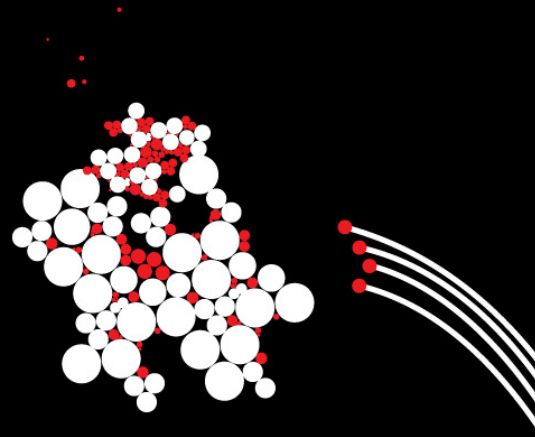
UNIVERSITY OF TWENTE.

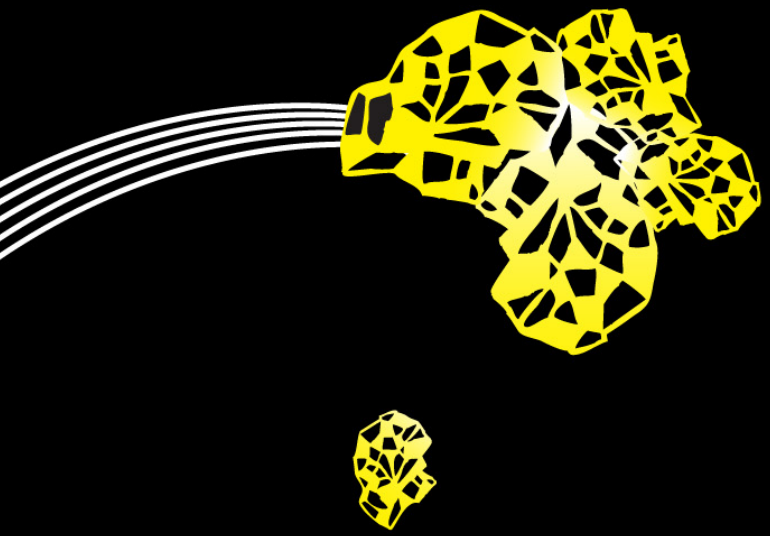


Rare event simulation for dynamic fault trees

Enno Ruijters, Daniël Reijsbergen,
Pieter-Tjerk de Boer, and Mariëlle Stoelinga

University of Twente

13 September 2017

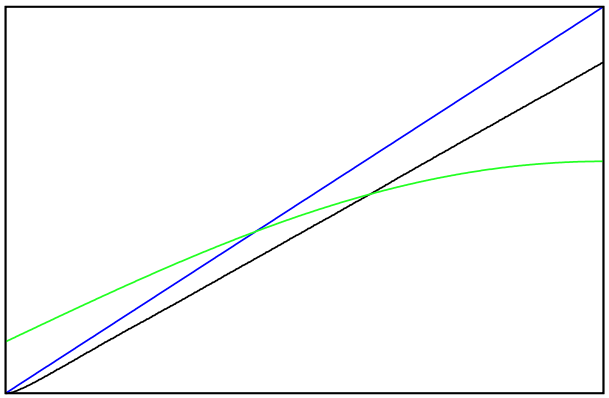


Our contribution in a nutshell

- Many frameworks can provide quantitative dependability analysis.
  - We use dynamic fault trees.
  - Compute system **availability**, reliability, MTTF, etc.
- Complex systems are computationally difficult to analyze:
  - Complex $\rightarrow$ analytic approaches are memory-intensive.
  - Rare failures $\rightarrow$ Monte Carlo simulation requires many samples.
- Our solution: rare event simulation (through importance sampling)
  - Make rare events more likely.
  - Compensate the final result.
  - Automatically.
- **Rare event simulation + dynamic fault trees $\rightarrow$ Faster/more accurate fault tree simulation.**

Dependability analysis

Traditionally two categories:

Analytic	**(Monte Carlo) Simulation**
	
Exact results	Confidence intervals
High memory requirements	Low memory requirements
Infeasible for complex systems	Infeasible for highly reliable systems

Monte carlo simulation

- Draw samples from probability distribution.
- Estimate property of interest (e.g. mean) from samples.
- Example:
  - Spin roulette wheel 1000 times.
  - Observe 36 times green outcome (95% CI boundary).
  - Estimate 3.6% probability of green.
  - (Actual: $\frac{1}{37} = 2.7\%$).
- Drawback: For improbable events, many samples are needed.

