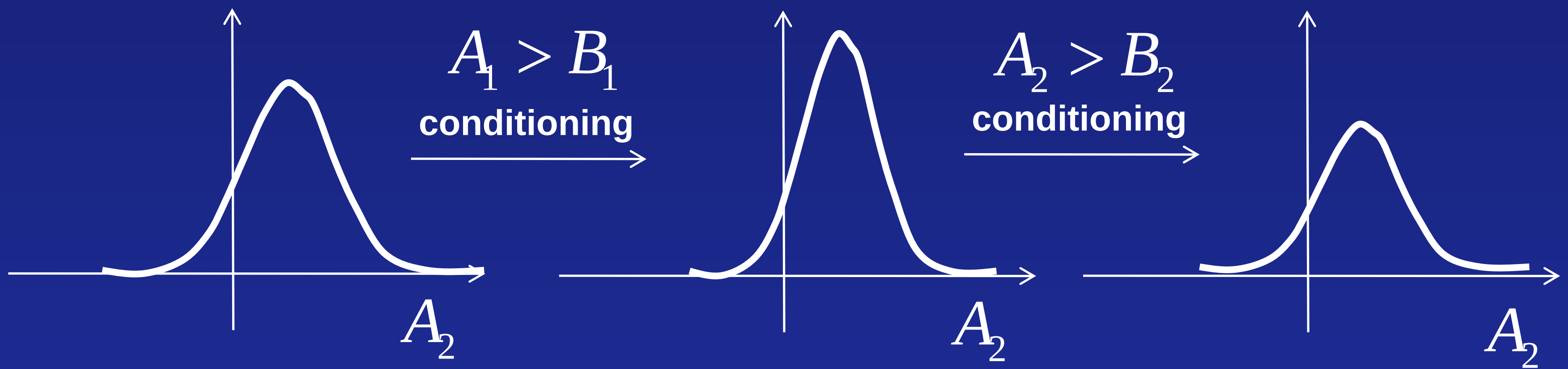
- Theorem 1. Let A and B be normal R.V.s.

$$E[\Delta X_i \mid A > B] = E[\Delta X_i] + \beta(\mathrm{cov}[A, \Delta X_i] - \mathrm{cov}[B, \Delta X_i]) / a$$

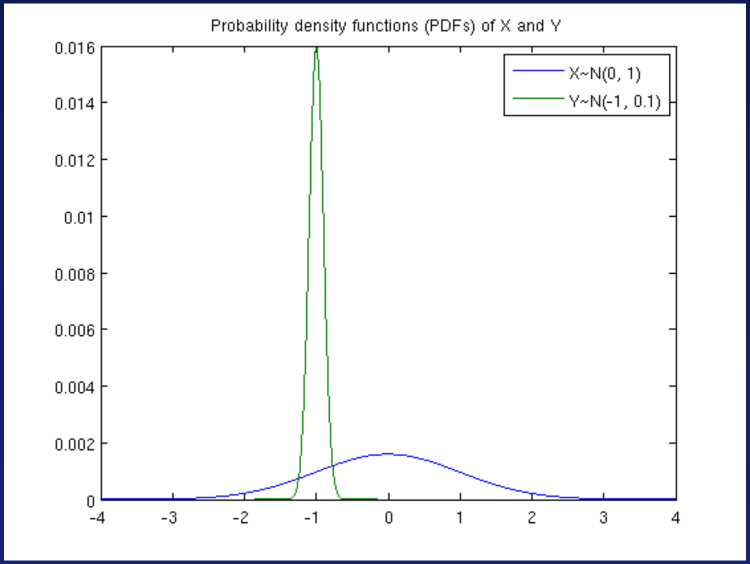
$$\mathrm{cov}(\Delta X_i, \Delta X_j \mid A > B) = \mathrm{cov}[\Delta X_i, \Delta X_j] - (\beta^2 + \alpha\beta)$$

