Verification of Real-Time Systems using Statistical Model Checking

Dr. Jeffery P. Hansen
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM-0002028

What is the Problem?

Time-sensitive systems in uncertain environments have complex behaviors. How do we validate correct timing in such systems?

- Exact probabilistic verification is infeasible due to model size
- Black box testing haphazard and does not yield bounded predictions
- Need formal approach for dealing with uncertainty
- Need to achieve accurate, bounded, probabilistic results in a reasonable amount of time for rare outcomes.

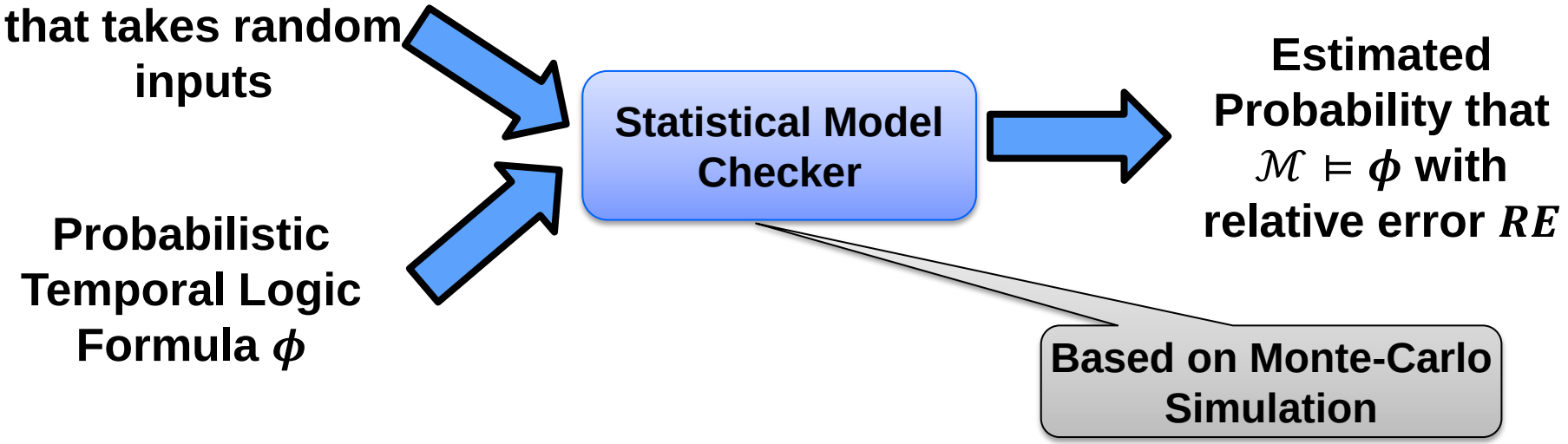
Use statistical model checking to do a “smart sampling of the world”

- Simulation captures both random variables and timing (scheduling)
- Importance sampling “tilts” input distributions for efficient probability estimation of “rare” events.

Note: We use “probability estimation” based statistical model checking. There is also a “hypothesis testing” based version.

Statistical Model Checking

$$Relative\ Error = \frac{Std.\ Dev.}{Mean}$$



- System properties described in formal language (UTSL, BLTL, etc.)
- Property is tested on “sample trajectories” (sequence of states).
- Each outcome can be treated as a Bernoulli random variable (i.e., coin flip).

Statistical Model Checking

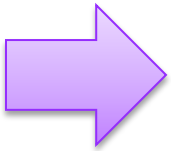
Goal: Calculate the probability p that some property holds:

$$p = E[I_{\mathcal{M} \vDash \Phi}(\vec{x})]$$

Where:

- $\vec{x}$ = vector of random variables
  - Represents all of the inputs or all random samples.
- $I_{\mathcal{M} \vDash \Phi}(\vec{x})$ = indicator function that returns 1 iff $\mathcal{M} \vDash \Phi$
  - Composition of system under test and property being tested.

```
total = 0;
for (i = 1;i <= 10;i++)
    total += rand();
assert(total <= 8);
```



$$I_{\mathcal{M} \vDash \Phi}(\vec{x}) = \begin{cases} 1 & if \sum_{i=1}^{10} x_i > 8 \\ 0 & otherwise \end{cases}$$

In this talk, we will consider the property Φ to be a "failure" condition.

Statistical Model Checking with Crude Monte-Carlo

The probability that condition Φ holds in model $\mathcal{M}$ when the input $\vec{x}$ is distributed according to joint pdf $f(\vec{x})$ is the expected value of that indictor function and can be calculated as:

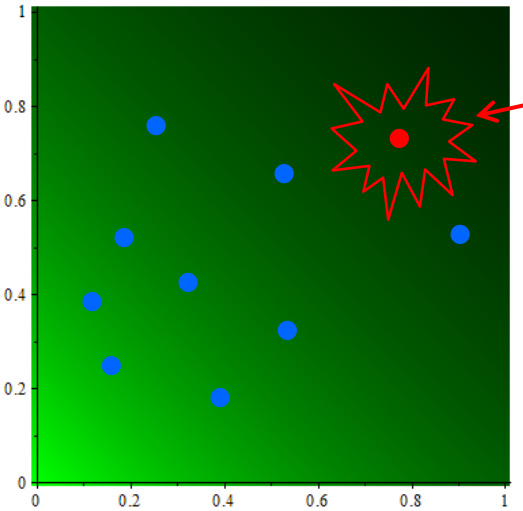
$$p = E[I_{\mathcal{M}\models\Phi}(\vec{x})] = \int I_{\mathcal{M}\models\Phi}(\vec{x}) f(\vec{x}) d\vec{x}$$

This can be estimated with Crude Monte-Carlo simulation as:

$$\hat{p} = \frac{1}{N}\sum_{i=1}^{N} I_{\mathcal{M}\models\Phi}(\overrightarrow{x_i})$$

where each $\overrightarrow{x_i}$ is a sample vector drawn from $f(\vec{x})$. As N gets large, $\hat{p}$ will converge to p.