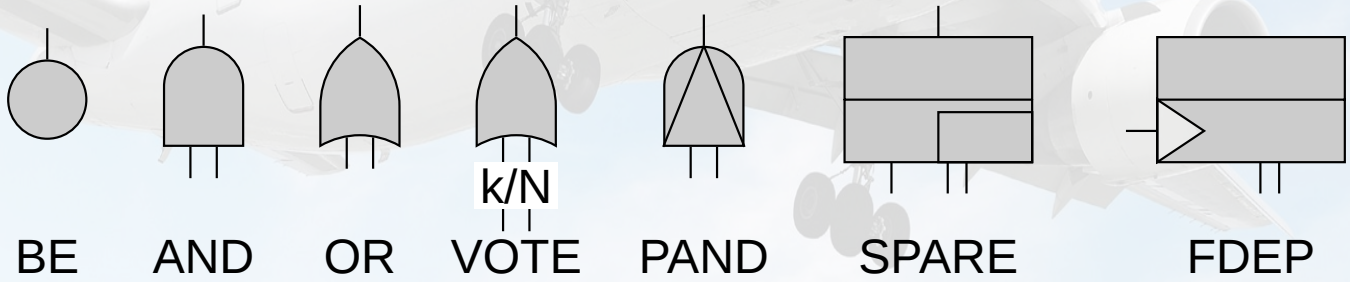
Rare event simulation: Importance sampling

- To reduce required samples: Adjust probabilities and compensate result (Change of Measure).
- Make rare events less rare.
- Example:
  - Spin American roulette wheel 1000 times.
  - Observe 65 times green outcome (95% CI boundary).
  - Estimate 6.5% probability of green **in adjusted system**.
  - (Actual: $\frac{2}{38} = 5.3\%$).
  - Estimate 3.3% probability of green **in original system**.
  - (Actual: $\frac{1}{37} = 2.7\%$).

- Yields more accurate results and/or needs fewer samples.

