UNIVERSITÀ DEGLI STUDI FIRENZE
Da un secolo, oltre.

HR EXCELLENCE IN RESEARCH

Adaptive Importance Sampling for Rare Event Simulation in Fault Tree Analysis

Candidate: Mattia Manneschi

Introduction

Goal: Estimate failure probability of complex systems (fault trees) where failures are rare events.

Problems with standard Montecarlo

- requires millions of simulations;
- most samples yield no failures;
- computationally expensive;
- high variance for rare events;

Solution: Adaptive Importance Sampling

- Modify simulation parameters ($\lambda \to \alpha\lambda$, $\mu \to \beta\mu$) to observe more failures
- Correct bias with likelihood ratio
- **Challenge:** How to find optimal α, β? GNN + MLP models.

System Architecture

Our approach:

- Generation of random fault trees (random number of components and ports);
- Self-supervised GNN predicts optimal α, β ranges for topology and period T;
- Self-supervised GNN predicts optimal samples number for IS and MC probability estimation;
- MLP + Cross-Entropy Method optimizes α, β within predicted range;
- CTMC Simulator: simulates with Importance Sampling.

Fault tree Generation

Random Topology Generator:

- 2 to 45 components (scalable);
- Gate types: AND, OR, KooN;
- Each gate type has a different probability to be chosen;
- Hierarchical bottom-up construction;
- $\lambda \in$ [10^-5, 10^-4], $\mu \in$ [0.1, 1.0];
- Used both in training and validation;

Topology determines Probability:

- Many OR gates → Higher P (multiple failure paths)
- Many AND gates → Lower P (all must fail)
- AND/OR ratio determines event rarity

IS Parameter Ranges Prediction

Key Technical Components

- **Graph Processing:** 3-Layer **GCNConv** stack extracts hierarchical features from gates (AND/OR) and components.
- **Temporal Context:** Injects Tnorm and 1/(1+Tnorm) into the latent space to enable time-adaptive biasing.
- **Global Feature Fusion:** Concatenates node embeddings with system-wide metrics (depth, Ncomp, avg λ/μ).

Execution Flow

- **Input:** Graph Data [λ,μ,Type,Tnorm]
- **Logic:** Message Passing → Global Pooling → Feature Concatenation.
- **Output:** Four-dimensional vector [αmin,αmax,βmin,βmax].

Adaptive Sample Size Optimization

Architecture Highlights

- **GNN Core:** A 2-layer **GCNConv** network that generates topological embeddings from node features (λ,μ,Type).
- **Global Context Integration:** Fuses graph embeddings with system-wide metadata: component counts, gate distribution, and tree depth.
- **Logarithmic Regression:** Predicts log10(N) using a **Sigmoid** activation scaled to a specific range.

Key Strategic Advantages

- **Adaptive Effort:** Automatically allocates more samples to rare-event scenarios (deep AND gates) and fewer to high-probability systems.
- **Exploration-Exploitation:** Gaussian sampling during training, ensuring the model discovers the minimum N required to achieve variance stability.

Self-supervised Training Loop

The system employs a **closed-loop feedback mechanism** where both Predictors (Range & Sample) are trained simultaneously without human-labeled data.

- **Step 1: Policy Prediction**
  - **RangePredictor** sets the "How" (biasing intensity α,β).
  - **SamplePredictor** sets the "How Much" (computational effort N).
- **Step 2: Simulation Execution**
  - The system runs parallel **CTMC simulations** using the predicted parameters on randomly generated Fault Trees.
- **Step 3: Statistical Feedback (The Reward)**
  - Performance is measured via the **Coefficient of Variation (CV)** and convergence speed.
  - A custom **Penalty Function** ensures physical consistency (e.g., preventing β from diverging).
- **Step 4: Gradient Update**
  - The errors are backpropagated through the GNNs to refine future predictions.