Estimated Failure Probability



# of samples in fault region

$$\hat{p} = \frac{1}{10} = 0.1$$

total # of samples

Relative Error: How large should N be?

Measure of accuracy for a prediction.

Defined as ratio of standard deviation to mean. For a probability estimate, the estimated relative error is:

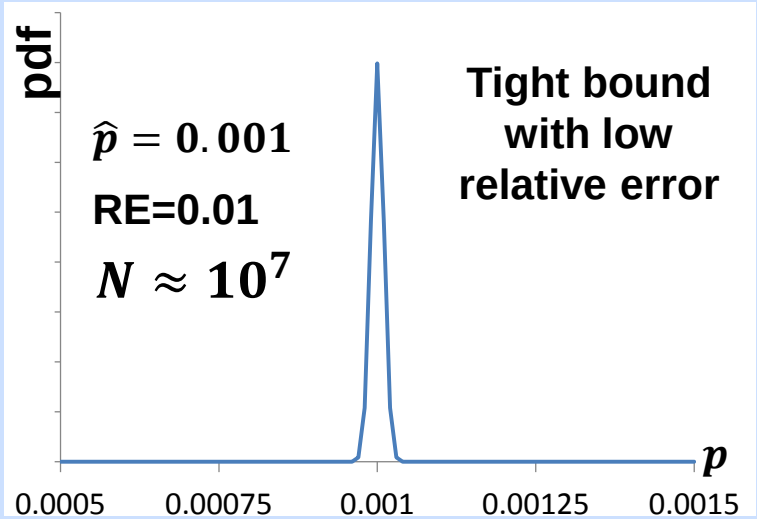
$$\widehat{RE} = \frac{\hat{\sigma}}{\hat{p}}$$

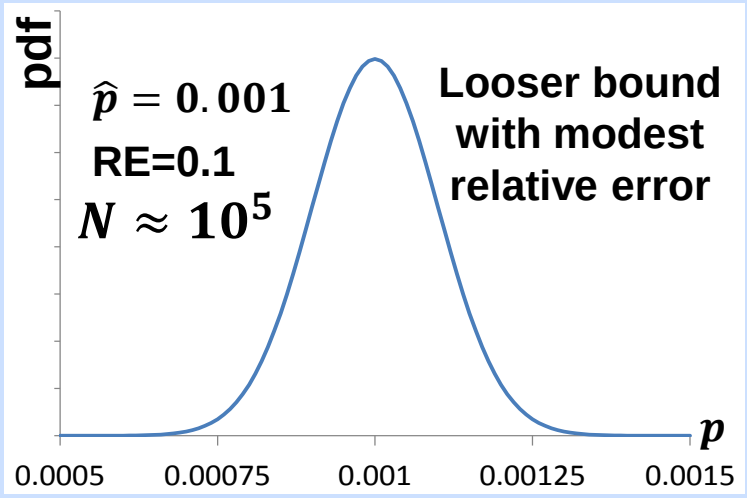
Number of samples to achieve a target relative error increases

- as target relative error decreases, or
- as estimated probability decreases

$$N \approx \frac{1}{p(RE)^2}$$

Distribution of actual p given estimated $\hat{p}$ and target relative error

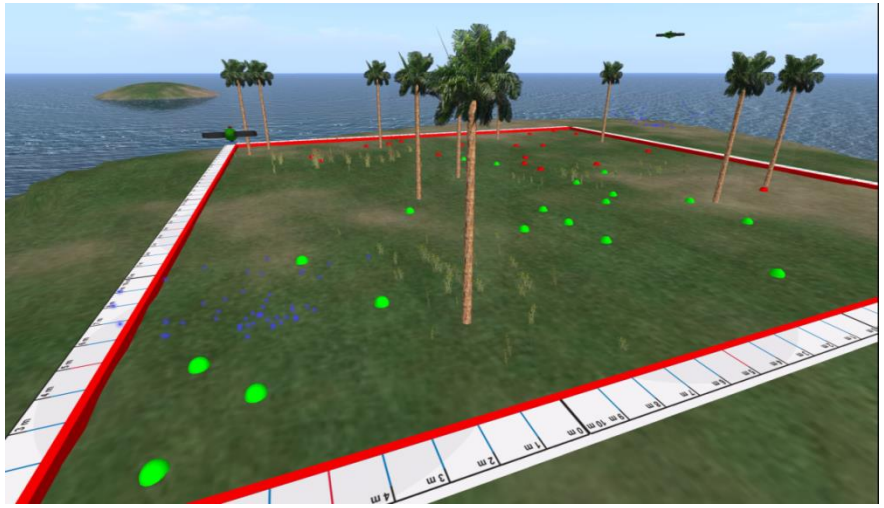




Example: UAS Mine Search

Problem: Real-time tasks with variable execution times

- Execution time depends on sensor data, e.g., number of detected obstacles.
- Each task has different deadline miss tolerance (e.g., would rather miss mine than collide with obstacle).