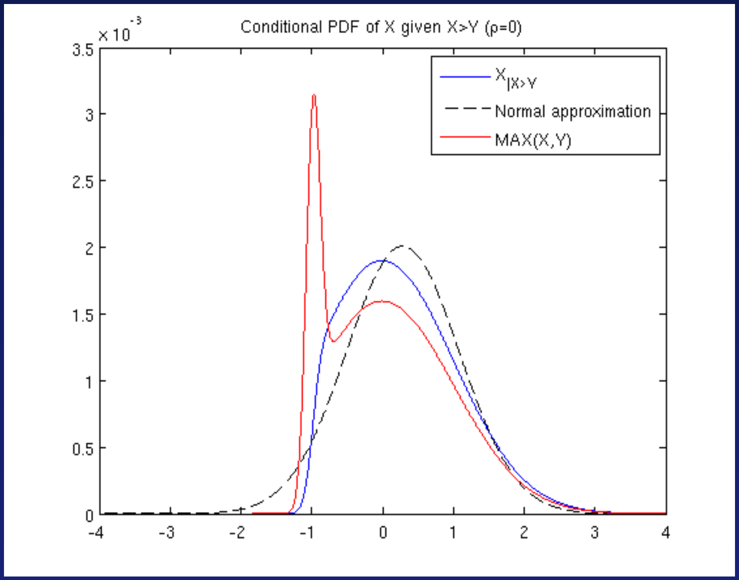
$$(\mathrm{cov}[A, \Delta X_i] - \mathrm{cov}[B, \Delta X_i])(\mathrm{cov}[A, \Delta X_j] - \mathrm{cov}[B, \Delta X_j]) / a^2$$

$$A_2 |_{A_1 > B_1} = a_0 + \sum_{i=1}^{n} a_i \Delta X_i + a_{n+1} \Delta R_a \qquad \mu = [\ldots], \Sigma = [\ldots]$$