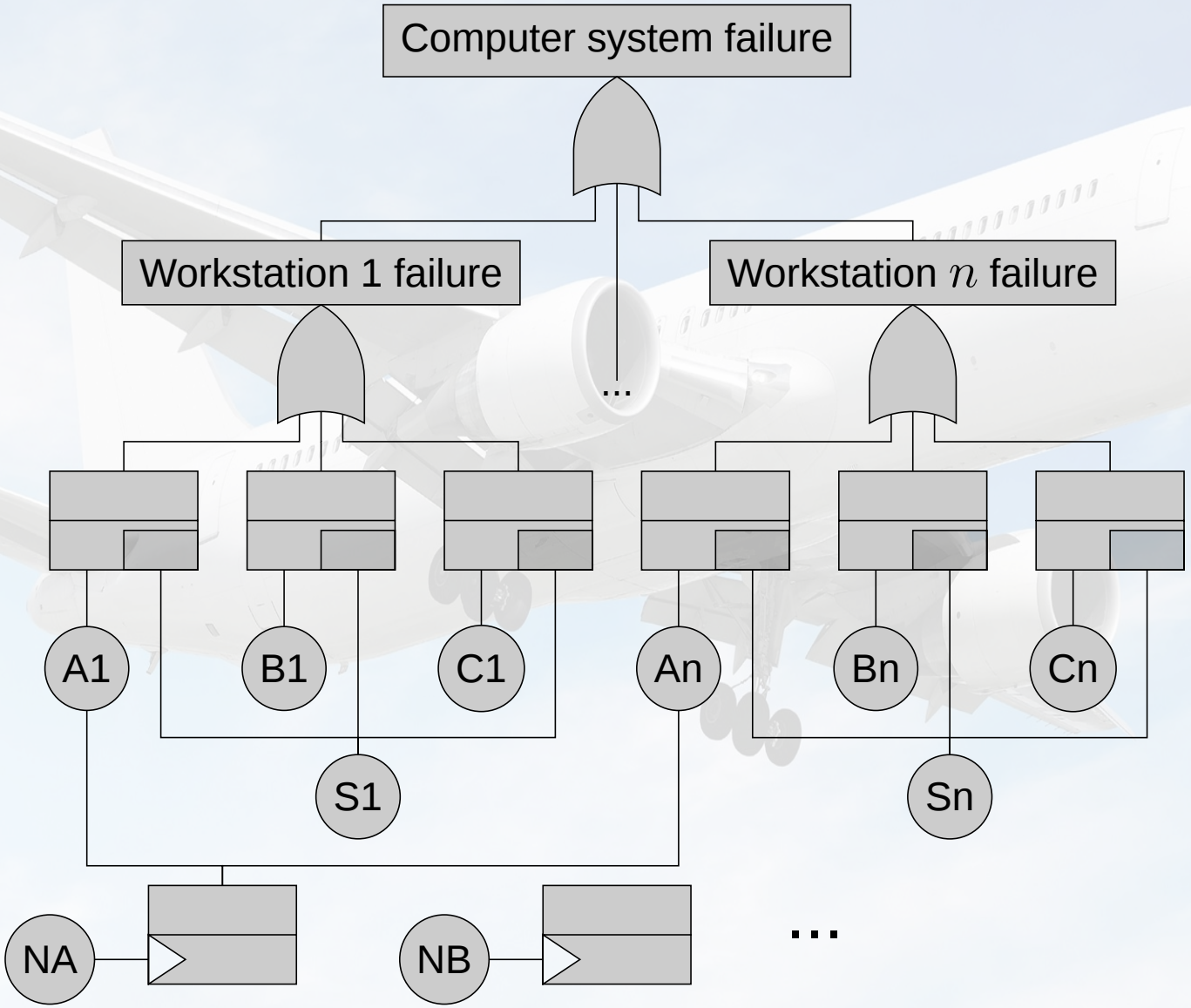
Fault trees

- Industry-standard tool for reliability analysis
- Describe combinations of faults leading to failures
- Root of tree: Top Event; i.e. system failure
- Leaves: Basic Events; i.e. elementary failures and faults
- Nodes: Gates; describe how faults combine



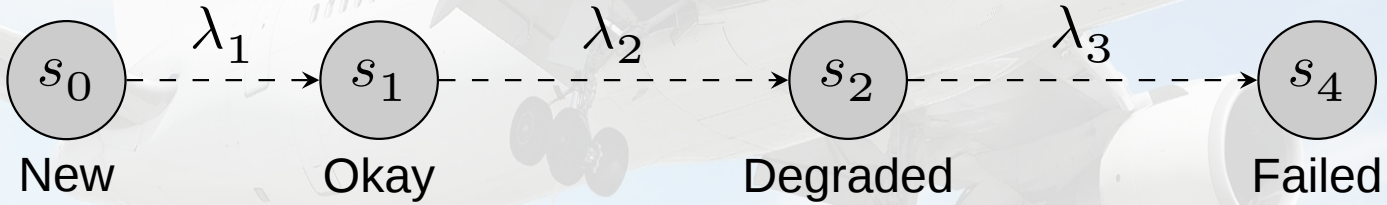
Images of the elements in a dynamic fault tree

DFT example


Computer system failure
Workstation 1 failure
Workstation n failure
...
A1
B1
C1
An
Bn
Cn
S1
Sn
NA
NB
...

Modelling Basic Events

- Degradation modeled in distinct phases.
- Interactive Markov chain:



Rare event simulation (RES)

Multiple techniques to improve Monte Carlo simulations of rare events.

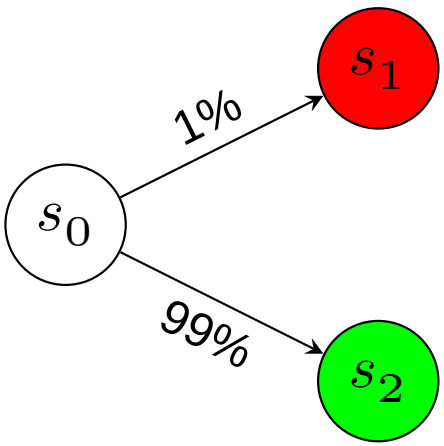
- Importance splitting:
  - Start simulating as usual.
  - Record which runs got ‘close’ to the rare event.
  - Restart multiple simulations from ‘close’ points.
  - Repeat as needed.
  - Estimate measure of interest.
- **Importance sampling:**
  - Alter system to increase likelihood of rare event.
  - Perform simulation runs, recording influence of alterations.
  - Adjust observations to correct for alterations.
  - Estimate measure of interest.

Comparison of RES techniques

Splitting	Sampling
Requires formalization of distance	Requires specification of ‘rare’ transitions
Changes simulation engine	Changes system under simulation
Good for rare events of many steps	Good for rare event of few steps
Limit case: fewer runs needed	Limit case: only one run needed

We use importance sampling as our system reaches the rare event after only a few, low-probability transitions. Such models provide few points to split the samples.