Self-supervised Training Loop

The Challenge of Complexity

Directly training a GNN on systems with 30+ components is prone to instability due to the massive state space and extreme rarity of events.

The Incremental Fine-Tuning Approach

We implemented a **Curriculum Learning** strategy to progressively build the model's "topological intuition":

- **(Base):** Training on small systems (**2-10 components**) to learn basic AND/OR logic and fundamental biasing.
- **(Mid-Scale):** Fine-tuning the base model on medium systems (**10-20 components**) to adapt to deeper hierarchies.
- **(Large-Scale):** Final optimization on complex systems (**20-30+ components**) to master high-dimensional variance reduction.

UNIVERSITÀ DEGLI STUDI FIRENZE
Da un secolo, oltre.

HR EXCELLENCE IN RESEARCH

Self-supervised Training Loop

Key Advantages

- **Faster Convergence:** The model starts with weights already optimized for simpler versions of the same problem.
- **Stability:** Prevents the "Gradient Explosion" or "Vanishing Rewards" typical of training on extremely rare events from scratch.
- **Knowledge Transfer:** Proves that the GNN learns generalized topological rules that scale across different system magnitudes.

Parameter Refinement – MLP + Cross-Entropy

Concept: Local Policy Optimization

While the GNN provides a global starting point, a dedicated **AlphaBetaMLP** performs fine-grained parallelized (16 workers) optimization for the specific Fault Tree and mission time T.

The CEM Workflow:

- **Stochastic Probing:** The MLP predicts the *mean* of a distribution; we sample N candidate configurations $[\alpha,\beta]_i$ using a Gaussian policy $N(\mu,\sigma)$.
- **Elite Selection (ρ-sampling):**
    - Runs N trajectories for each candidate.
    - Identifies the **Top ρ% (Elite)** candidates that yielded the highest variance reduction.
- **Softmax Weighting:** Elite samples are weighted using **Log-Sum-Exp** to stabilize training against extreme rare-event values ($<10^{-9}$).

Parameter Refinement – MLP + Cross-Entropy

Optimization & Convergence

- **Policy Gradient Loss:** Updates the MLP to shift the sampling distribution toward the "Elite" parameters.
- **Entropy Bonus:** Penalizes premature convergence, forcing the model to explore the parameter space thoroughly.
- **Dynamic Refinement:** As T increases, the loop adaptively converges to the most efficient biasing strategy.

CTMC Simulation

Objective

To model the dynamic evolution of the Fault Tree and apply **Importance Sampling (IS)** at the trajectory level.

The system state evolves as a Continuous-Time Markov Chain, where components transition between "Working" and "Failed".

- **Failure Acceleration (α):** Failure rates are scaled by $\lambda\_IS=\lambda \cdot \alpha$, making rare failures more frequent.
- **Repair Deceleration (β):** Repair rates are scaled by $\mu\_IS=\mu \cdot \beta$, keeping the system in a failed state longer.