Task	Period	Priority	Criticality	Execution Time		Deadline Tolerance
				Base	Per Object	
Flight control loop	100ms	4	4	10ms	n/a	0
Mine detection	250ms	3	1	50ms	5ms	0.2
Obstacle avoidance	500ms	2	2	50ms	6ms	0.05
Obstacle detection	1000ms	1	3	50ms	5ms	0.05

Large number of objects causes overrun

$\mathcal{M} \models \Phi$ **iff all deadline tolerances are respected**

Example: UAS Mine Search

Compare Two Scheduling Strategies

- **Rate Monotonic Scheduling** (RM): Higher arrival rate ⇒ higher priority
  - Ignores task criticalities ⇒ Criticality inversion
  - Mine detection overruns cause UAS to crash into obstacle
- **Zero-slack rate monotonic scheduling** (ZSRM): Protect high criticality tasks from overruns in lower criticality tasks
  - Skip lower criticality tasks when higher criticality task overruns
  - **Guarantee:** Each task gets full CPU budget if all other tasks with higher criticality do not overrun during its execution

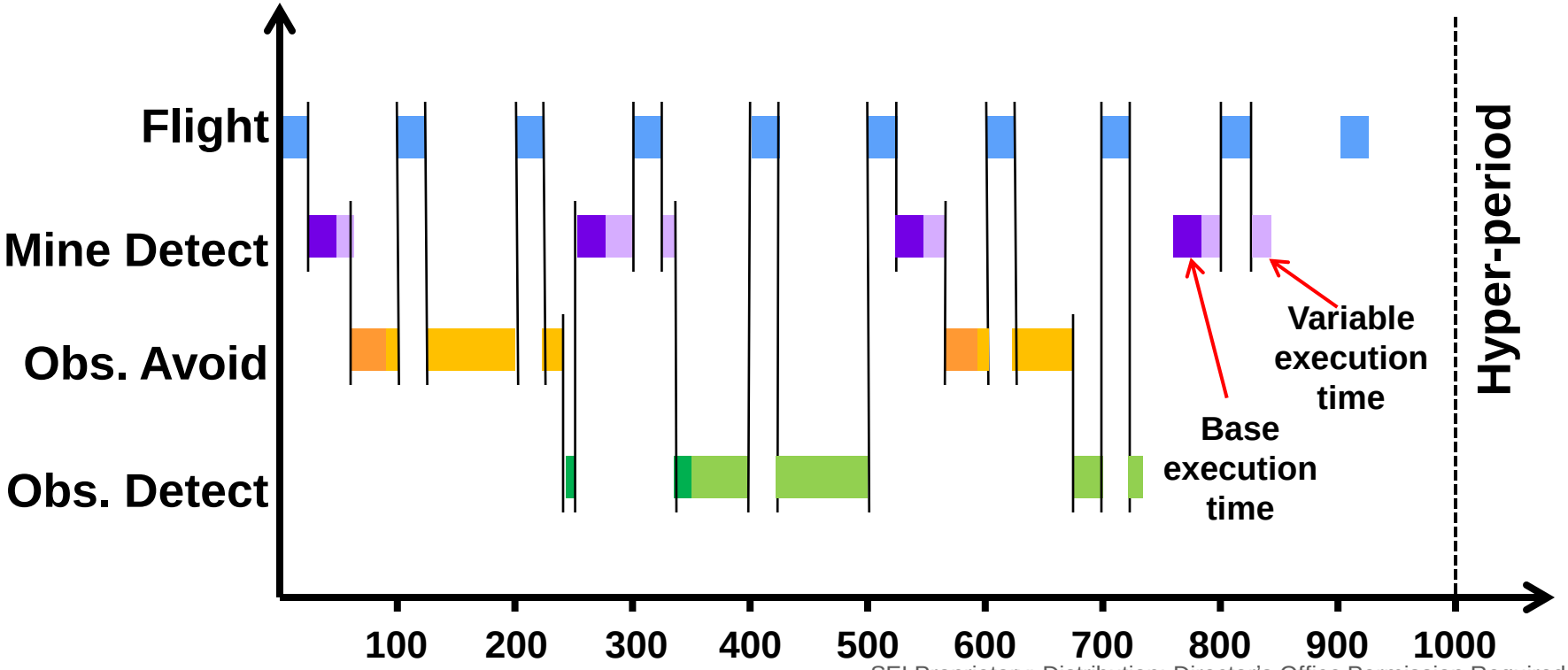
We expect…

- RM to perform better (i.e., $P(\mathcal{M}_{ZSRM} \vDash \Phi) > P(\mathcal{M}_{RM} \vDash \Phi)$), when we do not want any deadline misses.
- ZSRM to perform better (i.e., $P(\mathcal{M}_{ZSRM} \vDash \Phi) < P(\mathcal{M}_{RM} \vDash \Phi)$), when we can tolerate misses of low criticality tasks.

Hyper-Period-Based Simulation Approach

Assume independent behavior between hyper-periods

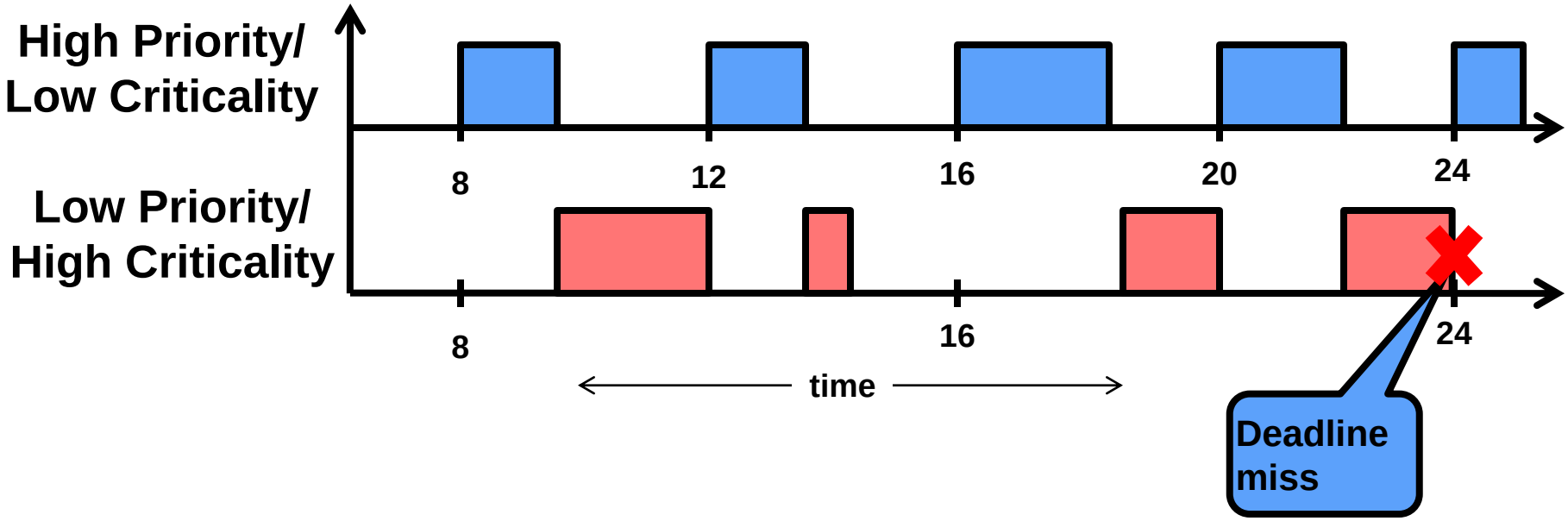
- Focus simulation effort on probability of failure in a hyper-period
- Simulate hyper-period many times using a given object density distribution.
- Apply analysis to extend predictions to system-level time scales.