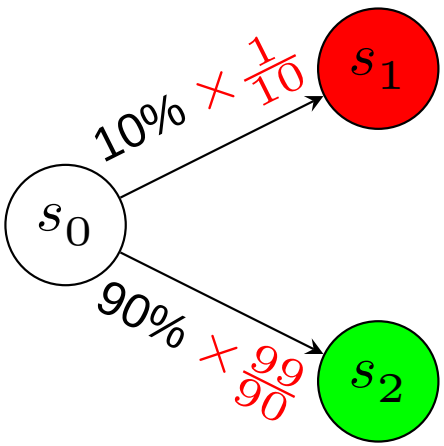
Importance sampling of Markov chains: an example



- Probability of red state?
- Estimate 1 (100 runs): 0%
- Estimate 2 (100 runs): 1%
- Estimate 3 (100 runs): 1%
- Estimate 4 (100 runs): 1%
- Estimate 5 (100 runs): 0%
- Estimate 6 (100 runs): 0%
- Estimate 7 (100 runs): 2%

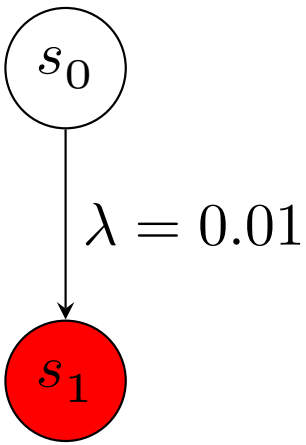
Importance sampling of Markov chains: an example

Make bad state 10 times more likely:



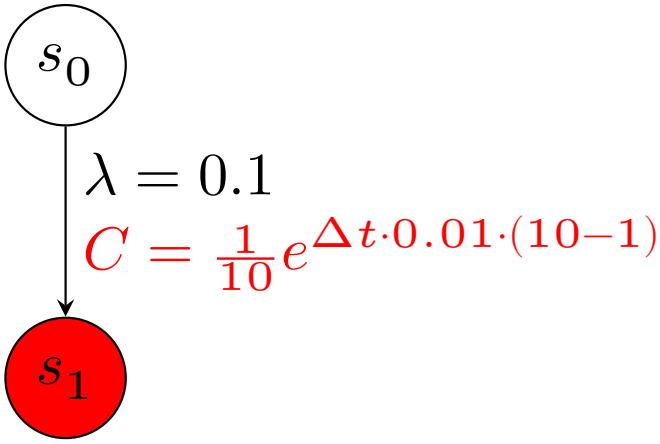
- Probability of red state?
- Estimate 1 (100 runs): $10 \cdot \frac{1}{10} = 1\%$
- Estimate 2 (100 runs): $15 \cdot \frac{1}{10} = 1.5\%$
- Estimate 3 (100 runs): $6 \cdot \frac{1}{10} = 0.6\%$
- Estimate 4 (100 runs): $10 \cdot \frac{1}{10} = 1\%$
- Estimate 5 (100 runs): $13 \cdot \frac{1}{10} = 1.3\%$
- Estimate 6 (100 runs): $12 \cdot \frac{1}{10} = 1.2\%$
- Estimate 7 (100 runs): $14 \cdot \frac{1}{10} = 1.4\%$

Importance sampling of Markov chains: continuous time



- Probability of reaching red state within 1 time unit?
  - Actual = $1 - e^{-0.01} \approx 0.995\%$.
- Estimate 1 (100 runs): 0%
- Estimate 2 (100 runs): 1%
- Estimate 3 (100 runs): 2%
- Estimate 4 (100 runs): 1%
- Estimate 5 (100 runs): 0%
- Estimate 6 (100 runs): 0%
- Estimate 7 (100 runs): 0%

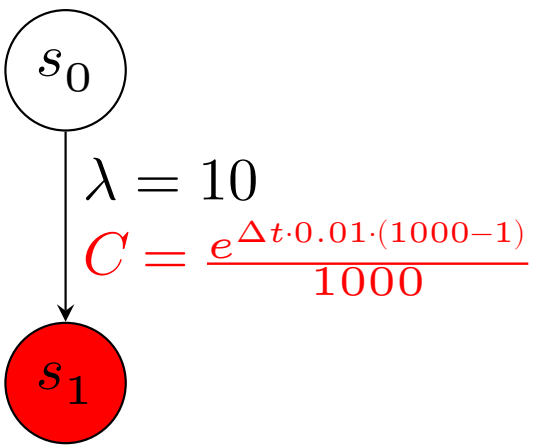
Importance sampling of Markov chains: continuous time



- Probability of reaching red state within 1 time unit?
- Estimate 1 (100 runs): 0.42%
- Estimate 2 (100 runs): 1.27%
- Estimate 3 (100 runs): 0.73%
- Estimate 4 (100 runs): 0.41%
- Estimate 5 (100 runs): 1.15%
- Estimate 6 (100 runs): 1.16%
- Estimate 7 (100 runs): 0.92%

Importance sampling of Markov chains: overcompensating

Increasing the rare event too much:



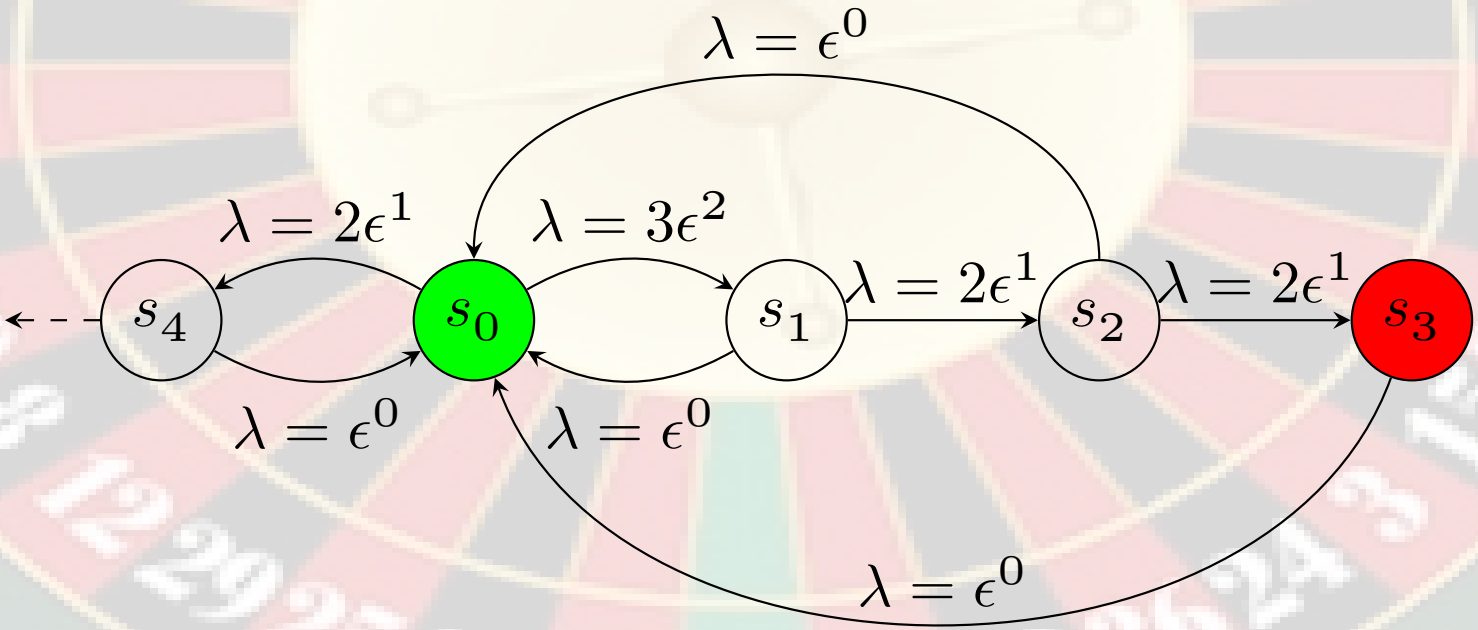
- Probability of reaching red state within 1 time unit?
- Estimate 1 (100 runs): 0.64%
- Estimate 2 (100 runs): 0.45%
- Estimate 3 (100 runs): 0.85%
- Estimate 4 (100 runs): 0.86%
- Estimate 5 (100 runs): 0.87%
- Estimate 6 (100 runs): 11.26%
- Estimate 7 (100 runs): 0.67%

