Graph Neural Networks for Low-Latency Rate Monotonic Admission Control in Edge IoT

Shashi Prabh

Ahmedabad University, Ahmedabad

***Abstract*—In dynamic edge IoT environments, task orchestrators must perform online admission control at high frequency. While exact Response-Time Analysis (RTA) for Rate Monotonic scheduling provides maximum resource utilization, its pseudo-polynomial computational complexity introduces unpredictable latency, rendering it unsuitable for high-frequency edge orchestration. Conversely, $\mathcal{O}(n)$ analytical bounds are fast but overly pessimistic, leading to wasted compute capacity. In this paper, we propose a graph-based admission controller that trades analytical exactness for predictable, low-latency inference. We map periodic task sets to directed priority-interference graphs with a 13-dimensional node feature set that encodes the key quantities of the RTA fixed-point computation. An inductive GraphSAGE architecture predicts per-task schedulability logits, aggregated by a worst-task operator that mirrors exact RTA semantics. Combined with an asymmetric loss function that penalizes unsafe admissions, the model acts as a conservative admission controller. Trained exclusively on small task sets ($n \leq 10$), the model generalizes inductively to larger task sets without retraining. Evaluations against exact RTA, analytical bounds, and three ML baselines demonstrate that the architecture compresses the variable, millisecond-scale latency of exact RTA into a predictable inference profile, achieving up to $6.4\times$ speedup while maintaining strong false-positive suppression.**

***Index Terms*—Edge IoT, Admission Control, Rate Monotonic Scheduling, Graph Neural Networks, Real-Time Systems.**

I. Introduction

The proliferation of dense wireless edge Internet-of-Things (IoT) deployments and swarm robotics has pushed the boundaries of edge computing. In these environments, edge nodes must act as dynamic orchestrators, constantly admitting and scheduling transient, multi-rate (tasks with widely differing periods) sensor tasks. Because these tasks often carry timing constraints, the edge admission controller must determine whether a candidate task set is schedulable, i.e., whether every task will meet its deadline under preemptive scheduling policies such as Rate Monotonic (RM) [1].

The core challenge in edge orchestration is the computational latency of the admission decision and its variability. Admission controllers rely on exact Response-Time Analysis (RTA) [2] to maximize the utilization of scarce edge resources. RTA is exact but its time complexity is *pseudo-polynomial*. The number of iterations needed to reach a fixed point grows with the number of tasks n and with the ratio between the largest and the smallest task period. In multi-rate IoT systems this can run into thousands of iterations. The resulting latency jitter degrades the responsiveness of edge task dispatching. Further, task dispatchers cannot afford to wait milliseconds to simply *decide* if a workload is viable. System designers often default to classical analytical utilization bounds. These bounds execute in $O(n)$ time. Although fast, they are pessimistic and often reject task sets that are in fact schedulable, which suppresses utilization. Therefore, modern edge orchestrators face a trilemma: they require low and predictable latency, high schedulability accuracy to maximize resource utilization, and the ability to reject unschedulable task sets to prevent deadline violations. They must balance these three conflicting requirements, which render traditional methods insufficient for large-scale, dynamic edge deployments.

Recent applications of machine learning (ML) to real-time systems have attempted to approximate exact schedulability [3]. However, standard architectures such as Multi-Layer Perceptrons (MLPs) and transductive Graph Convolutional Networks (GCNs) [4] require a fixed input dimension. They therefore do not scale to unseen workloads. Further, standard loss functions treat false positives and false negatives equally. Such losses ignore the asymmetric cost structure of admission control errors and do not resolve the edge trilemma.

