Learning-Based Minimization of Packet Loss in Time-Sensitive Networking

Feyzullah Kara[1,2] and Figen Oztoprak[1]

[1] Gebze Technical University, 41400, Kocaeli, Turkey
[2] Aselsan, 06300, Ankara, Turkey feykara@aselsan.com

Abstract. Time-Sensitive Networking (TSN) standards introduce deterministic capabilities to Ethernet for safety-critical applications. However, optimizing the high-dimensional TSN parameter space while accounting for stochastic hardware imperfections remains a significant challenge. This paper proposes a learning-based robust optimization framework for TSN configuration. We introduce an adaptive surrogate modeling approach leveraging active learning to approximate network behavior across operational regimes of progressive severity. Surrogate models, specifically gradient-boosting and random forest, are trained using realistic simulation data. Solutions to the robust optimization model are obtained via solving a mixed-integer programming (MIP) formulation obtained by the replacing the black-box components with MIP-representable learned surrogates. Our initial tests demonstrate the performance of the proposed framework in effectively computing robust network configurations.

Keywords: Learning-based optimization · Active learning · Time-sensitive networking · Packet loss minimization · Surrogate modeling · Mixed-Integer Programming.

1 Introduction

Ethernet, standardized as IEEE 802.3, has evolved from a best-effort delivery protocol into the ubiquitous backbone of modern digital communication. While robust, the protocol was originally designed for probabilistic media access rather than deterministic control [9]. However, modern safety-critical systems, such as autonomous driving and industrial automation, require strict latency and reliability guaranties. To bridge this gap, the IEEE 802.1 working group introduced Time-Sensitive Networking (TSN), a suite of standards designed to add deterministic guaranties to Ethernet networks [12]. Key mechanisms such as the Time-Aware Shaper (IEEE 802.1Qbv) for egress scheduling, Per-Stream Filtering and Policing (IEEE 802.1Qci) for ingress protection, and Precision Time Protocol (IEEE 802.1AS) for synchronization provide the necessary tools for managing latency and jitter.

Despite these advancements, the deployment of TSN in complex topologies introduces a hyper-dimensional configuration space. Network architects must

tune a vast array of parameters—ranging from Gate Control Lists (GCL) to policing bandwidths—while contending with the physical reality of network hardware. Variations in bandwidth availability, micro-latencies in switching fabrics, and packet delay variation (jitter) are inherent to switched networks. Furthermore, hardware-level imperfections, such as Bit Error Rates (BER) leading to CRC errors or thermal throttling causing processing latency variations, introduce stochastic noise that can disrupt control loops. Adjustment of these configuration parameters is traditionally approached through two main avenues: analytical methods like network calculus, which offer worst-case bounds but are often overly pessimistic [11], and discrete-event simulations, which provide high fidelity information on the system performance but are computationally expensive for iterative design loops.

Motivations. Configuration of Ethernet communication in complex topologies is an optimization problem involving significant uncertainties. Managing mixed-criticality traffic requires addressing both controllable TSN parameters and stochastic environmental noise. While vital for safety-critical applications such as autonomous driving, existing work on this problem remain scarce.

This study aims to propose a methodology for the computation of robust solutions to the optimal TSN configuration problem. We work on a robust optimization formulation of the problem as that would enable solutions that are less sensitive to the realizations of the uncertain parameters[14]. Inspired by recent advancements on embedding machine learning models into optimization formulations[2], we consider employing machine learning surrogates to replace the black-box objective of the optimization problem which can be computed via expensive simulation runs. The use of machine learning surrogates not only eliminates the computational cost of simulations during the optimization phase, but also allows solving the problem as a mixed integer program (MIP) as we elaborate in subsequent parts of this paper.

Another idea motivating the methodology proposed in this work is the use of expert knowledge in the process of surrogate learning. As the data used to train such a surrogate model is obtained via simulation runs, the efficiency of the learning process as well as the success of the learned model relies on sampling. We conjecture that domain expertise can help guiding the sampling in this particular application.

Similar to our approach, Bezerra et al. [3] proposes a machine learning-based optimization framework for minimizing end-to-end (E2E) latency in TSN. Using the NeSTiNg simulator, they generate a dataset of over 10,000 network configurations, which they then use to train surrogate models to predict E2E latency. Furthermore, they employ evolutionary algorithms, specifically NSGA-II [6] and SMPSO [16], to optimize the network configuration using their surrogate models. Their work validates the feasibility of replacing expensive discrete-event simulations with fast-executing ML surrogates for optimization.