Path-ZVA algorithm

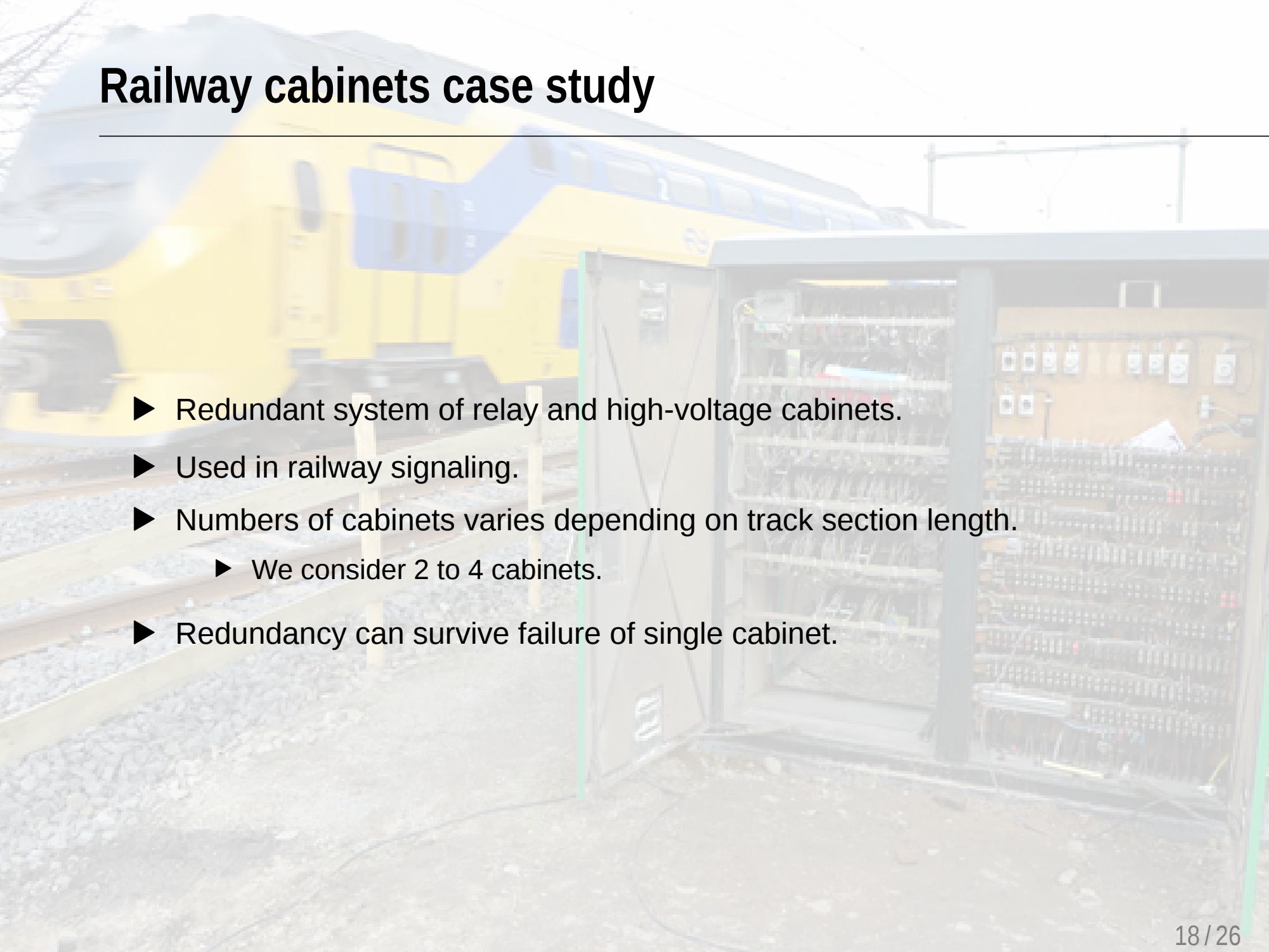
- Importance sampling algorithm for Markovian models.
- Divides states into three categories:
  - ‘Perfect’ states reached frequently.
  - ‘Bad’ states reached rarely.
  - ‘Connecting’ states inbetween.
- Estimates:
  - Probability of reaching ‘bad’ states before returning to ‘perfect’ states.
  - Fraction of time spend in ‘bad’ states.
- Transition rates parameterized as $r \cdot \epsilon^n$ with $0 < \epsilon << 1$ to indicate ‘rareness’.

Applying Path-ZVA to DFTs

- Basic idea: Compute state space on-the-fly.
- Path-ZVA stores the subset of states in dominant paths.
- All other states only generated as reached, and not stored.



Railway cabinets case study



- Redundant system of relay and high-voltage cabinets.
- Used in railway signaling.
- Numbers of cabinets varies depending on track section length.
  - We consider 2 to 4 cabinets.
- Redundancy can survive failure of single cabinet.

Results: Accuracy

Exact result for DFTCalc, 95% confidence for others:

	N	P	Unavailability DFTCalc	FTRES	MC
Railway cabinets	2	1	$4.25685 \cdot 10^{-4}$	$[4.256; 4.258] \cdot 10^{-4}$	$[4.239; 4.280] \cdot 10^{-4}$
	3	1	$7.71576 \cdot 10^{-4}$	$[7.713; 7.716] \cdot 10^{-4}$	$[7.694; 7.751] \cdot 10^{-4}$
	4	1	$1.99929 \cdot 10^{-3}$	$[1.998; 2.000] \cdot 10^{-3}$	$[1.999; 2.004] \cdot 10^{-4}$
	2	2	$4.55131 \cdot 10^{-8}$	$[4.548; 4.555] \cdot 10^{-8}$	$[1.632; 4.387] \cdot 10^{-8}$
	3	2	$6.86125 \cdot 10^{-8}$	$[6.846; 6.873] \cdot 10^{-8}$	$[0.673; 1.304] \cdot 10^{-7}$
	4	2	–	$[2.358; 2.394] \cdot 10^{-7}$	$[2.282; 3.484] \cdot 10^{-7}$
	2	3	$5.97575 \cdot 10^{-12}$	$[5.714; 6.252] \cdot 10^{-12}$	–
	3	3	–	$[5.724; 7.914] \cdot 10^{-12}$	–
	4	3	–	$[0.337; 1.871] \cdot 10^{-11}$	–