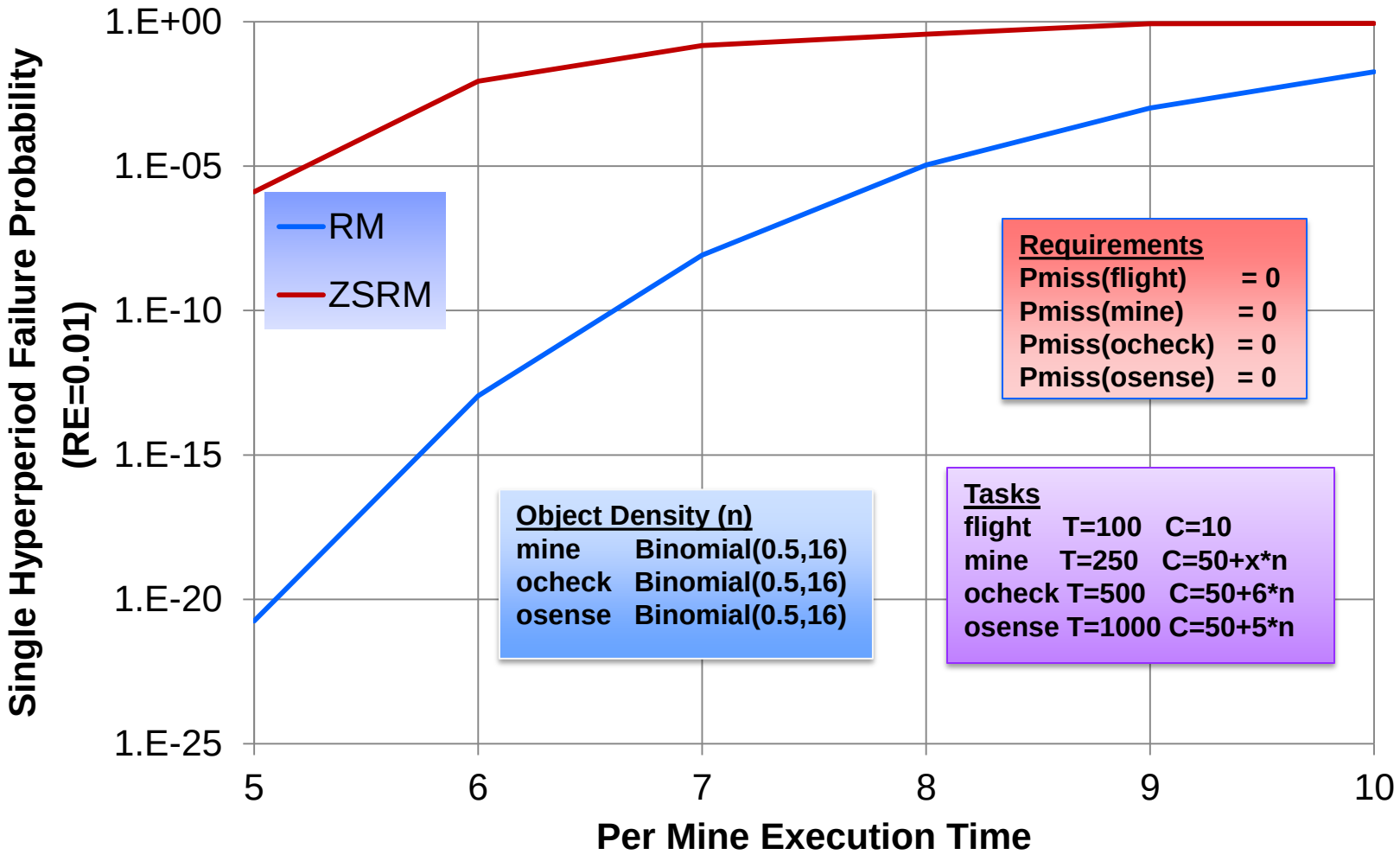
Criticality Inversion in Mixed-Criticality Tasks

Standard Rate Monotonic (RM) scheduling gives preferential treatment to high priority tasks.