In this paper, we bridge this gap by framing RM admission control as a per-task graph prediction problem with asymmetric error costs. Our main contributions are:

1) *RTA-aware Graph Representation and Per-Task Prediction:* We map implicit-deadline task sets to directed priority-interference graphs with a 13-dimensional node feature set encoding the key quantities of the RTA fixed-point computation. An inductive GraphSAGE architecture [5] predicts per-task schedulability logits, which are aggregated by a worst-task operator that mirrors exact RTA semantics. A model trained on small task sets ($n \leq 10$) executes inference on unseen larger task sets ($10 < n \leq 100$) without retraining.
2) *Asymmetric Loss Training:* We train with an asymmetric loss function that heavily penalizes false positives, forcing the model to act as a conservative admission controller that suppresses deadline violations even on out-of-distribution (OOD) topologies.
3) *Predictable Inference Latency:* We empirically demonstrate that the proposed architecture transforms the

pseudo-polynomial, variable latency of exact RTA into a low-variance inference process, achieving up to $6.4\times$ speedup.

The remainder of this paper is organized as follows. Section II reviews related work. Section III details the system model and the RTA bottleneck. Section IV presents the GraphSAGE architecture and asymmetric training methodology. Section V provides the performance evaluation against RTA and ML baselines, and Section VI concludes the paper.

II. Related Work

A. Analytical Schedulability Analysis

The foundation of fixed-priority schedulability analysis was established by the classical utilization bound for RM scheduling [1], which provides an $\mathcal{O}(n)$ sufficient-only test. Various polynomial-time tests have been proposed to reduce the pessimism of this bound, such as the Hyperbolic Bound [6]. Although efficient, these bounds still reject a fair fraction of feasible task sets and leave resources underutilized. Exact schedulability is determined using RTA [2]. RTA iteratively computes the worst-case interference for each task. Its worst-case time complexity is pseudo-polynomial. This makes it unsuitable for online execution in dynamic, resource-constrained environments where task sets scale into the dozens or hundreds and predictable, low-variance decision latency is needed.

B. Machine Learning for Schedulability

The intersection of machine learning and real-time scheduling has recently been surveyed by Guo [7], who identifies schedulability prediction, WCET estimation, and RL-based task dispatching as three distinct application classes. Most directly related to our work, Baruah et al. [8] recently proposed a deep-learning framework for schedulability analysis that guarantees zero unsafe classifications. Their approach constrains the network output so that every positive prediction is provably safe, achieving over 70% accuracy on systems of up to 20 tasks. While their safety guarantee is formal, it comes at the cost of high pessimism on larger systems and does not address inference latency.

C. Graph Neural Networks for Structured Prediction

Graph Neural Networks (GNNs) have emerged as a powerful paradigm for learning over relational data. Kipf and Welling [4] introduced Graph Convolutional Networks (GCNs) for semi-supervised node classification, but GCNs are inherently transductive, that is, they learn fixed spectral embeddings tied to a specific graph topology. Hamilton et al. [5] addressed this limitation with GraphSAGE, an inductive framework that learns neighborhood aggregation functions transferable to unseen graphs. This inductive property is critical for our application, where the number of tasks n varies dynamically at the edge. More broadly, GNNs have been applied to combinatorial optimization problems such as routing, scheduling, and constraint satisfaction. Cappart et al. [9] provide a comprehensive survey, where the authors note that GNNs' permutation invariance and ability to exploit relational structure make them well suited for problems where the input is a variable-size set of interacting entities, which is precisely the structure of a priority-scheduled task set.

D. The Identified Gap and This Paper

Classical approaches face a fundamental trilemma for online admission control at the edge where the analytical bounds sacrifice accuracy for speed, exact RTA sacrifices speed for accuracy, and existing ML-based tests [3], [8] lack structural scalability, i.e., fail to provide inductive generalization to unseen task set sizes, or do not address inference latency. This paper bridges this gap by combining three elements absent from prior work, which are: (i) an *inductive* GraphSAGE architecture that generalizes to arbitrary workload sizes without retraining, (ii) a per-task prediction structure with worst-task aggregation that mirrors exact RTA semantics, and (iii) an *asymmetric loss function* inspired by cost-sensitive learning [10] that forces the model to act as a conservative admission controller. Together, these yield a low-latency, conservative classifier suited for dynamic wireless edge deployment.

III. System Model and Problem Formulation

A. System and Task Model

We consider a single-processor edge computing node responsible for executing a set of n independent, preemptable, periodic tasks, denoted as $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$. Each task τ_i is characterized by the tuple (C_i, T_i), where C_i represents the Worst-Case Execution Time (WCET) and T_i represents the period. We assume an implicit-deadline model, where the relative deadline D_i of each task is exactly equal to its period, i.e., $D_i = T_i$.

