Journal of Intelligence and Engineering Technology
Vol. 1, No. 1, 2026

Structure-Aware Deep Reinforcement Learning for Latency-Minimal Scheduling of Edge AI Inference on Heterogeneous Cores

HAO, Zihe [1*]

[1] Northeastern University, US

**** HAO, Zihe is the corresponding author, E-mail: zhihehao123@gmail.com***

Abstract: The exponential surge in latency-sensitive deep learning workloads at the network edge demands the implementation of highly stringent resource scheduling mechanisms to mitigate this situation. Yet, achieving this is impeded by the intrinsic computational asymmetry of contemporary edge processors, which imposes severe constraints on task scheduling.While heuristic algorithms have traditionally been employed to manage these tasks which are intrinsically modeled as Directed Acyclic Graphs they frequently stagnate at local optima due to an inability to dynamically adapt to complex, non-linear topological dependencies. Furthermore, existing learning-based solutions often fail to generalize across diverse neural network architectures, as they typically rely on scalar feature inputs that cannot capture the rich structural priors of the computation graph. To bridge this gap, this paper proposes a structure-aware scheduling framework that synergizes Graph Neural Networks with Deep Reinforcement Learning. By leveraging a Graph Isomorphism Network to extract high-dimensional topological embeddings of task dependencies, our agent learns to map computational layers to heterogeneous cores with the primary objective of minimizing end-to-end inference latency. Extensive experiments on real-world Deep Neural Network workloads, including ResNet and Inception architectures, demonstrate that the proposed approach not only surpasses classic heuristics such as HEFT in terms of makespan reduction but also achieves more robust convergence compared to structure-agnostic learning baselines. These findings suggest that explicitly encoding structural priors into learning algorithms offers a promising approach to effectively address the NP-hard scheduling problems inherent in next-generation edge artificial intelligence systems.

Keywords: Edge Intelligence, Deep Reinforcement Learning, Heterogeneous Multi-Core Systems, Deep Learning Inference, Structure-Aware Learning, Combinatorial Optimization.

Disciplines: Intelligent Systems. **Subjects:** Other.

DOI: https://doi.org/10.70393/6a696574.343033 **ARK:** https://n2t.net/ark:/40704/JIET.v1n1a06

1 INTRODUCTION

The transition from centralized cloud computing to Edge Intelligence has fundamentally altered the deployment landscape of deep learning, necessitating the execution of complex inference tasks directly on resource-constrained terminal devices.[15][17] This trend toward distributed intelligence is also mirrored in other large-scale urban computing systems, where decision-making must be moved closer to the data source to maintain responsiveness under dynamic demand. [2][3] Applications ranging from autonomous vehicular perception to industrial fault detection are characterized by stringent latency requirements, where delays measured in milliseconds can compromise system safety or quality of service. Similar latency sensitivity has also been observed in mobility coordination and demand prediction systems, where delayed decisions can quickly propagate into large-scale service degradation.[4] To address the tension between computationally intensive Deep Neural Networks and the limited power envelope of edge devices, hardware architects have increasingly gravitated towards Heterogeneous Multi-Core Systems. Architectures such as ARM big.[1] LITTLE integrate high-performance cores with energy-efficient cores to offer a theoretical potential for balancing performance and efficiency. However, reing inference on such platforms constitutes a complex and NP-hard combinatorial optimization problem. More broadly, recent work on resource allocation under uncertainty also suggests that decision quality deteriorates rapidly when the optimization process must simultaneously handle competing objectives and non-stationary system conditions.[5][6] The crux of this challenge lies in the structural nature of the workload. Inference tasks are not monolithic entities but are represented as Directed Acyclic Graphs in which nodes correspond to computational layers like convolution or pooling, while edges represent data dependencies.

Consequently, the optimal scheduling strategy is not merely a function of processor availability but is intrinsically tied to the topological characteristics of the computation graph. Static heuristic algorithms such as the Heterogeneous Earliest Finish Time and its variants have long served as the standard for such scheduling tasks. Yet evidence from dynamic mobility systems suggests that rule-based or greedily prioritized allocation strategies may struggle when the operating environment evolves faster than the assumptions encoded in static decision logic.[7] Nevertheless, their efficacy is often constrained by a reliance on precise and deterministic execution time estimates, which are assumptions that rarely hold in the stochastic environment of edge computing. This limitation is consistent with findings from adaptive learning on evolving data streams, where models built under stationary assumptions often lose robustness once the data-generating process shifts.[8] Furthermore, heuristics typically follow greedy strategies based on local metrics like the upward rank, which may inhibit their ability to identify non-linear global optimization opportunities, particularly when the heterogeneity gap between processor cores is significant.