Although [3] optimizes egress traffic shaping primarily using deterministic parameters, our model extends this scope by integrating ingress policing (IEEE 802.1Qci) and incorporating stochastic hardware imperfections, such as jitter and

CRC errors, to reflect realistic network instability. Furthermore, the selection of the PTP synchronization message frequency within the configuration parameters creates a critical trade-off between minimizing clock drift and managing link congestion. As highlighted in [4], while higher message rates significantly improve frequency accuracy, they impose a penalty on bandwidth consumption. Therefore, the explicit integration of PTP dynamics into the simulation environment strengthens the robustness of the end-to-end TSN architectural model. We propose a completely different methodology for a different optimization problem formulation, motivated by different ideas both for learning and optimization components as mentioned above.

Contributions. This paper presents a comprehensive framework for learning-based robust optimization of TSN configuration. The main contributions are as follows.

- *High-fidelity simulation environment.* We develop a rigorous OMNeT++ simulation framework that accounts for mandatory Ethernet overheads (Preamble, SFD, IFG) and stochastic hardware imperfections (jitter, CRC errors), ensuring physically valid data generation across *stable*, *stress*, and *crisis* operational regimes.
- *Adaptive surrogate learning exploiting domain expertise.* We introduce a composite stratified sampling strategy that guides the training of ensemble learning models. By leveraging domain knowledge, we intentionally seed the training set with baseline stable configurations and progressively introduce worst-case parameter combinations –such as high traffic loads coupled with minimal packet sizes– to expose edge-case failures. This approach ensures that the model captures critical failure modes often missed by uniform sampling.
- *Explanation of packet loss via the surrogate model.* As we build the surrogate models via active learning and aim at making them representative at random input points (unlike the focused infill strategies of conventional simulation optimization), we are able to use these surrogates to analyze the effect of individual configuration parameters on the reliability of the network in terms of the packet loss.
- *Optimization via mixed integer programming.* We compute robust solutions to the packet loss minimization problem via solving a mixed-integer program by using a MIP formulation of the trained random forest regressor. This allows global optimization of the approximate problem obtained by replacing the original black-box objective with its learned surrogate.

We should note that the problem studied in this paper is a robust simulation optimization problem[7]. Surrogate models both for uncertainty quantification and for approximation of the objective/constraint functions have already been suggested in the literature[5,7,10]. However, these surrogates are typically highly nonconvex and solved via continuous local or global optimization techniques.

2 Proposed Approach

2.1 Robust Optimization Model

We formulate the packet loss minimization problem as

$$\min_{u \in U} \; y = R_w(f(u, p; w)), \tag{1}$$

where u denotes the configurable TSN parameters (i.e. the controls of the system listed in Table 3, which can be specified in the configuration file), p are the deterministic parameters of the network such as the traffic planning parameters given in Table 1 and the network design parameters such as the capacities of the hardware (i.e. the deterministic parameters that cannot be controlled via configuration), and w are the random variables that represent the uncertainty of the system (which are given in Table 2). Here, u, p, and w are n_u, n_p, and n_w dimensional vectors, respectively. Some entries in u and p are real values and some of them are integer/binary. The distributions for the random variables in w are not necessarily identical.

The function f returns the fraction of the lost packets, which is clearly a random quantity as a function of w. A robust formulation is constructed via the risk measure R_w, which consists of a combination of the expected value and the variance in our implementation.

Table 1. Deterministic Parameters (p): Hardware and Scenario Constraints

Parameter	**Unit / Type**	**Description / Constraint Role**
Packet Length	Bytes	Fixed size of the transmitted data packets in the simulation.
Link Usage (Bandwidth)	Percentage (%)	Target link utilization derived from the scenario traffic load.
Prioritized Traffic Ratio	Percentage	Proportion of link usage allocated to critical traffic versus best-effort.
Queue Size	$N \times KB$	Fixed buffer capacity for both ingress and egress interfaces.