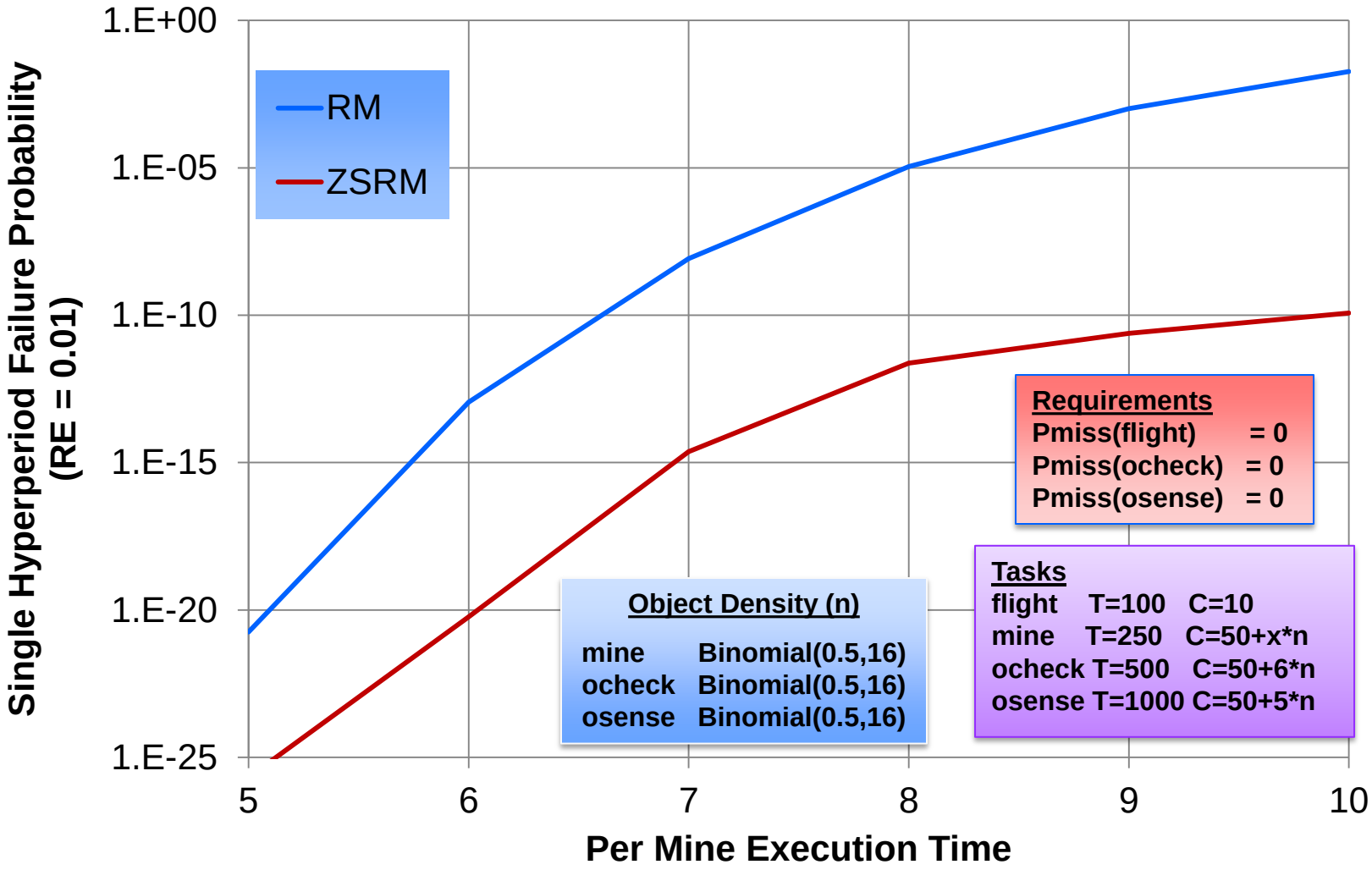
- RM priority determined by task period, not semantic importance
- Overloads can lead to “criticality inversion” where
- Zero-Slack Rate Monotonic (ZSRM) “fixes” this criticality inversion



Comparison of RM and ZSRM (Any Failure)



Comparison of RM and ZSRM (Flight Safety)



Mission Failure Probability

Assuming independence between hyper-periods, the mission failure probability is:

$$p_M = 1 - (1 - p)^N$$

where N is the number of hyper-periods and p is failure probability for one hyper-period.

For $Np < 0.1$, it can be shown that:

$$p_M \approx Np$$

and:

$$RE_M \approx RE$$

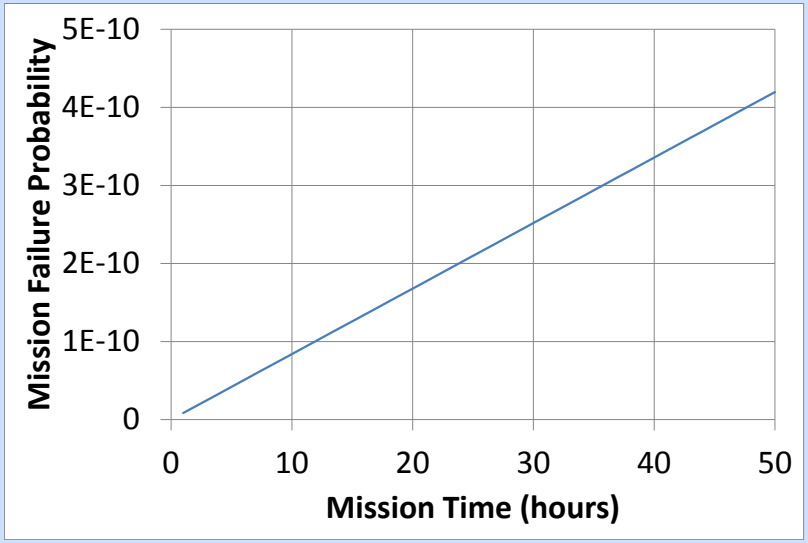
Example

Given a hyper-period with:

Period: 1 second

Failure Prob.: 2.36×10^{-15}

Mission failure probability (at least one deadline miss) is shown below.