Tasks are scheduled according to the preemptive Rate Monotonic (RM) scheduling policy. RM is a *static-priority algorithm* that assigns higher priorities to tasks with shorter periods. Without loss of generality, we assume the tasks in Γ are indexed in decreasing order of priority, such that whenever $i < j$, it implies $T_i \leq T_j$.

The processor utilization of an individual task τ_i is given by $U_i = C_i/T_i$, and the total utilization of the task set is $U = \sum_{i=1}^{n} U_i$. A necessary condition for the task set to be schedulable is $U \leq 1$.

B. Exact Response-Time Analysis (RTA)

For a task set to be schedulable, no task can miss its deadline. Therefore, the worst-case response time R_i of each task τ_i must satisfy $R_i \leq T_i$. Under RM scheduling, the exact worst-case response time R_i is computed by finding the smallest fixed point of the following recursive equation:

$$R_i^{(k+1)} = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{R_i^{(k)}}{T_j} \right\rceil C_j. \tag{1}$$

The recurrence relation (1) is initialized with $R_i^{(0)} = C_i$. The term inside the summation represents the cumulative interference caused by all the higher-priority tasks τ_j (where $j < i$) preempting τ_i during its execution. The iteration terminates under one of two conditions: (a) $R_i^{(k+1)} = R_i^{(k)}$ meaning the fixed point is reached. If $R_i^{(k+1)} \leq T_i$, task τ_i is schedulable. (b) $R_i^{(k+1)} > T_i$ meaning the response time exceeds the period, thus task τ_i is unschedulable. Then the exact RTA can safely reject the entire task set Γ.

C. The Admission Control Bottleneck at the Edge

While the RTA provides an exact, necessary, and sufficient schedulability test for implicit-deadline RM systems, its mathematical structure presents a serious operational bottleneck for real-time edge systems. The computational complexity of the fixed-point equation is pseudo-polynomial. The number of iterations required to converge is highly sensitive to the hyperperiod of the task set, the ratio between the maximum and minimum periods, and the total system utilization U. Exact fixed-priority schedulability of implicit-deadline tasks is NP-complete once U exceeds the Liu–Layland bound $\ln 2$ [11]. This bound separates tractable from intractable testing, and it is the high-utilization regime targeted here. Computing a task's worst-case response time is itself NP-hard [12].

Edge orchestrators must perform online admission control for transient tasks at high frequencies. The execution time of the RTA algorithm spikes when the node operates under heavy load ($U \approx 1$) or when the number of concurrent tasks n scales up. The resulting latency and jitter degrade the responsiveness of edge orchestration. Purely analytical polynomial bounds offer fast $\mathcal{O}(n)$ admission decisions but are pessimistic [1]. One such bound is the classical Liu and Layland bound $U \leq n(2^{1/n} - 1)$. These bounds have a high false-negative rate. They reject task sets that are in fact schedulable and leave wireless edge compute resources underutilized. An admission control mechanism is therefore needed that trades analytical exactness for predictable, low-latency classification while maintaining high accuracy in the high-utilization regime.

IV. Graph-based Schedulability Architecture

To overcome the pseudo-polynomial latency of exact RTA, we frame the schedulability test as a per-task graph prediction problem. By modeling the preemption interference between tasks as directed edges and predicting each task's schedulability individually, a GNN can learn to approximate the exact RTA decision boundary with predictable, low-variance inference latency (Figure 1).

A. Task Sets as Priority-Interference Graphs

We map a periodic task set Γ to a directed, attributed graph $G = (\mathcal{V}, \mathcal{E})$. Each node $v_i \in \mathcal{V}$ corresponds to a task $\tau_i \in \Gamma$. Tasks are sorted by non-decreasing period, i.e., by RM priority. A directed edge $(v_i, v_j) \in \mathcal{E}$ is created for every pair where $i < j$, indicating that τ_i can preempt τ_j. The resulting graph is a complete directed acyclic graph (DAG) over the priority order, explicitly encoding the full interference cascade that RTA resolves iteratively.