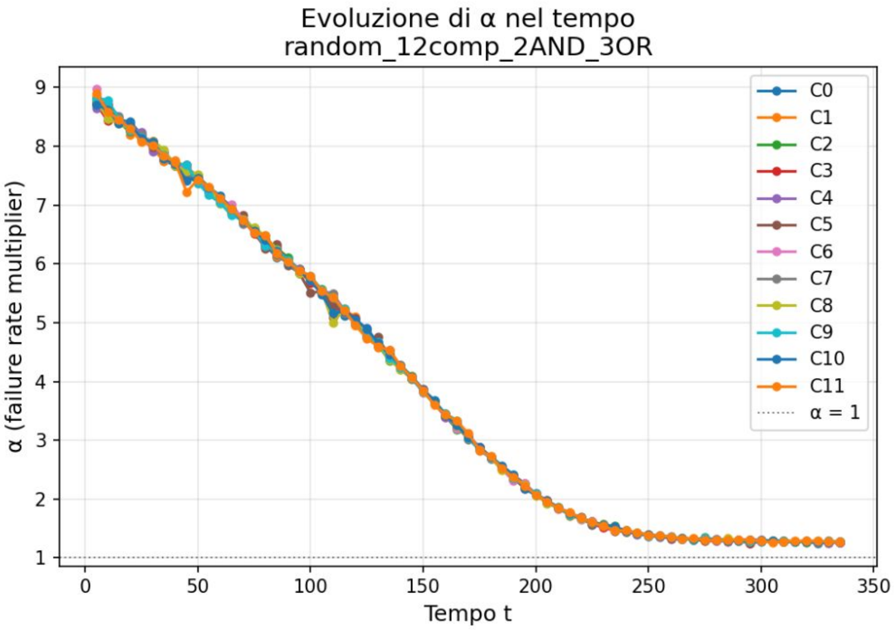
CTMC Simulation

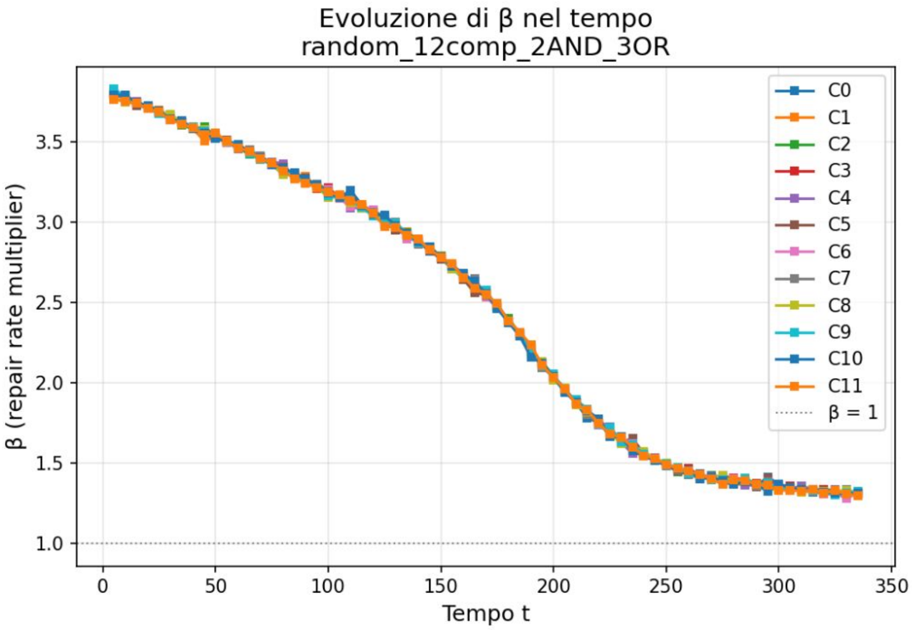
Likelihood Ratio Accumulation

To ensure the final estimate is unbiased, we maintain a **Log-Weight (log_w)** for every trajectory, updated at each step:

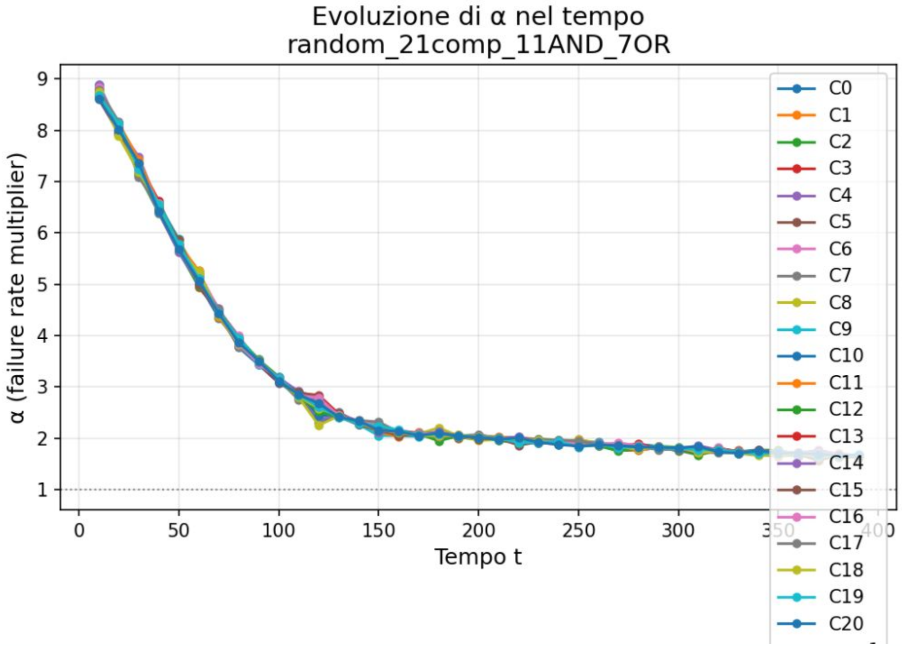
1. **Holding Time Correction:** Adjusts for the biased time spent in each state: (R_IS−R_orig) · dt;
2. **Transition Correction:** Adjusts for the biased choice of which component fails or is repaired next: log(rate_orig/rate_IS);