Each evaluation of f given u, p, and w requires running a simulation of the communication network. Computation of y requires quantification of the uncertainty in f. Our methodology for solving (1) relies on learning a surrogate model to represent y as a function of u and p. Each data point to be used in training is generated by computing Monte-Carlo estimations to y for given values of u and p.

2.2 Learning-based Optimization Framework

We follow an active learning approach for surrogate model building. The basic mechanism of active learning is successive updates of the prediction model by

Table 2. Stochastic Parameters (w): Environmental Noise

Parameter	**Critical Level**	**Stochastic Impact**
Latency (Processing)	$> 100\,\mu s$ variance	Variations in internal processing time cause jitter, disrupting TAS synchronization.
Jitter	$> 30\,\mu s$ variance	The standard deviation of packet arrival times, representing the unpredictability of the channel.
CRC Error Rate	$> 10^{-5}$	Probabilistic bit errors that force frame drops at the MAC layer, independent of queue status.
Packet Loss Prob.	Stochastic	Non-congestion related packet loss probability due to physical medium degradation.
Device Down Prob.	Stochastic	The probability of a sudden hardware failure occurring during a simulation run.
Device Down Time	$10ms -> 1s$	The random duration required for a device to recover after a failure event.

Table 3. TSN Parameter Configurations and Modes

TSN Mechanism	**Configuration Modes**
QBV Mode (802.1Qbv)	11 values (Off / TDMA Structure: $12.5ms$ - $125ms$)
QCI Mode (802.1Qci)	11 values (Off / TDMA Structure): $12.5ms$ - $125ms$
PTP Mode (802.1AS)	Infinitely many values (n = frequency of PTP Req/ PTP Sync)
Master/Slave Status	2 values (Static Master or Slave assignment)

computing the labels of selected data so that the learning can be achieved with the smallest possible number of labeled data points[15]. The selection of data is often done with respect to an acquisition function. This is very similar to the so-called *infill* strategies found in simulation optimization literature[8]. The main difference is that the goal in active learning is to have a prediction model successful at random input points, whereas the infill regions in simulation optimization literature are generally focused on regions around minimizers of the earlier forms of the surrogate. We follow the active learning approach since we eventually embed the resulting surrogate to a mixed integer model to compute its global minimizer, and we also would like to use these models to explain the effects of configuration parameters on the packet loss.

As mentioned before, we decompose the parameter space into deterministic input (u, p) and random factors (w). Since the precise value of w (e.g., instantaneous jitter or specific background traffic arrival times) is unknown at the deployment stage, the objective is to predict the statistics of the packet loss rather than the result of a single realization. Therefore, for each input $x_i = (u_i, p_i)$, we conduct a set of repeated simulation runs ($k \in [10, 20]$ repetitions), varying w according to its underlying distributions. The target variables for the learning model are defined as the statistical moments of the packet loss

$$\mathcal{T}(x_i) = [\mathbb{E}[f(x_i; w)], \mathrm{Var}(f(x_i; w))]\,. \tag{2}$$

To ensure that the model generalizes well across the entire operational envelope, we structured the stochastic parameter space w into three distinct regimes: *Stable*, *Stress*, and *Crisis*.

- *Stable region.* Represents nominal operating conditions with low latency, jitter and no other hardening conditions.
- *Stress region.* Represents middle-latency, jitter scenarios with low-probability device closures, packet losses, crc errors.
- *Crisis region.* Represents high-latency, jitter scenarios with high-probability device closures, packet losses, crc errors.

While real-world operations predominantly occur in the *stable* and *stress* regions, the exclusion of the *crisis* region would lead to a model that is overly optimistic and blind to *black swan* events. Conversely, uniform sampling would undersample these critical failure modes.

We employ an active learning strategy initialized with domain expertise [17]. The initial dataset generation uses Latin hypercube sampling weighted towards known stable configurations. The iterative learning process is described in Algorithm 1 and Figure 1. Following the initial training, we utilize *stratified cross-validation* to identify regions of high prediction error. Points with high variance in prediction error are flagged as *uncertainty zones*. These zones guide the generation of new, targeted simulation scenarios, effectively shifting the sampling focus from exploration to exploitation. The loop continues until the model performance saturated, ensuring that the surrogate is reliable not only in predicting nominal behavior but also in flagging potential network failures in the *crisis* region.