Importance Sampling: Same RE with smaller N

Problem:

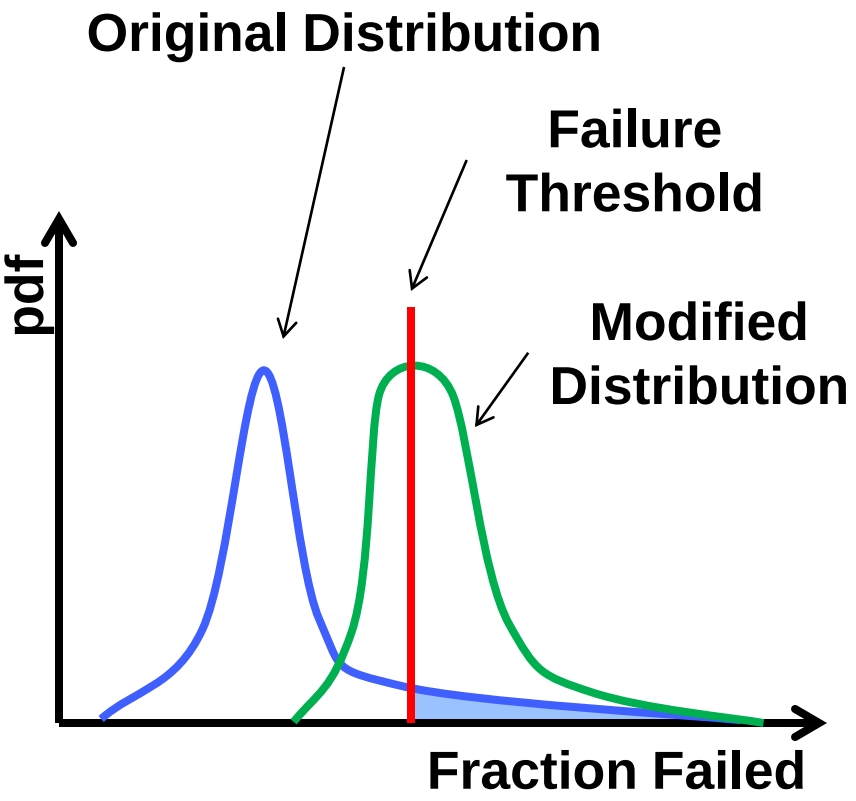
Estimating probabilities of rare events with low RE requires many samples.

- To estimate failure probability of $p = 10^{-5}$ with relative error of 0.01 would require one billion simulation runs.

Solution:

Use Importance Sampling to sample "important" area of a distribution.

- Sample with an "modified" distribution.
- Map back to original distribution.
- Can dramatically reduce number of experiments needed to verify "rare" events.



$$N \approx \frac{1}{p(RE)^2}$$

Importance Sampling

Recall probability of failure is:

$$p = \int I_{\mathcal{M} \vDash \Phi}(\vec{x}) f(x) dx$$

We can introduce an arbitrary density function $g(x)$ and rewrite as:

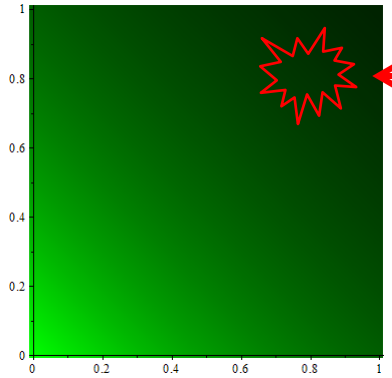
$$p = \int I_{\mathcal{M} \vDash \Phi}(\vec{x}) \frac{f(x)}{g(x)} g(x) dx$$

Now if we define $W(x) = \frac{f(x)}{g(x)}$ we get:

$$p = \int I_{\mathcal{M} \vDash \Phi}(\vec{x}) W(x) g(x) dx$$

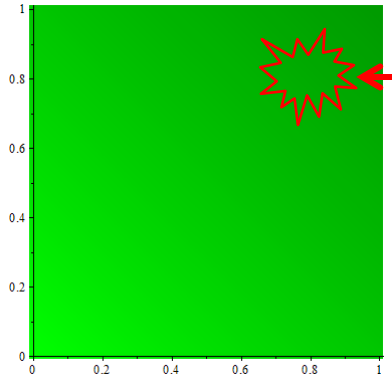
which is just the expected value of $I_{\mathcal{M} \vDash \Phi}(\vec{x}) W(x)$ sampled with $g(x)$.

$f(x)$



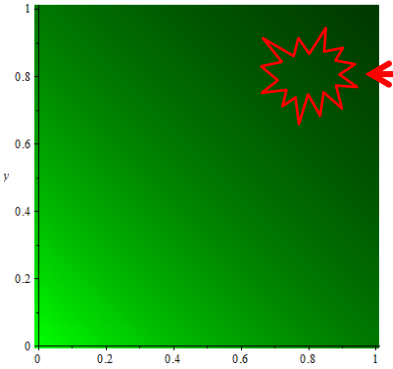
Fault region

$g(x)$



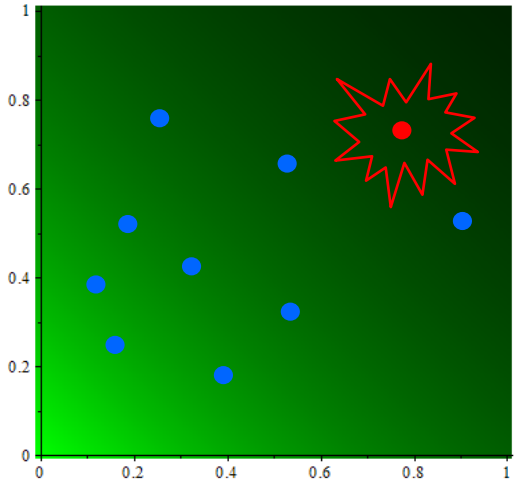
Increased visibility of fault region

$W(x)$



Weight function helps map back to original distr.