Each node carries a feature vector $\mathbf{x}_i \in \mathbb{R}^{13}$ designed to expose the quantities that govern the RTA fixed-point computation as listed below:

- *Prefix accumulations:* cumulative utilization $\sum_{j \leq i} U_j$, residual slack proxy $1 - \sum_{j \leq i} U_j$, margin to the Liu-Layland bound, and margin to the Hyperbolic Bound [6], all computed over the prefix of higher-priority tasks up to and including τ_i.
- *Local timing:* per-task utilization U_i, $\log(1+C_i)$, and $\log(1+T_i)$.
- *Priority context:* normalized priority rank $i/(n-1)$ and fractional position i/n.
- *Period structure:* log-ratio to the preceding and following periods, a binary harmonicity indicator for the adjacent period pair, and $\log(1+n)$.

This 13-dimensional representation is used by all architectures evaluated in Section V, ensuring a fair comparison.

B. Inductive Interference Aggregation via GraphSAGE

Motivated by the need for inductive inference in dynamic edge IoT environments where the number of tasks n fluctuates dynamically, we utilize GraphSAGE, an inductive framework that learns aggregation functions rather than structural embeddings. At each layer l, the embedding $\mathbf{h}_i^{(l)}$ of node v_i is updated by aggregating the representations of its neighborhood $\mathcal{N}(i)$ as follows:

$$\mathbf{h}_{\mathcal{N}(i)}^{(l)} = \text{AGGREGATE}_l\left(\left\{\mathbf{h}_j^{(l-1)} s.t.\ j \in \mathcal{N}(i)\right\}\right), \quad (2)$$

$$\mathbf{h}_i^{(l)} = \sigma\left(\mathbf{W}^{(l)} \cdot \text{CONCAT}\left(\mathbf{h}_i^{(l-1)},\ \mathbf{h}_{\mathcal{N}(i)}^{(l)}\right)\right). \quad (3)$$

Our three-layer GraphSAGE architecture has node embedding dimensions $13 \to 64 \to 64 \to 64$ and uses LayerNorm and dropout (p=0.2) after each layer. Because the aggregator functions are size-agnostic, a model trained on smaller task sets can execute inference on larger edge workloads without retraining.

C. Per-Task Prediction and Worst-Task Aggregation

Exact RTA determines schedulability by checking *every* task individually. The task set is unschedulable if any single task misses its deadline. We mirror this structure in our architecture. Rather than collapsing all node embeddings into a single graph-level vector via global pooling, each node's final embedding $\mathbf{h}_i^{(3)}$ is passed through a per-task MLP head to produce a scalar pass logit $\hat{s}_i$ as

$$\hat{s}_i = \text{MLP}_{\text{task}}\left(\mathbf{h}_i^{(3)}\right), \quad \forall i \in \mathcal{V}.$$

The graph-level schedulability logit is then obtained by a worst-task aggregation operator that selects the most pessimistic per-task prediction:

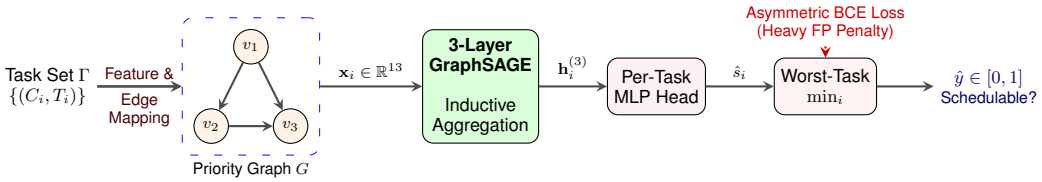

Fig. 1. Architectural pipeline. A task set is mapped to a priority-interference graph with RTA-aware node features, processed by GraphSAGE layers producing per-task embeddings, and classified via per-task pass logits aggregated by a worst-task min operator. Asymmetric loss penalizes unsafe admissions.

$$\hat{y}_{\text{graph}} = \min_{i \in \mathcal{V}} \hat{s}_i.$$

This formulation directly encodes the RTA invariant that a single failing task renders the entire set unschedulable. It also provides interpretability. The identity of the task with the lowest logit $\hat{s}_i$ indicates the task the model considers the bottleneck, similar to the first failing task in exact RTA.