Algorithm 1 Adaptive Surrogate Learning Workflow

Require: Initial Scenarios based on Domain Knowledge
1: **Initialization:** Execute simulations for base scenarios to create the initial dataset.
2: **repeat**
3: **Model training:** Train surrogate models using the current dataset.
4: **Validation:** Evaluate model accuracy using stratified cross-validation.
5: **Error analysis:** Identify input regions with high prediction residuals (uncertainty zones).
6: **Exploitation:** Generate new, targeted scenarios focused on these high-error regions.
7: **Simulation:** Execute high-fidelity simulations for the new scenarios.
8: **Data augmentation:** Append new results to the training dataset.
9: **until** Convergence criteria are met

The learned model is then plugged into the optimization model (1) replacing the objective function with the learned surrogate. In particular, we employ a mixed integer program formulation of the learned surrogate model and solve it via MIP techniques, which provides an approximate solution to problem (1). The MIP representability of various machine learning models has been discussed in recent work; see for instance [2,13,1]. For completeness, the detailed mixed-integer programming formulation of the random forest model used in our numerical study is provided in the Appendix.

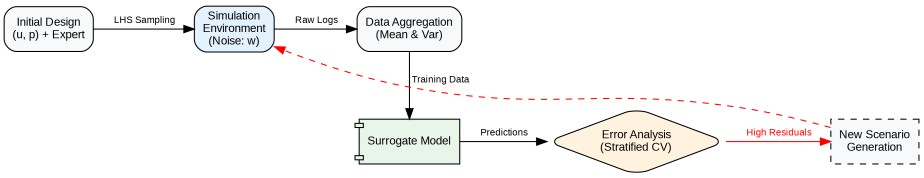

Fig. 1. Surrogate Learning Framework

3 Numerical Results

3.1 Experimental Setup

The experimental network architecture is designed as a star topology comprising a single switch and four endpoint devices, specifically engineered to induce bottleneck congestion at the switching node. The fundamental topology is illustrated in Figure 2. Color-coded devices establish communication pairs. Endpoints create traffic and forwarded via Switch as a main logic flow. The control parameters u, stochastic environmental factors w, and deterministic constraints p, which were detailed in comprehensively in Section 2.1, are implemented as variable attributes capable of dynamic modification across all devices within the topology. Note that inconsistent u parameters were not included in the simulations. For instance, if a device has an egress schedule of 125ms for PRI0, such gate open time represented by Qbv is given as 125ms.

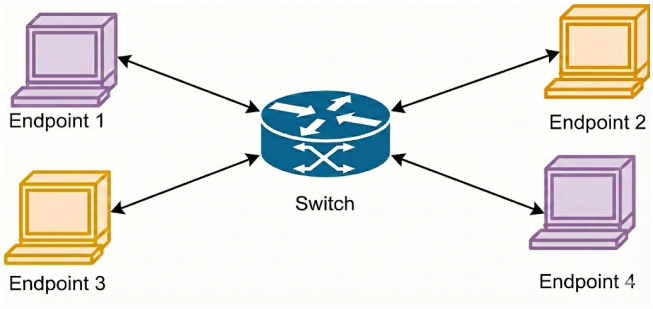

Fig. 2. Tested Network Topology

The simulation environment is established using the OMNeT++ 5.6.2 discrete event simulator running on the Ubuntu 18.0.4 LTS operating system. To ensure high fidelity and granular control, all network devices are modeled from fundamental components, and the entire orchestration—including data stream initiation and management—is executed via a command-line interface (CLI).

3.2 Surrogate Learning

To approximate the network behavior, we trained extreme gradient boosting (XGBoost) and random forest regressor (RFR) models. As detailed in Section 2.2, the training process followed an iterative active learning approach in a way to ensure balanced learning across critical regimes.

The data collection phase was progressive. Initial training iterations with sparse data yielded limited performance, with maximum R^2 scores plateauing around 0.3, where the models were heavily biased towards dominant features such as traffic load and packet length. To overcome this underfitting, the dataset was incrementally expanded in batches (e.g., 100, 200 samples). At the 200-sample mark, model performance exhibited high variance dependent on the random split; while specific random seeds achieved $R^2 > 0.9$, others failed to generalize, remaining at unacceptably low levels.