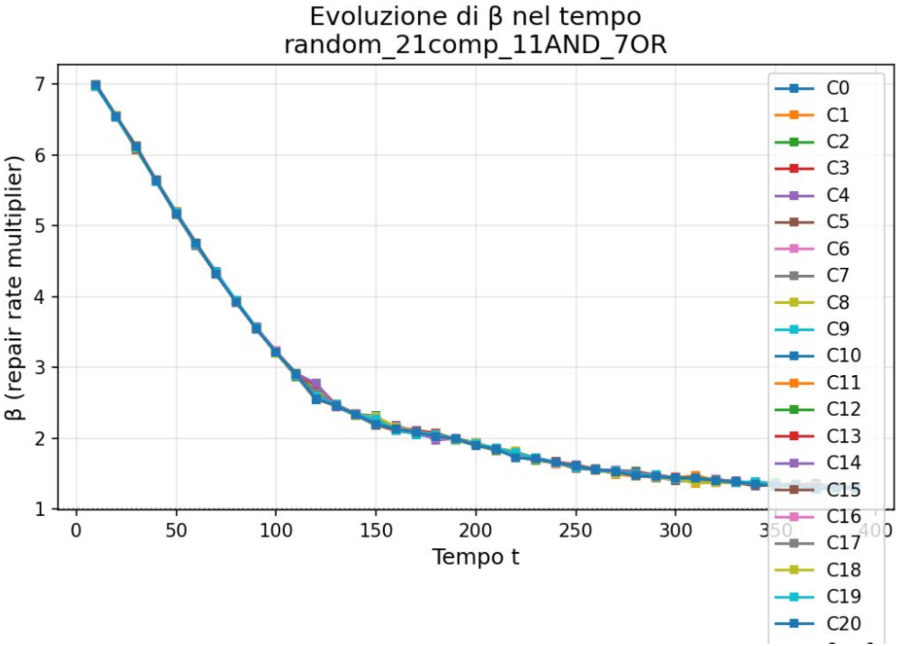
Results – Biasing Policy Dynamics



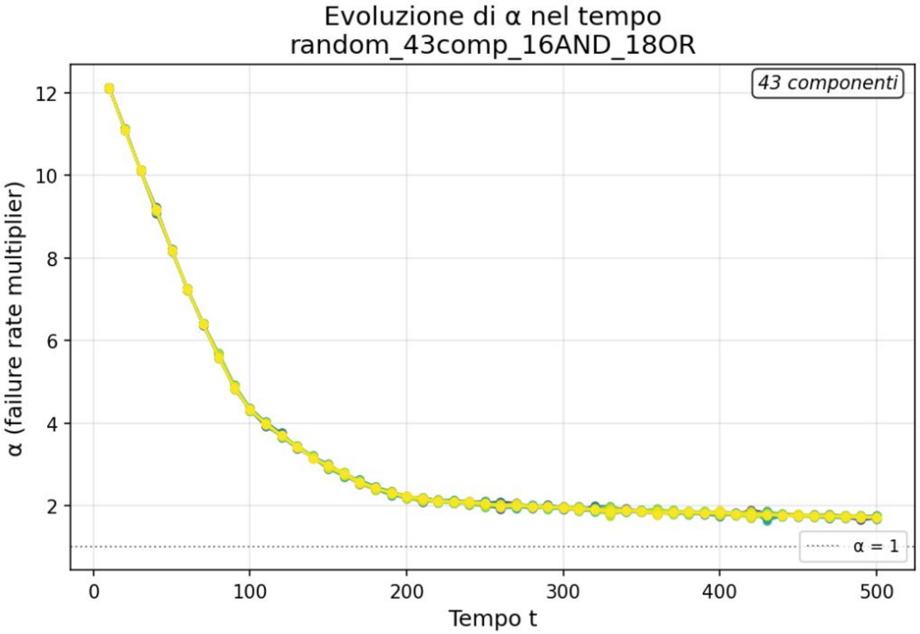


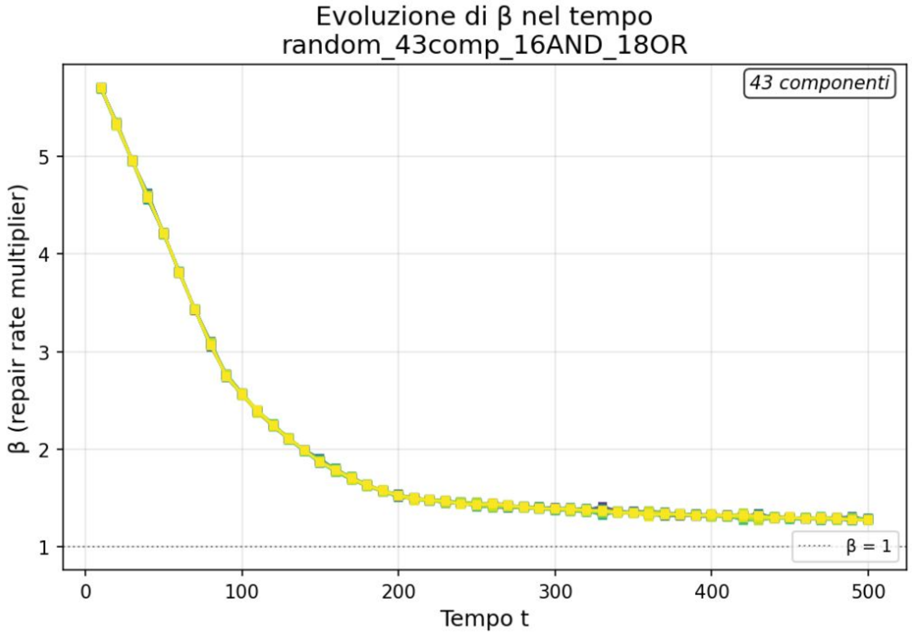
Results – Biasing Policy Dynamics