Conditioning operation



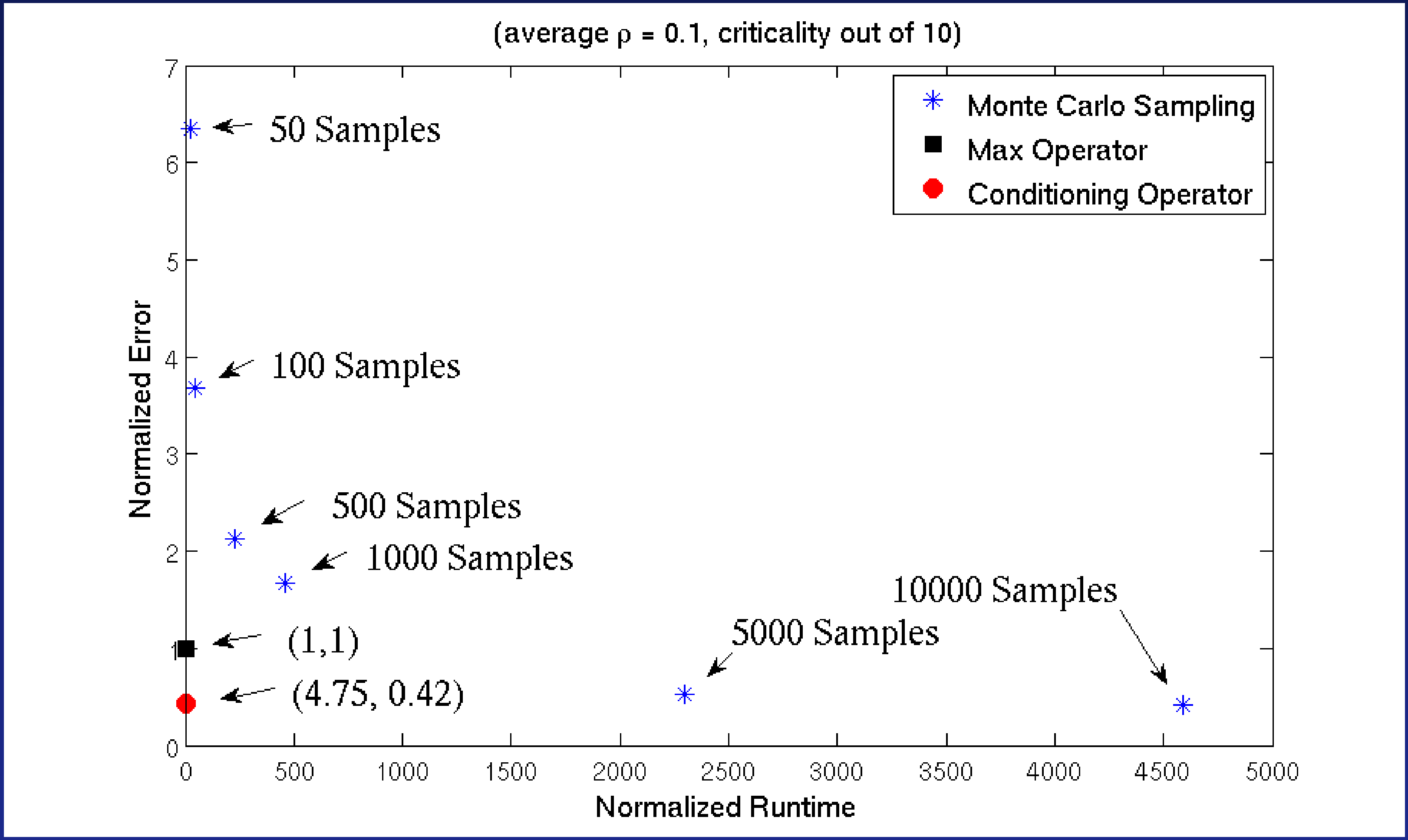


- Error analysis of normal approximation in max operation [D. Sinha et. al. TCAD 2007]
- The same analysis was done for conditioning operation
  - More than 2x less error
  - Error is much less for positively correlated timing quantities