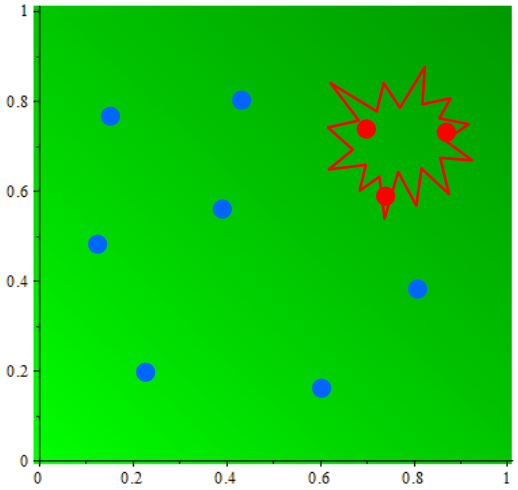
Estimating with Importance Sampling



Basic SMC

- Indicator function $I(\vec{x}) = 1$ iff property holds for input $\vec{x}$.
- Relative Error $RE(\hat{p}) = \frac{\sqrt{var(\hat{p})}}{E[\hat{p}]}$ is measure of accuracy.
- Draw random samples from input distribution $f(\vec{x})$ until target Relative Error is met.
- Estimated probability that property holds is:

$$\hat{p} = \frac{1}{N}\sum_{i=1}^{N} I(\vec{x}_i) = \frac{1}{10} = 0.1 \qquad RE(\hat{p}) = \frac{0.32}{0.1} = 3.2$$



SMC with Importance Sampling

- Modify input distribution to make rare properties more visible.
- Goal is variance reduction.
- Weighting function $W(\vec{x})$ maps solution to original problem.
- Reduced relative error with same number of samples.

$$\hat{p} = \frac{1}{N}\sum_{i=1}^{N} I(\vec{x}_i)W(\vec{x}_i) = \frac{0.2 + 0.5 + 0.3}{10} = 0.1$$

$$RE(\hat{p}) = \frac{0.18}{0.1} = 1.8$$

Calculation of Weighting Function

Suppose we do N test runs of a simulation where each test run involves generating K random variables. If all of the random variables are independent, the weighting function can be written as:

$$W(x_i) = \prod_{j=1}^{K} \frac{f_j(x_{ij})}{g_j(x_{ij})}$$

where x_{ij} is the j^{th} random number generated in the ith simulation run.

Issues:

- We may have simulations where not all random variables are independent.
- If the simulation involves many random variables (i.e. K is large), there is a risk of numerical overflow/underflow in calculating $W(x_i)$ if $g(x)$ is chosen poorly.

Selection of Alternate Distribution

Goal is to reduce variance of estimate

- Not necessarily the same as simply increasing probability sample is in the fault region. Variance of the weight values matter.

An optimal distribution exists (but you must already know answer)

$$g(x) = \frac{I_{\mathcal{M} \vDash \Phi}(\vec{x}) f(\vec{x})}{p}$$

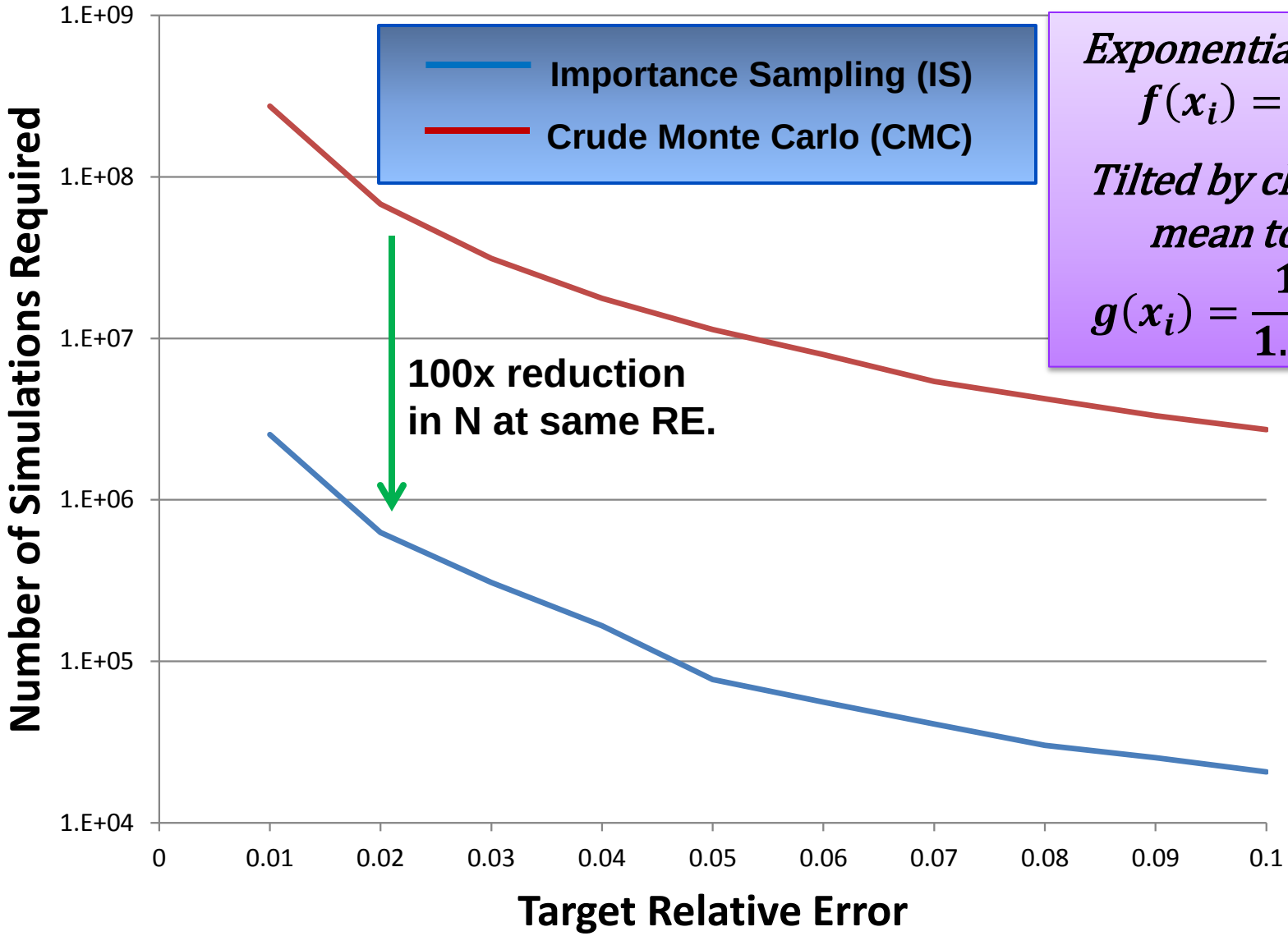
The selected $g(x)$ must have non-zero density wherever $\mathcal{M} \vDash \Phi$ holds.