Results – Biasing Policy Dynamics

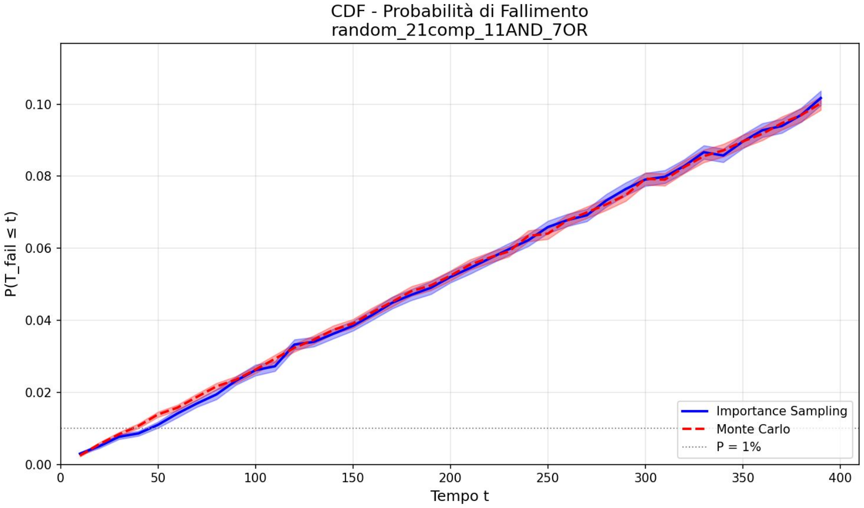


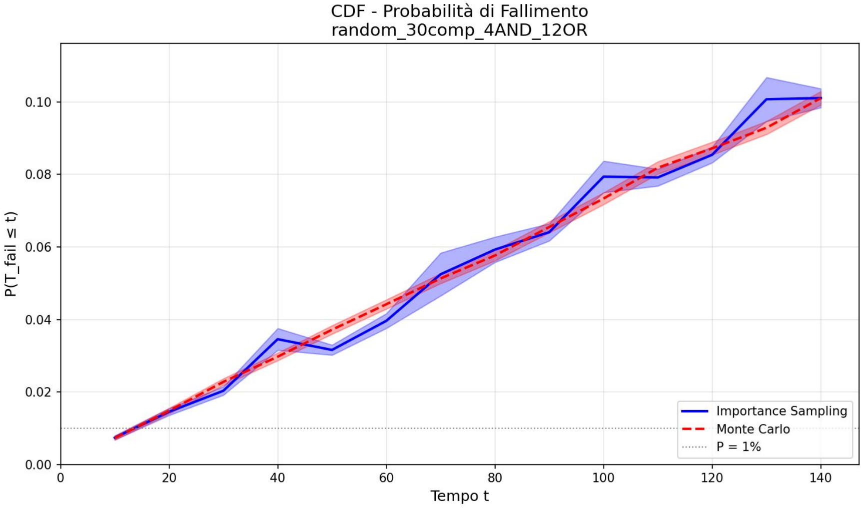


UNIVERSITÀ DEGLI STUDI FIRENZE
Da un secolo, oltre.