Convergence was definitively established at 300 samples. At this threshold, the variance stabilized, and no randomized training iteration yielded an R^2 score below 0.8.The final models were trained on this converged dataset, and their performance metrics are summarized in Table 4.

Table 4. Performance Comparison of Surrogate Models (R^2 Scores)

	Accuracy	
Model	**Mean**(R^2_μ)	**Variance**(R^2_σ)
XGBoost	0.9837	0.8765
RFR	0.9334	0.9029

As evidenced in Table 4, XGBoost achieved superior accuracy regarding the expected packet loss ($R^2_\mu = 0.9837$). In parallel, RFR demonstrated robust performance with an R^2_μ of 0.9334. While slightly less accurate than XGBoost in predicting the mean, RFR's ensemble structure of independent decision trees allows for direct linearization into MIP constraints, a prerequisite for the global optimization stage described in the following section.

The feature importance analysis in Figure 3 identifies the Time-Aware Shaper (Qbv) configuration on both endpoints and switches as the dominant determinant of packet loss, accounting for over 54% of the model's decision weight. This insight validates the necessity of precise Qbv tuning in the optimization phase. Also Qci settings importance on model decision is about 19%.

3.3 MIP Solution

The optimization objective in (1) is explicitly defined as the sum of the expected packet loss and its standard deviation. We formulated a MIP using the trained RFR model as the objective function (see Appendix for the details of

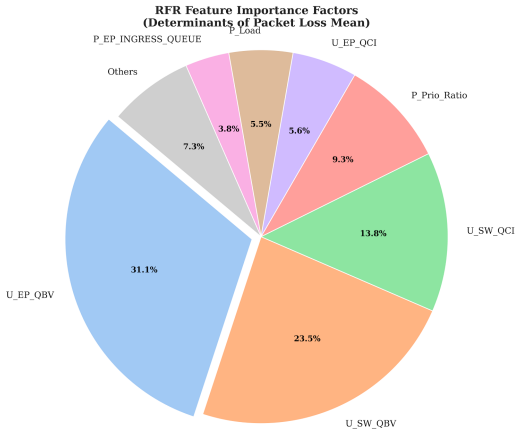

Fig. 3. Feature Importance Factors Derived from the RFR Model

the formulation) and employed Gurobi 13.0 for solving it. Given the fixed deterministic parameters (p), Gurobi searches the solution space for the optimal robust configuration u^*.

To validate the reliability of the optimization outcomes, we subjected the optimal configuration u^* found by the Gurobi optimizer to high-fidelity simulations under 10 different realizations of w. The results, consolidated in Table 5, demonstrate the capability of the surrogate model to accurately predict the risk profile of the network across different operational regimes.

Table 5 contrasts two distinct operational environments. **Scenario A** is characterized by a heavy aggregate traffic load of 380%. It is important to clarify that this figure represents the cumulative ingress traffic arriving at the switch from multiple endpoints (e.g., four endpoints each utilizing 95% of their link capacity), thereby creating a severe bottleneck at the switch egress. Despite the large buffer capacities (512 KB), the simulation reveals a bimodal *all-or-nothing* behavior where catastrophic losses occur in specific runs (e.g., Run 9 and 10).

Scenario B illustrates a *crisis* regime driven by micro-burst traffic (100 Bytes) and severely constrained queues (8 KB). In this volatile environment, the Gurobi optimizer correctly activated Ingress Policing (QCI) on both switches and endpoints to mitigate overflow. Similar to Scenario A, the losses range from negligible ($< 0.1\%$) to total collapse ($> 98\%$). These extreme deviations in both scenarios are direct consequences of the crisis regime dynamics, driven by prolonged device shutdown durations and intentionally chaotic latency profiles introduced into the simulation. Conversely, the *stress* region (Runs 1-8) exhibits negligible deviation from nominal performance because device failures in this regime are modeled with extremely short recovery times, allowing buffers to absorb transient interruptions. Crucially, the surrogate model correctly predicted the statistics of these high-risk profiles, remarkably close to the actual simulation results for both scenarios.