- If not true, then $W(x) = \frac{f(x)}{g(x)}$ will result in a hidden divide-by-zero and simulation result will be incorrect.

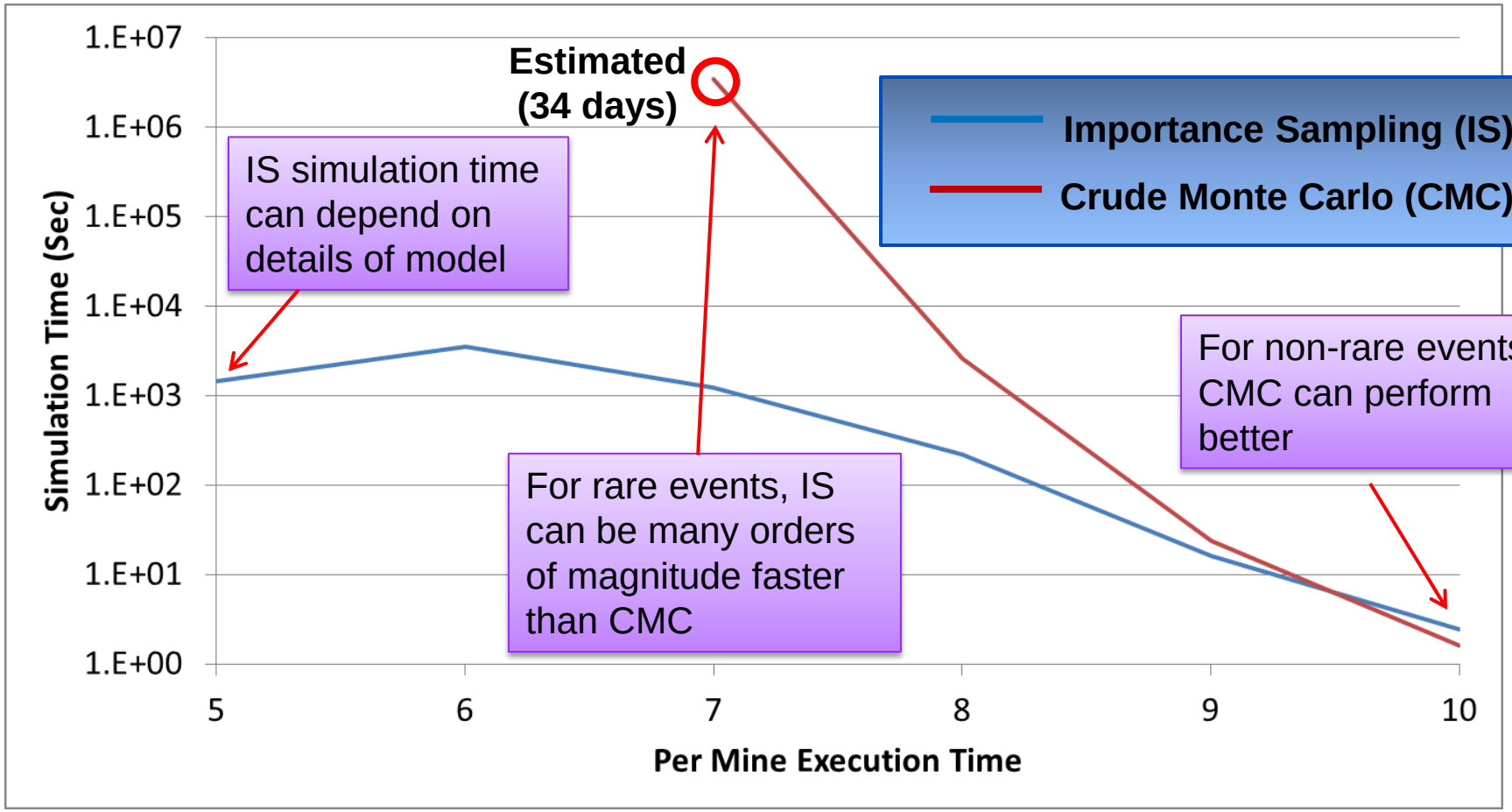
Multiple heuristics exist for optimizing $g(x)$ using probe simulations

- Most methods involve choosing a $g(x)$ in same family as $f(x)$ and optimizing on a distribution parameter (e.g., if $f(x) = e^{-x}$ choose $g(x) = \frac{1}{s} e^{-x/s}$ where s is a "tilt" parameter).
- Methods for choosing optimal tilt parameter include
  - Brute force "sweeping", Cross-entropy method, Non-linear minimization

Simulation Time Reduction (by Relative Error)



Simulation Time Reduction (by Event Rareness)



Conclusion

There is a need for approaches to verify complex time-critical systems in uncertain environments.

- Traditional model checking unfeasible due to model size.
- Black-box testing may not yield reliable results.
- Statistical model checking is one an approach that can address these issues.
  - Formalized approach to expressing failure conditions.
  - Treat simulation runs as Bernoulli trials using Monte-Carlo simulation.
  - Estimates include error bounds (relative error).
  - Applicable to periodic real-time systems by focusing on hyper-period.

Crude Monte-Carlo simulation can be too slow to estimate probability of rare events.

- Importance sampling can dramatically reduce necessary simulation effort.
- Reduction in effort increases as event rareness increases.