During training, the per-task logits receive an auxiliary supervision signal. Let $p_i \in \{0, 1\}$ denote the per-task pass label from exact RTA, $p_i = 1$ if $R_i \leq T_i$, and 0 otherwise. The total loss combines the graph-level and task-level terms as:

$$\mathcal{L} = \mathcal{L}_{\text{asym}}(\hat{y}_{\text{graph}},\, y) \;+\; \alpha \cdot \mathcal{L}_{\text{asym}}(\hat{\mathbf{s}},\, \mathbf{p}), \tag{4}$$

where $\alpha = 0.5$ weights the auxiliary per-task loss and $\mathcal{L}_{\text{asym}}$ is the asymmetric loss defined below.

D. Asymmetric Loss for Safe Admission Control

In edge orchestration, classification errors carry asymmetric operational costs. A false negative (FN), i.e., rejecting a schedulable task set, wastes compute capacity. A false positive (FP), i.e., admitting an unschedulable workload, causes deadline violations and degrades system reliability.

Standard Binary Cross-Entropy (BCE) treats these errors equally, which is inadequate for conservative admission control. We train all models using an asymmetric BCE loss that penalizes FPs. Let $y \in \{0, 1\}$ be the exact RTA label and $\hat{y}$ the model output. The asymmetric loss is:

$$\mathcal{L}_{\text{asym}} = -\frac{1}{N} \sum_{k=1}^{N} \left[y_k \log(\hat{y}_k) + \lambda\, (1 - y_k) \log(1 - \hat{y}_k) \right], \tag{5}$$

where $\lambda > 1$ inflates the gradient for unschedulable samples (y=0). This pushes the decision boundary inward, causing the controller to default to rejection under uncertainty. The same asymmetric loss is applied to both the graph-level and per-task terms in Eq. (4).

V. Performance Evaluation

A. Experimental Setup

We generated a synthetic dataset of implicit-deadline periodic task sets using the UUniFast algorithm [13], [14] to prevent statistical utilization bias. Periods were sampled log-uniformly between 10 ms and 1000 ms to model heterogeneous, multi-rate edge sensor environments. Total utilization was biased toward the critical boundary, $U \in [0.75, 1.0]$, to maximize training signal in the region where analytical bounds fail. The dataset was balanced with 75% schedulable and 25% unschedulable task sets to reflect the natural distribution in the critical band. All models were trained with the asymmetric loss keeping λ=5 and evaluated over 3 random seeds. Experiments were conducted on a 12th-generation Intel Core i7 processor (16 cores) with 16 GB system memory and an NVIDIA GeForce MX550 GPU (1 024 CUDA cores, 2 GB dedicated memory). All latency measurements were taken on the GPU.

Two evaluation splits were generated as follows:

- *Standard Test*, $N \in \{2, \ldots, 10\}$. 7 500 task sets drawn from the same size range used during training. During training, 85 000 training task sets and 7 500 validation task sets were used.
- *OOD Swarm*, $N \in \{15, \ldots, 100\}$. 50 000 task sets with sizes strictly larger than any seen during training, testing inductive generalization.

The 3-layer GraphSAGE model was benchmarked against three ML baselines, namely, (i) a mean-pooling MLP, (ii) a Set Transformer [15], and (iii) a PrefixRiskMLP, which applies masked `min` aggregation to per-task logits without message passing. We also benchmarked against the classical Hyperbolic Bound [6]. All ML models share the same 13-dimensional feature set and asymmetric loss, thus isolating the architectural contribution.