Table 5. Validation of Optimization Results: Stochastic Realizations vs. Model Predictions

Metric / Parameter	**Scenario A (Large Buffers)**	**Scenario B (Crisis Regime)**
	Deterministic Constraints (p)	
Traffic Load	380%	390%
Packet Length	1500 Bytes	100 Bytes
Queue Capacity	512 KB	8 KB
	Optimal Control Configuration (u^*)	
Active Mechanisms	SW_{QCI} : **On** EP_{QCI} : Off SW_{QBV} : Off EP_{QBV} : Off	SW_{QCI} : **On** EP_{QCI} : **On** SW_{QBV} : Off EP_{QBV} : Off
PTP Frequency	450 Hz	375 Hz
	Stochastic Realizations ($w_{1...10}$) - Packet Loss (%)	
Run 1	0.0	0.01
Run 2	0.0	0.0
Run 3	0.0	0.0
Run 4	0.0	0.01
Run 5	0.0	4.09
Run 6	0.0	0.17
Run 7	0.0	1.50
Run 8	0.0	4.71
Run 9	76.94	98.14
Run 10	80.15	98.56
	Model vs. Simulation Statistics	
Actual Mean	**15.71%**	**20.72%**
Predicted Mean	15.56%	20.77%
Actual Std. Dev.	**31.40%**	**39.15%**
Predicted Std. Dev.	30.78%	37.10%

4 Conclusions

In this work, we introduced a robust optimization framework for packet loss minimization in TSN. Our solution methodology is based on MIP-representable learned surrogates with a model training process driven by active learning and domain knowledge. We tested the performance of our approach on an example communication network, which yielded encouraging results.

Our work can be extended in several directions. First of all, the proposed methodology can be tested on more complex topologies. Different MIP-representable machine learning surrogates can be tried, such as multilayer perceptrons with ReLU activation. Moreover, the robustness of the framework can be further improved by defining additional rules for TSN and deterministic parameters, ensuring a closer alignment with real-world application scenarios.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Anderson, R., Huchette, J., Ma, W., Tjandraatmadja, C., Vielma, J.P.: Strong mixed-integer programming formulations for trained neural networks. Mathematical Programming **183**(1), 3–39 (2020)
2. Bergman, D., Huang, T., Brooks, P., Lodi, A., Raghunathan, A.U.: Janos: an integrated predictive and prescriptive modeling framework. INFORMS Journal on Computing **34**(2), 807–816 (2022)
3. Bezerra, V., Ibanez, F., M., H.: Machine Learning-based Optimization of Time-Sensitive Networking. IEEE Access **10**, 12345–12360 (2022)
4. Bui, D.T., Dupas, A., Le Pallec, M.: Packet delay variation management for a better ieee1588v2 performance. In: 2009 International IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication. pp. 1–6 (2009). `https://doi.org/10.1109/ISPCS.2009.5340197`
5. Cozad, A., Sahinidis, N.V., Miller, D.C.: Learning Surrogate Models for Simulation-Based Optimization. AIChE Journal **60**(6), 2211–2227 (2014)
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
7. Dellino, G., Kleijnen, J.P., Meloni, C.: Metamodel-based robust simulation-optimization: An overview. Uncertainty Management in Simulation-Optimization of Complex Systems: Algorithms and Applications pp. 27–54 (2015)
8. Forrester, A., Sobester, A., Keane, A.: Engineering design via surrogate modelling: a practical guide. John Wiley & Sons (2008)
9. IEEE 802.3 Working Group: IEEE Standard for Ethernet. IEEE Std 802.3-2018 (2018), doi: 10.1109/IEEESTD.2018.8457469
10. Koziel, S., Ciaurri, D.E., Leifsson, L.: Surrogate-based methods. In: Computational optimization, methods and algorithms, pp. 33–59. Springer (2011)
11. Le Boudec, J.Y., Thiran, P.: Network Calculus: A Theory of Deterministic Queuing Systems for the Internet, Lecture Notes in Computer Science, vol. 2050. Springer-Verlag (2001)
12. Lo Bello, L., Wilwert, C.: The Quest for Determinism in Industrial Ethernet: A Survey. IEEE Transactions on Industrial Informatics **15**(9), 5355–5367 (2019)
13. Maragno, D., Wiberg, H., Bertsimas, D., Birbil, Ş.İ., den Hertog, D., Fajemisin, A.O.: Mixed-integer optimization with constraint learning. Operations Research **73**(2), 1011–1028 (2025)
14. Mulvey, J.M., Vanderbei, R.J., Zenios, S.A.: Robust optimization of large-scale systems. Operations research **43**(2), 264–281 (1995)
15. Murphy, K.P.: Probabilistic machine learning: Advanced topics. MIT press (2023)
16. Nebro, A.J., Durillo, J.J., Garcia-Nieto, J., Coello, C.A.C., Luna, F., Alba, E.: SMPSO: A New PSO-based Metaheuristic for Multi-objective Optimization. In: IEEE Congress on Evolutionary Computation. pp. 66–73 (2009)
17. Settles, B.: Active Learning Literature Survey. University of Wisconsin-Madison Department of Computer Sciences (2009), technical Report 1648