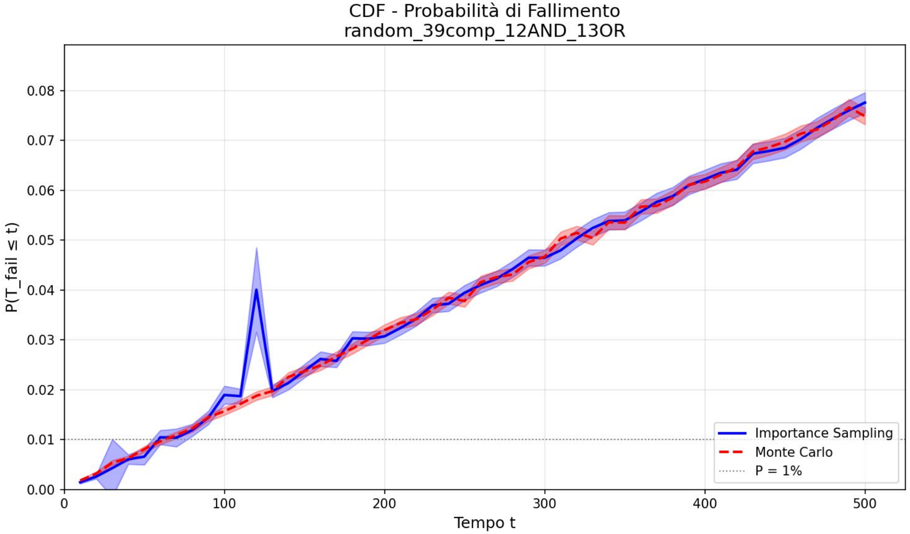
HR EXCELLENCE IN RESEARCH

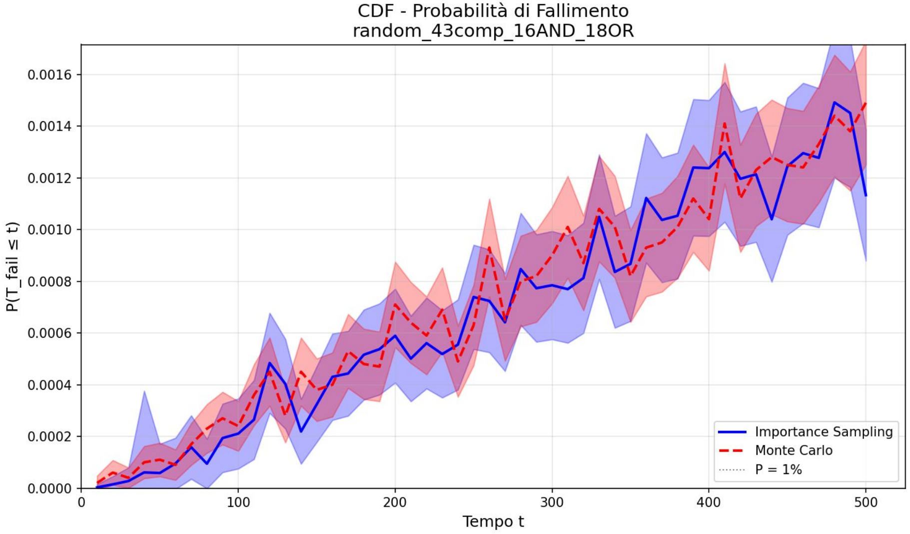
Results – CDF Estimation





Results – CDF Estimation





Challenges & Computational Bottlenecks

The MLP-CEM Bottleneck

Despite the high precision achieved, the local refinement phase (MLP + CEM) presents significant computational challenges:

- **Training Overhead:** While GNN inference is nearly instantaneous, training the MLP via CEM for a specific Fault Tree can take **several hours**, even for probabilities around 10^{-7}.
- **Simulation Intensity:** The "Stochastic Probing" requires thousands of CTMC trajectories per epoch to provide a stable gradient, making the optimization loop the primary bottleneck.
- **Cost-Benefit Trade-off:** Currently, the time saved in the final simulation is partially offset by the time required for parameter tuning on a single, specific system.
- **High variance for extremely rare events:** in order to keep computational costs affordable, the number of samples and trajectories per epoch has been kept quite small, leading to a non negligible variance.

Future Work

- **Amortized Optimization**
  - **Direct Parameter Prediction:** Instead of using the MLP-CEM loop for every new tree, the goal is to further train the **GNN** to predict the *refined* α and β directly.
  - **Meta-Learning:** Implementing a "Learn to Learn" approach where the GNN internalizes the optimization strategy of the CEM, eliminating the need for local training.
- **Advanced Architectures**
  - **GAT & Transformers:** Moving from GCN to **Graph Attention Networks** to automatically weigh the importance of specific failure paths without manual ρ factor tuning.
  - **Large-Scale Stress Tests:** Extending the framework to industrial-grade systems with **100+ components** and more complex dynamic dependencies.
- **Hybrid Engines**
  - Integration with **GPU-accelerated CTMC engines** to reduce the time per trajectory.
- **Component-specific biasing:**
  - adjust biasing parameters with respect to the component criticality.