Experimental Results

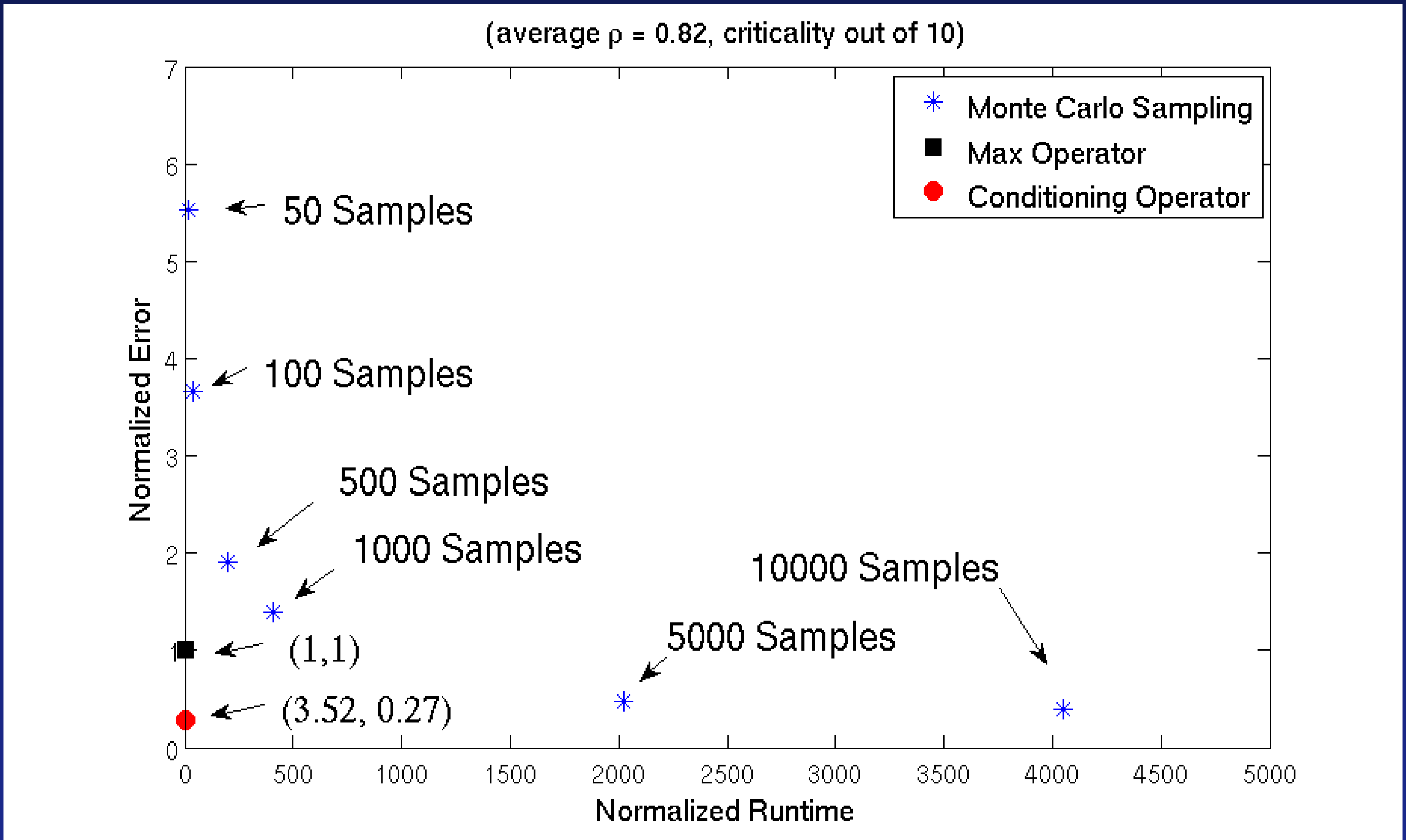
- Randomly generated 10 timing quantities represented in the canonical form with 21 global sources of variation
  - Mean: within a range of [1.0, 3.0]
  - Std.: from 10% to 20% of the mean
  - Sensitivity values:
    - Case 1) chosen within a range of [-1.0,1.0] and then normalized in order to meet the std. value
    - Case 2) chosen within a range of [0,1.0] and normalized
- Compute criticality of randomly chosen one out of the 10 timing quantities