Appendix: Mixed Integer Programming Formulation of the RFR Surrogate

A. Notation and Sets

Let $\mathcal{T}_\mu$ and $\mathcal{T}_\sigma$ denote the set of decision trees in the Random Forest ensembles for the mean and standard deviation models, respectively. For each tree $t \in \mathcal{T}$ (where $\mathcal{T} = \mathcal{T}_\mu \cup \mathcal{T}_\sigma$), let $\mathcal{L}_t$ be the set of leaf nodes.

Decision Variables (u): The control vector u consists of decision variables representing the TSN configuration parameters:

- $x_{mech} \in \{0, 1\}$: Activation status for TSN mechanisms (where $mech \in \{Qbv, Qci\}$).
- $x_{ptp} \in \mathbb{Z}_{\geq 0}$: The PTP synchronization frequency(integer).
- $x_{slope,i} \in \{0, 1\}$: Selection of the i-th discrete value for $IdleSlope$.
- $x_{size,j} \in \{0, 1\}$: Selection of the j-th discrete value for $FrameSize$.

Auxiliary Variables:

- $y_{t,l} \in \{0, 1\}$: Binary variable indicating if leaf l is active in tree t.

Parameters:

- $V_{t,l}$: The scalar prediction value (packet loss) stored in leaf l of tree t.
- $\mathcal{P}_{t,l}$: The set of split conditions (decision path) required to reach leaf l in tree t.

B. Objective Function

The optimization problem minimizes the composite risk metric:

$$\min \; Z = \underbrace{\frac{1}{|\mathcal{T}_\mu|} \sum_{t \in \mathcal{T}_\mu} \sum_{l \in \mathcal{L}_t} V_{t,l} \cdot y_{t,l}}_{\text{Predicted Mean } (\mu)} + \underbrace{\frac{1}{|\mathcal{T}_\sigma|} \sum_{t \in \mathcal{T}_\sigma} \sum_{l \in \mathcal{L}_t} V_{t,l} \cdot y_{t,l}}_{\text{Predicted Std. Dev. } (\sigma)} \tag{3}$$

C. Constraints

1. Unique Configuration Constraints: To ensure a valid physical configuration, exactly one setting must be selected for discrete parameters (IdleSlope and FrameSize) for each flow class:

$$\sum_i x_{slope,i} = 1, \quad \sum_j x_{size,j} = 1 \tag{4}$$

2. Tree Consistency Constraints: In a Random Forest, exactly one leaf must be active per tree based on the input configuration u. We map the decision path of each leaf to linear constraints. If the path to leaf l requires a feature x_k to be 1 (active) or 0 (inactive), the following constraints enforce this logic:

$$\sum_{l \in \mathcal{L}_t} y_{t,l} = 1, \quad \forall t \in \mathcal{T} \tag{5}$$

For each leaf $l \in \mathcal{L}_t$, let $\mathcal{S}_l^+$ be the set of decision variables required to be 1, and $\mathcal{S}_l^-$ be the set required to be 0 to reach that leaf. The path consistency is enforced by:

$$y_{t,l} \leq x_k, \quad \forall k \in \mathcal{S}_l^+ \tag{6}$$

$$y_{t,l} \leq (1 - x_k), \quad \forall k \in \mathcal{S}_l^- \tag{7}$$

These constraints ensure that a leaf $y_{t,l}$ can only be active ($= 1$) if and only if the global configuration u satisfies all split conditions along the path to that leaf. The solver (Gurobi) then searches for the optimal combination of x variables that steers the active leaves towards the minimal total $V_{t,l}$ sum.