B. Architectural Generalization and Edge Safety

Table I summarizes accuracy, false positive rate (FPR), and false negative rate (FNR) on both evaluation splits. On the Standard test set, all four ML architectures achieve ≥89% accuracy with FPR of at most 4.3%, well above the analytical bound, which had 45.4% accuracy with 72.7% FNR. The critical differentiation emerges on the OOD Swarm, where models must generalize to task sets 1.5–10× larger than any training example. The MLP, which pools task features into a fixed-length vector, collapses on OOD workloads. Its accuracy is 65.7% with 45.7% FNR, that is, it rejects most unseen topologies (Figure 2). The Set Transformer maintains high OOD accuracy of 95.0% but at the cost of 5.3% FPR. Recall that FPs are unsafe admissions that would cause deadline misses in deployment. GraphSAGE achieves the best safety-accuracy trade-off with 91.8% OOD accuracy and only 0.9% FPR, an 85% reduction in the

TABLE I
ARCHITECTURAL COMPARISON: ACCURACY AND SAFETY (MEAN OVER 3 SEEDS, λ=5). FPR: UNSAFE ADMISSIONS; FNR: WASTED CAPACITY.

Split	Architecture	Acc.	FPR	FNR
Standard ($N \leq 10$)	Hyperbolic Bound	45.4%	0.0%	72.7%
	MLP Baseline	92.6%	1.4%	9.4%
	Set Transformer	93.3%	1.8%	8.3%
	PrefixRiskMLP	89.1%	4.3%	13.1%
	GraphSAGE (Ours)	**92.1%**	**1.8%**	**9.9%**
OOD ($N \geq 15$)	Hyperbolic Bound	34.9%	0.0%	86.8%
	MLP Baseline	65.7%	0.2%	45.7%
	Set Transformer	95.0%	5.3%	4.8%
	PrefixRiskMLP	84.7%	0.9%	20.1%
	GraphSAGE (Ours)	**91.8%**	**0.8%**	**10.7%**

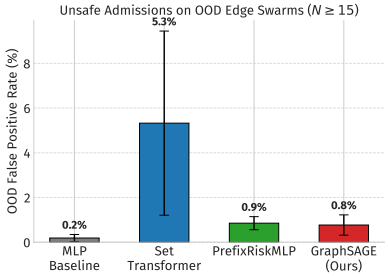

Fig. 2. OOD false positives across architectures. GraphSAGE achieves 85% fewer unsafe admissions than the Set Transformer while maintaining higher accuracy than the PrefixRiskMLP.

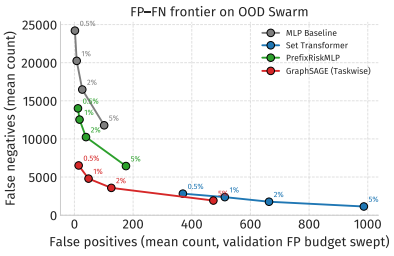

Fig. 3. FP–FN Pareto frontier on the OOD Swarm under matched-FP-budget evaluation. Each point is a validation FPR budget from 0.5% to 5%. GraphSAGE (red) dominates the lower-left with low FP *and* low FN at every budget level.

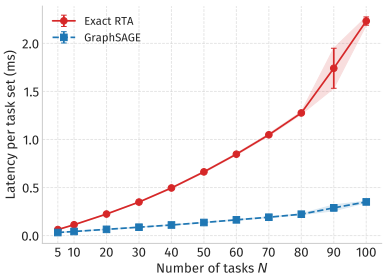

Fig. 4. Inference latency: exact RTA vs. batched GraphSAGE across task set sizes n from 5 to 100.

unsafe admission rate compared to the Set Transformer. The PrefixRiskMLP matches GraphSAGE's 0.8% OOD FPR but sacrifices 7 percentage points of OOD accuracy. It rejects far more viable workloads, namely, 20.1% FNR, compared to 10.7% for GraphSAGE.

To evaluate models under a common safety budget, we employ a matched-FP-budget methodology, that is, for each model, we tune the classification threshold on the validation set to admit at most a target FPR, then evaluate on OOD data at that threshold. Figure 3 shows the resulting FP–FN Pareto frontier on the OOD Swarm at budgets of 0.5%, 1%, 2%, and 5%. GraphSAGE (shown in red) consistently occupies the lower-left region of the plot, achieving the best trade-off between safety and utilization. At the strictest budget, 0.5%, GraphSAGE achieves 0.11% OOD FPR while maintaining 86.9% OOD accuracy, and is the only architecture that simultaneously sustains high OOD accuracy and near-zero unsafe admissions.