Experimental Results



Case 1 $\rho = 0.1$

Experimental Results

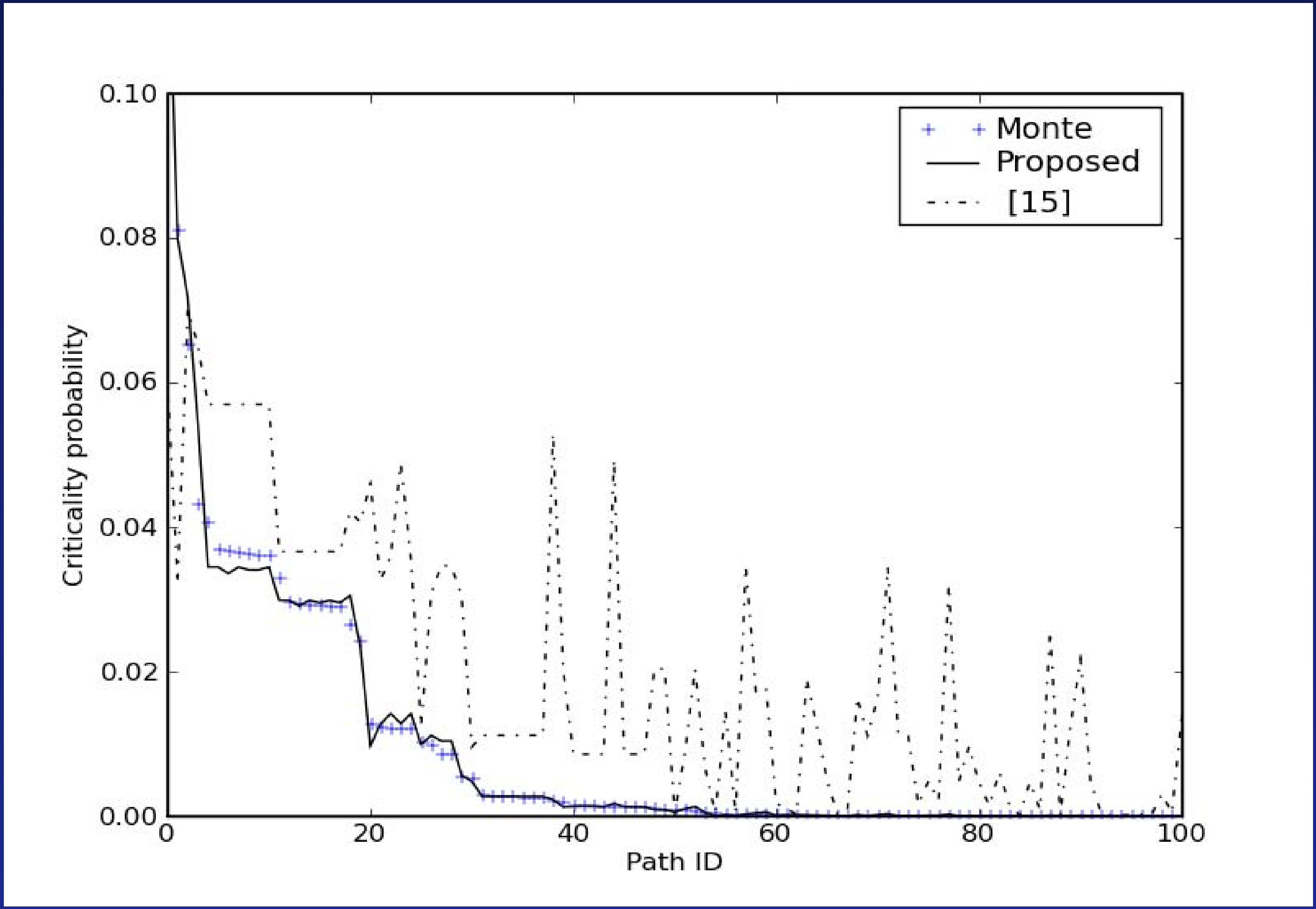


Case 2 $\rho = 0.82$

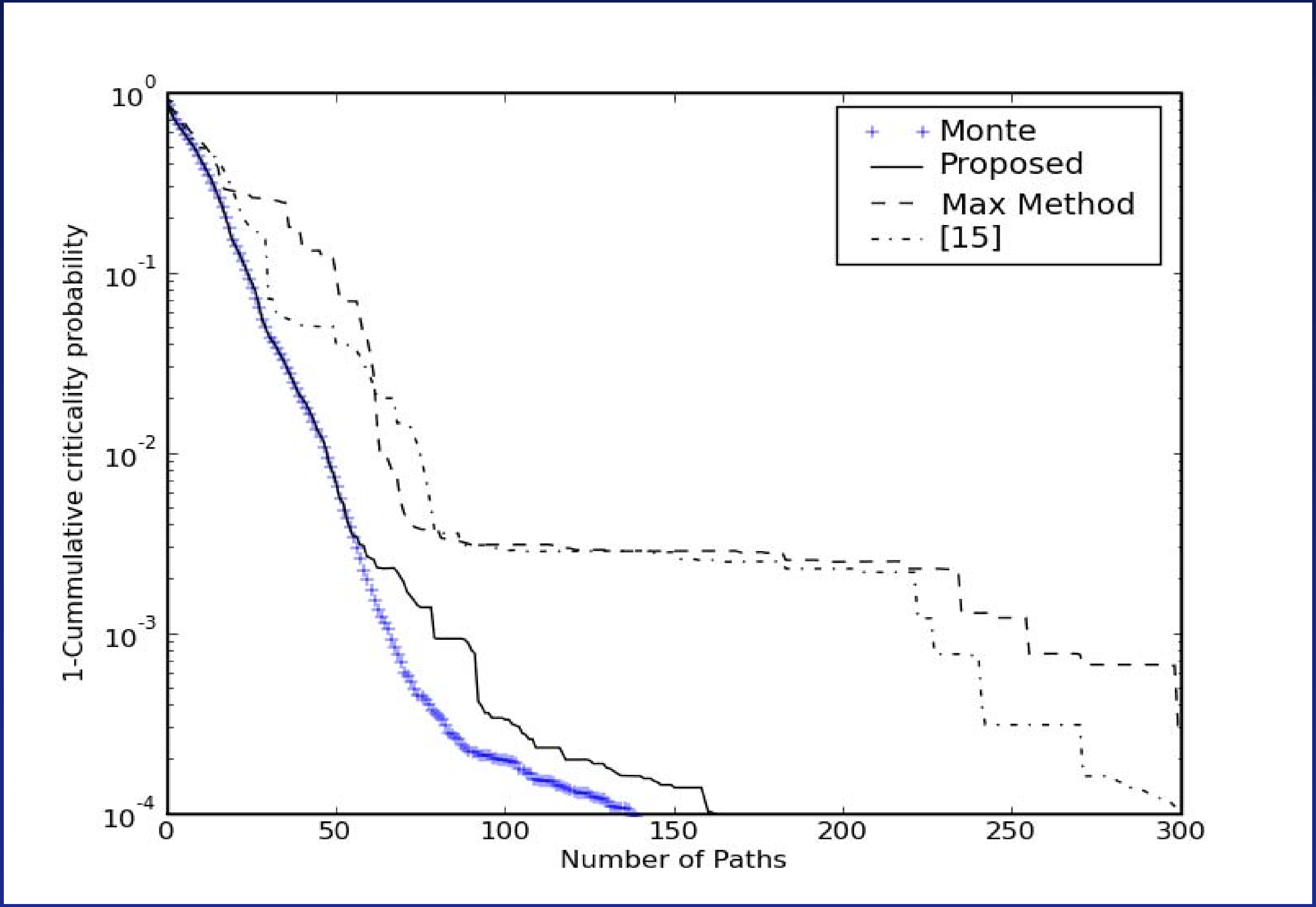
Experimental Results

- SSTA algorithm used: [Visweswariah DAC 2004]
- Refactoring is employed: [Chung et al ICCAD 2009]
  - Capture topological (structural) correlation
  - Improve the accuracy of the arrival times
- Spatial correlation model: A quad tree with 3 levels
  - 4%, 5%, and 6% variation at 1^{st}, 2^{nd}, and 3^{rd} level
  - 21 global sources of variation
- 5% random independent variation