Literature case studies

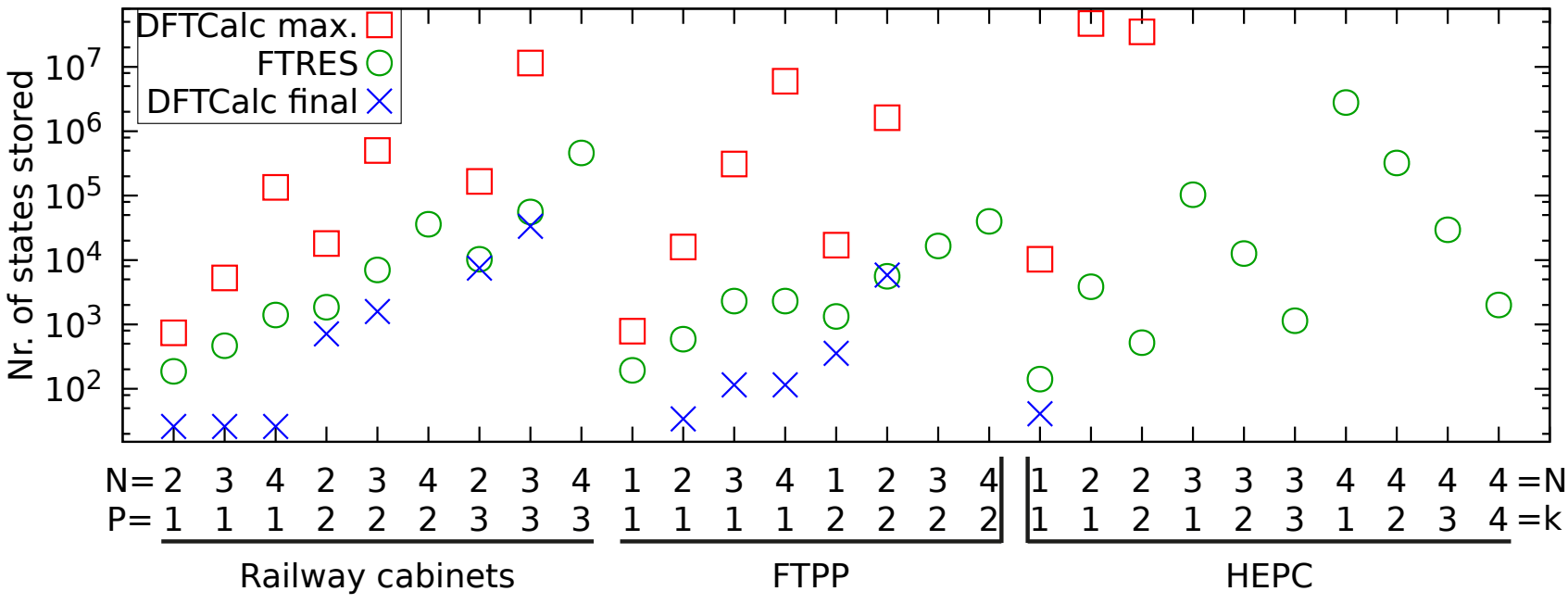
- Fault Tolerant Parallel Processor:
  - Literature case study, birth of DFTs
  - Network of workstations interconnected with network elements.
  - Every workstation have multiple processors, with one spare.
  - Vary number of processor groups 1 – 4.
- Hypothetical example computer system:
  - Literature case study, NASA handbook of (D)FTs
  - Computer with redundant processors, memory, and buses.
  - Hardware and software components of operator interface.
  - Entire system replicated N times, of which k must remain operational.

Results: Accuracy

Exact result for DFTCalc, 95% confidence for others:

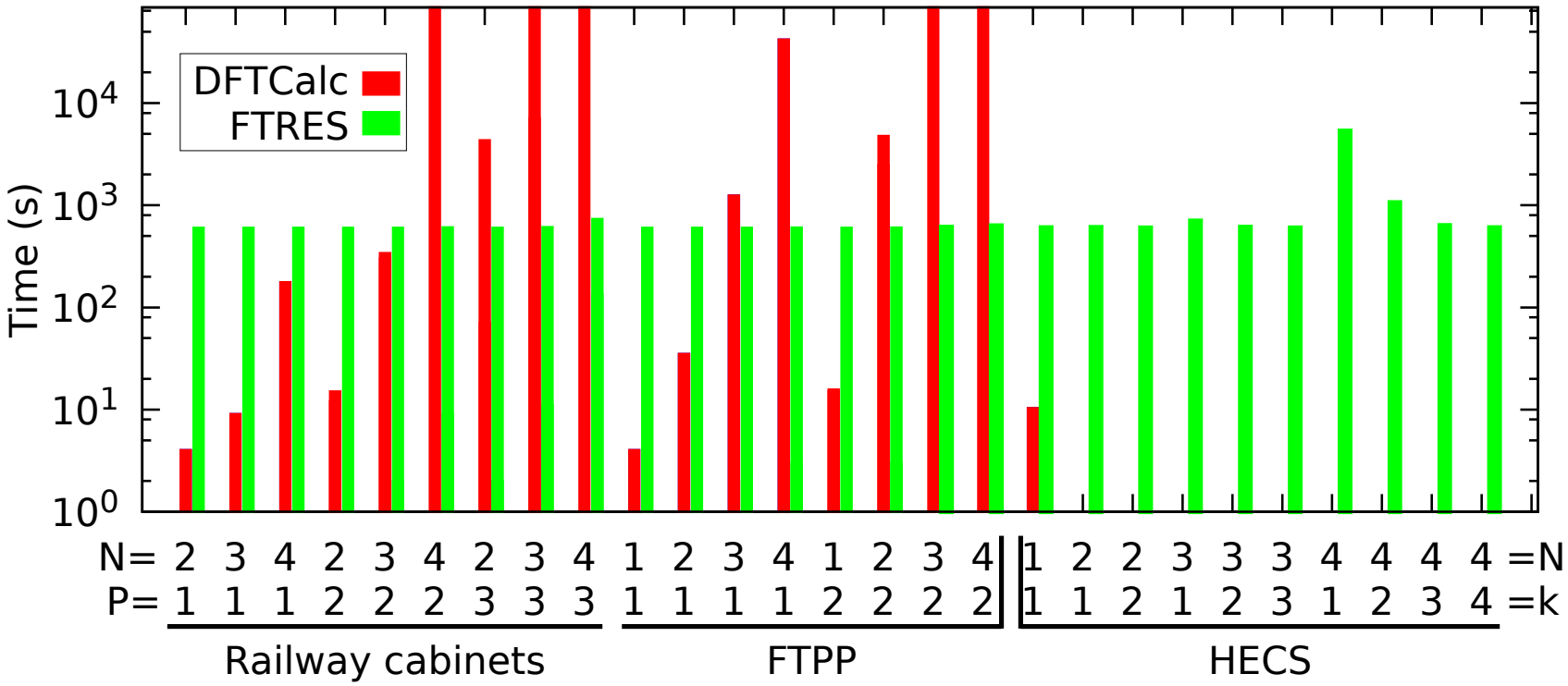
			Unavailability		
	N	P	DFTCalc	FTRES	MC
FTPP	1	1	$2.18303 \cdot 10^{-10}$	$[2.182; 2.184] \cdot 10^{-10}$	–
	4	1	$2.22979 \cdot 10^{-10}$	$[2.229; 2.230] \cdot 10^{-10}$	$[0; 2.140] \cdot 10^{-8}$
	1	2	$1.76174 \cdot 10^{-20}$	$[1.761; 1.763] \cdot 10^{-20}$	–
	4	2	–	$[1.257; 2.553] \cdot 10^{-20}$	–
	N	**k**	**DFTCalc**	**FTRES**	**MC**
HECS	1	1	$4.12485 \cdot 10^{-5}$	$[4.118; 4.149] \cdot 10^{-5}$	$[2.615; 10.64] \cdot 10^{-5}$
	2	1	–	$[3.010; 3.061] \cdot 10^{-9}$	–
	2	2	–	$[8.230; 8.359] \cdot 10^{-5}$	$[0; 1.734] \cdot 10^{-4}$
	4	1	–	$[1.328; 8.213] \cdot 10^{-17}$	–
	4	2	–	$[1.145; 1.270] \cdot 10^{-12}$	–
	4	3	–	$[1.744; 1.817] \cdot 10^{-8}$	–
	4	4	–	$[1.609; 1.667] \cdot 10^{-4}$	–

Overall results: State space



- FTRES always below DFTCalc maximal state space size.
- FTRES computes results where DFTCalc does not.

Overall results: Speed



- FTRes and MC spend a constant 5 mins. simulating.
- Simulation time mostly dominates state-space exploration.
- Almost all DFTCalc experiments for HECS ran out of memory.

Conclusions

- New (in fact, first) method applying rare event simulation to dynamic fault trees.
- On-the-fly composition reduces state-space storage.
- Importance sampling provides tight confidence intervals even for very rare events.
- We handle larger models than DFTCalc, and models of more reliable systems than Monte Carlo simulation.

Future work

- Measures beyond availability (e.g. reliability).
- Non-Markovian timing of e.g. maintenance actions.
- On-the-fly reduction of equivalent states.
- Nondeterminism to cover full set of DFTs.

Thank you for your attention.

Questions?