C. Inference Latency

Beyond classification accuracy, the primary objective of this architecture is to replace the pseudo-polynomial, high-variance execution time of exact RTA with a predictable, low-variance inference pass. Figure 4 compares inference latency on an adversarial task-set family designed to stress RTA convergence.

RTA latency grows super-linearly with n and exhibits high variance. At n=100, mean RTA latency reaches 2.23 ms with peaks above 2.3 ms. GraphSAGE inference stays sub-millisecond. Its latency grows from 0.03 ms at n=5 to 0.35 ms at n=100, a 6.4× mean speedup at the largest workload size. The GNN's latency profile also has low variance. The standard deviation was <0.01 ms at each n. Such a profile suits orchestrators that require bounded, predictable decision times.

D. Asymmetric Loss Ablation

To quantify the safety–utilization trade-off, we swept the asymmetric penalty $\lambda \in \{1, 2, 5, 10, 20\}$ and evaluated GraphSAGE on the critical utilization band, $0.85 \leq U \leq 1.0$, where analytical bounds achieve only 42–45% accuracy. Table II and Figure 5 present the results.

At λ=1, corresponding to standard BCE, the model exhibits an 8.3% FPR in the critical zone. Increasing to λ=5 reduces FPR to 1.6% while FNR rises from 8.8% to 20.5% and accuracy decreases by 3.9 percentage points. At λ=20, FPR drops to 0.2% but FNR climbs to 33.2%. The λ=5 operating point provides a practical balance for conservative deployment, where there are near-zero unsafe admissions with moderate pessimism. At λ=20 FPR drops to just 0.2% (3 unsafe admissions out of 1 861 unschedulable sets in the critical band), approaching the zero-FP guarantee of the Hyperbolic Bound while retaining 80.9% accuracy which is nearly double the Bound's 45%. The FNR of the Hyperbolic Bound, in contrast, is 95.7%.

TABLE II
LAMBDA ABLATION ON THE CRITICAL BAND, $0.85 \leq U \leq 1.0$. HIGHER λ TRADES UTILIZATION (HIGHER FNR) FOR SAFETY (LOWER FPR).

Loss Configuration	FPR	FNR	Acc.
Standard BCE (λ=1)	8.3%	8.8%	91.4%
Asymmetric (λ=2)	4.2%	13.6%	90.4%
Asymmetric (λ=5)	1.6%	20.5%	87.5%
Asymmetric (λ=10)	0.6%	26.9%	84.3%
Asymmetric (λ=20)	0.2%	33.2%	80.9%

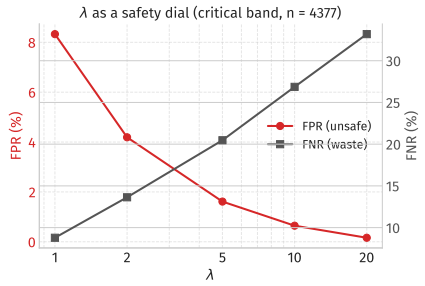

Fig. 5. λ as a safety dial on the critical band, $0.85 \leq U \leq 1.0$. Increasing λ smoothly trades FPR (unsafe admissions, red, left axis) for FNR (wasted capacity, gray, right axis).

E. Feature Ablation

To isolate the contribution of individual feature groups, we retrained GraphSAGE under four nested feature subsets (see Table III). All configurations share the same architecture, loss, and hyperparameters, and only the input features differ.

TABLE III
FEATURE ABLATION. GRAPHSAGE ACCURACY AND ERROR RATES UNDER PROGRESSIVELY REDUCED FEATURE SETS; λ=5, AVERAGED OVER 3 SEEDS.