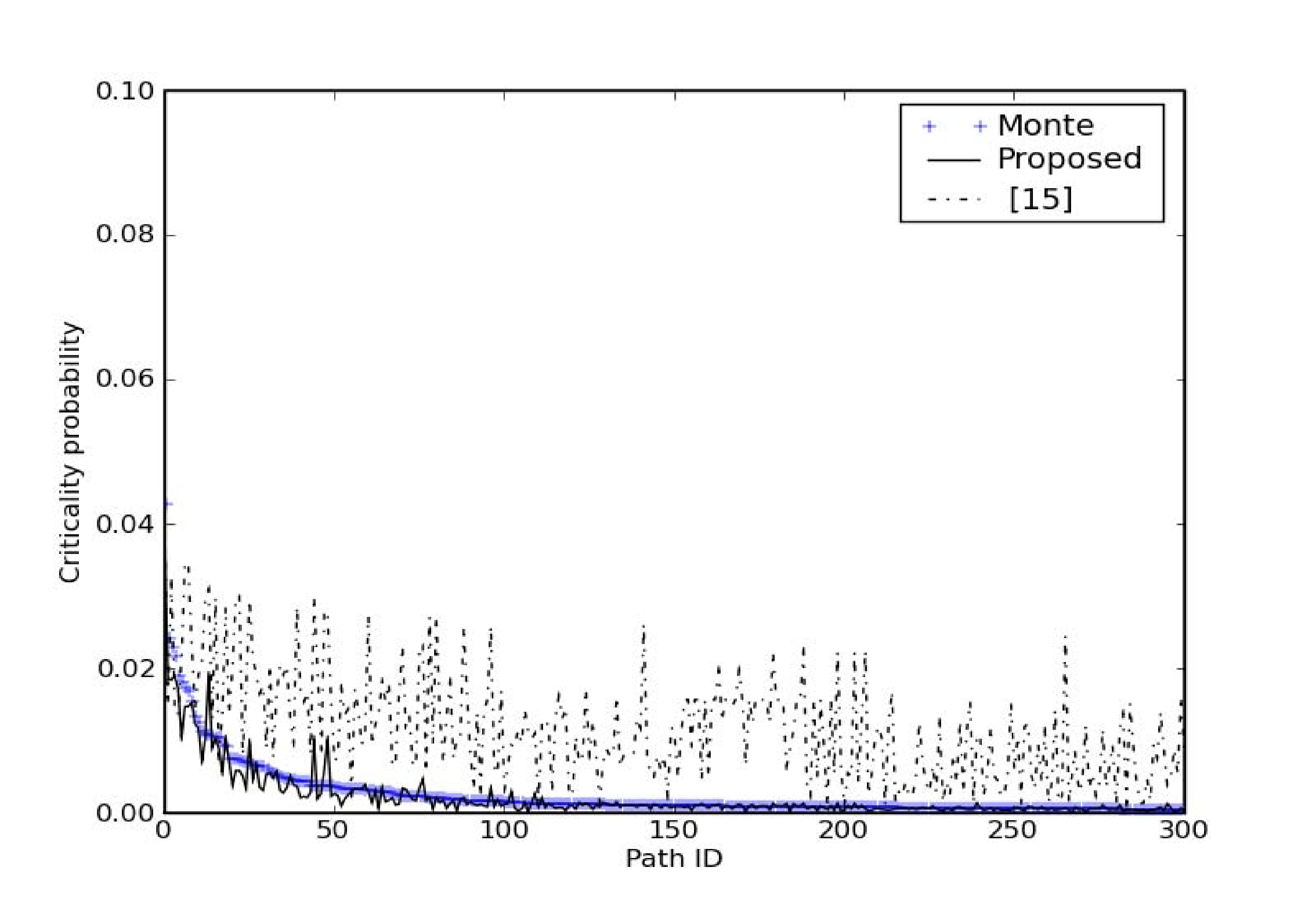
Experimental Results



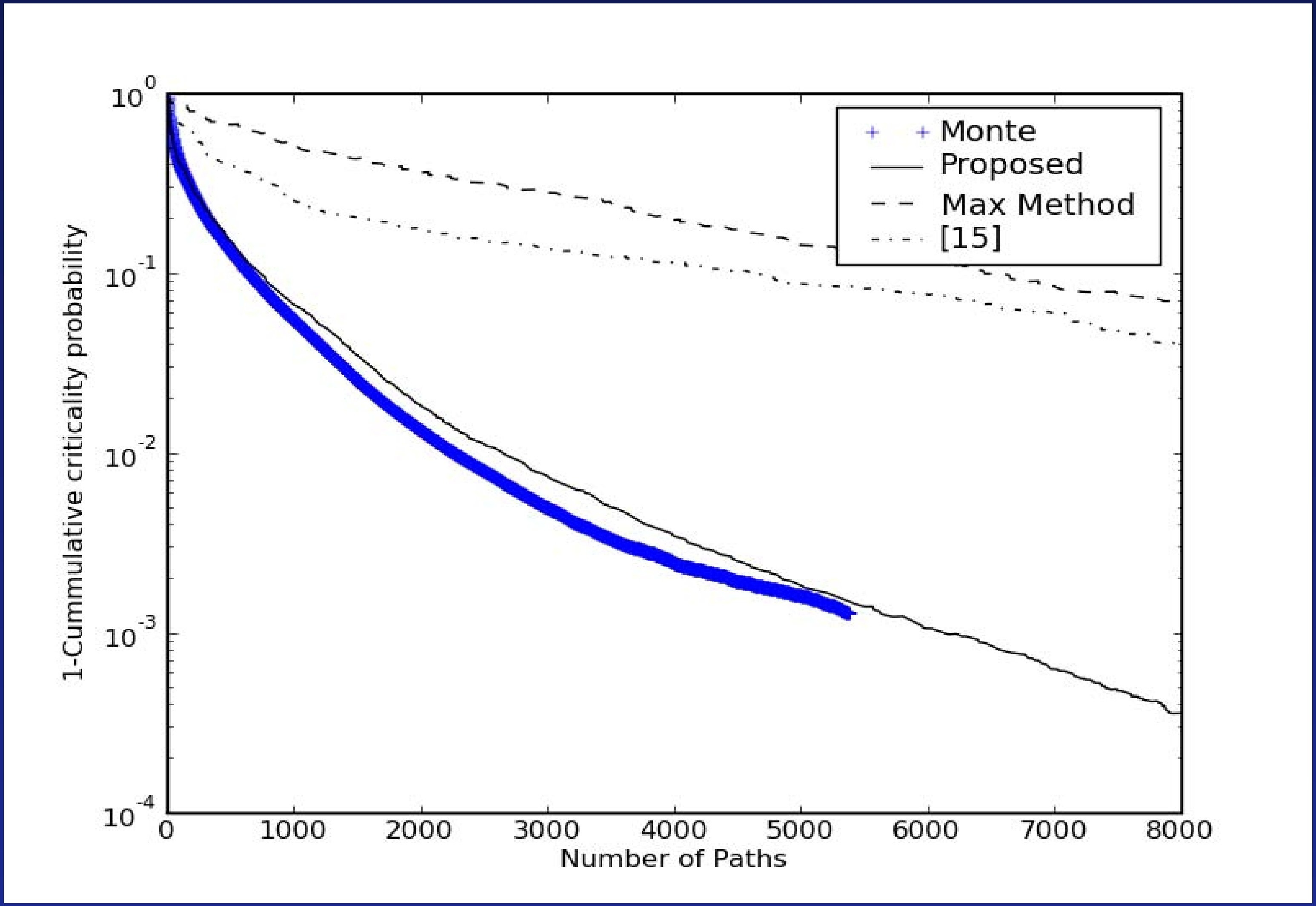
Experimental Results



Experimental Results



Experimental Results



Conclusions

- If you develop a statistical algorithm on top of SSTA, and our conditioning operation is employed to compute a certain joint probability,
  - Compared to the max operation
    - The quality of results can be improved significantly
    - The algorithm can become more stable
  - Compared to Monte Carlo sampling
    - Significant speed-up can be achieved
- This is demonstrated in path criticality computation

Conclusions

- Path criticality values are difficult to be computed accurately
- If you use the max operation, the accuracy change depending on the number of near-critical paths
- The combination of the conditioning operation and refactoring
  - allow us to compute it as accurate as Monte Carlo simulation unless your design is like c6288
  - is important when your design has many near-critical paths

Partitioning