Feature Set	Dim	Standard		OOD Swarm	
		Acc.	FPR	Acc.	FNR
Full	13	92.1%	1.8%	91.8%	10.7%
No bound margins	11	93.1%	2.3%	87.9%	16.1%
No prefix features	9	88.8%	2.2%	48.6%	68.1%
Timing only	6	88.3%	3.0%	48.7%	66.6%

The key findings are: (i) *Prefix accumulations are the critical OOD enabler*. Removing all four prefix features collapses OOD accuracy from 91.8% to 48.6% which is near the "always accept" baseline, while in-distribution accuracy degrades only 3.3 percentage points. These features encode the running interference budget that RTA iterates over, and without them the model cannot extrapolate beyond training sizes. (ii) *Bound margins help but are not doing all the work*. Removing only the Liu-Layland and Hyperbolic margins (from 13 to 11 features) reduces OOD accuracy by 3.9 percentage points, yet the model still achieves 87.9% accuracy. The GNN learns patterns beyond what closed-form bounds encode. (iii) In-distribution accuracy is robust across all subsets (88–93%). The differences manifest primarily on OOD generalization, the regime that matters at the edge.

VI. CONCLUSION AND FUTURE WORK

We presented a graph-based admission controller that replaces the pseudo-polynomial latency of exact RTA with predictable, low-variance GNN inference. An inductive GraphSAGE network with 13 RTA-aware node features and worst-task aggregation achieves up to $6.4\times$ speedup at n=100 while maintaining $<$1% OOD FPR, and generalizes from $n\leq$10 training sets to n=100 without retraining. Feature ablation shows that the prefix accumulation features are the main driver of OOD generalization. The analytical bound margins contribute, but removing them costs only 3.9 percentage points of OOD accuracy.

Future work will extend the graph representation to DAG task models for multi-core edge workloads.

ACKNOWLEDGMENT

This work was supported by Ahmedabad University grant number URBSEASE20A6/SUG/20-21/03-SP-08.23.

REFERENCES

[1] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.

[2] M. Joseph and P. Pandya, "Finding response times in a real-time system," *The Computer Journal*, vol. 29, no. 5, pp. 390–395, 1986.

[3] S. Yao, Y. Hao, Y. Zhao, H. Shao, D. Liu, S. Liu, T. Wang, J. Li, and T. Abdelzaher, "Scheduling real-time deep learning services as imprecise computations," in *2020 IEEE 26th Intl. Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2020, pp. 1–10.

[4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[5] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017, pp. 1024–1034.

[6] E. Bini, G. C. Buttazzo, and G. M. Buttazzo, "Rate monotonic analysis: The hyperbolic bound," *IEEE Transactions on Computers*, vol. 52, no. 7, pp. 933–942, 2003.

[7] Z. Guo, "When machine learning and neural networks marry real-time scheduling," *Real-Time Systems*, vol. 61, pp. 320–325, 2025.

[8] S. Baruah, P. Ekberg, and M. Sudvarg, "Learning-assisted schedulability analysis: Opportunities and limitations," *Real-Time Systems*, vol. 61, pp. 332–358, 2025.

[9] Q. Cappart, D. Chételat, E. B. Khalil, A. Lodi, C. Morris, and P. Veličković, "Combinatorial optimization and reasoning with graph neural networks," *Journal of Machine Learning Research*, vol. 24, no. 130, pp. 1–61, 2023.

[10] T. Ridnik, E. Ben-Baruch, N. Zamir, A. Noy, I. Friedman, M. Protter, and L. Zelnik-Manor, "Asymmetric loss for multi-label classification," in *IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021, pp. 82–91.

[11] P. Ekberg, "Rate-monotonic schedulability of implicit-deadline tasks is NP-hard beyond Liu and Layland's bound," in *IEEE Real-Time Systems Symposium (RTSS)*, 2020.

[12] F. Eisenbrand and T. Rothvoß, "Static-priority real-time scheduling: Response time computation is NP-hard," in *IEEE Real-Time Systems Symposium (RTSS)*, 2008.

[13] E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Systems*, vol. 30, no. 1, pp. 129–154, 2005.

[14] P. Emberson, R. Stafford, and R. Davis, "Techniques for the synthesis of multiprocessor tasksets," in *Intl. Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, 2010, pp. 6–11.

[15] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *International Conference on Machine Learning (ICML)*. PMLR, 2019, pp. 3744–3753.