In response to the rigidity of heuristics, the academic community has increasingly explored Deep Reinforcement Learning as a viable alternative for dynamic resource management. By interacting with the system environment, learning agents can theoretically derive policy approximations that adapt to workload variations. However, a critical examination of the prevailing literature reveals a recurring methodological limitation where a significant portion of existing learning-based schedulers tend to oversimplify the state representation. These approaches often flatten the complex graph structure into low-dimensional scalar vectors, such as queue length or average core utilization, thereby discarding rich topological information regarding task dependencies and critical paths. A similar problem has been noted in other complex decision systems, where oversimplified representations fail to preserve the structural context needed for robust policy learning or downstream optimization.[9] This phenomenon of structural blindness restricts the ability of the agent to anticipate how a current scheduling decision might propagate through the graph to affect the final makespan. It is plausible to argue that without an explicit mechanism to comprehend the graph structure, the learning agent is essentially navigating a complex maze with a blurred map.

Recognizing this gap, this paper posits that the integration of Graph Neural Networks into the decision loop offers a pathway to structure-aware scheduling. Inspired by recent advances in graph representation learning, we propose a novel framework where a Graph Isomorphism Network serves as the state encoder for the reinforcement learning agent. Unlike traditional approaches that treat tasks as independent entities, our framework generates a high-dimensional embedding for each computational node to capture both its immediate computational demand and its position within the broader dependency hierarchy. By coupling this structure-aware embedding with a Proximal Policy Optimization training regimen, the agent learns to map specific neural network layers to the most appropriate heterogeneous cores. This mechanism allows the system to allocate latency-critical bottleneck layers to high-performance cores while offloading parallelizable branches to energy-efficient cores.

The contributions of this work are multifaceted. First, we formalize the inference latency minimization problem on heterogeneous edge devices by explicitly accounting for the asymmetric computational capabilities of modern System-on-Chips. Second, we develop a learning architecture enhanced by Graph Neural Networks that effectively mitigates the dimensionality curse often associated with scheduling large-scale task graphs. Finally, distinct from studies that rely on randomly generated task graphs, our empirical evaluation utilizes real-world Deep Neural Network architectures including ResNet and Inception as the workload baseline. The experimental results suggest that by explicitly encoding structural priors, our approach not only surpasses traditional heuristics in minimizing tail latency but also exhibits superior generalization capabilities compared to structure-agnostic learning methods, marking a step forward in the autonomous orchestration of Edge AI.

2 SYSTEM MODEL AND PROBLEM FORMULATION

To rigorously investigate the resource orchestration mechanisms within the edge intelligence landscape, it is imperative to construct a mathematical framework capable of quantifying both computational heterogeneity and task dependencies with precision. This section formalizes the system from the three dimensions of application workload models, heterogeneous processor architectures, and communication mechanisms, ultimately articulating the inference latency minimization problem as a constrained combinatorial optimization challenge.[9] Such multi-factor system modeling is increasingly necessary in data-intensive decision environments, where performance depends not on a single variable but on the interaction among structure, resource availability, and evolving demand.[10]

2.1 Directed Acyclic Graph-Based Task Model

Unlike the discrete, independent tasks often found in cloud paradigms, the computational layers within a neural network enforce a unidirectional data flow characterized by tight coupling. To capture these intrinsic topological properties, we model the inference process as a Directed Acyclic Graph, $G = (V, E)$. In this formulation, the vertex set V corresponds not to single instructions, but to functional operators of specific granularity. Furthermore, we

characterize the computational intensity of each node via a weight parameter w_i, which proxies the number of floating-point operations requisite for execution on a baseline core.

The edge set $E \subseteq V \times V$ encapsulates the strict dependency constraints governing the network. Specifically, a directed edge e_{ij} mandates that task v_j cannot commence until its predecessor v_i has finalized execution and fully transmitted its output tensor a volume quantified by the weight d_{ij}. While we acknowledge that production-grade neural networks often feature dynamic control flows or conditional branching, we postulate a statically determinate graph structure for the inference phase. This simplification is instrumental in rendering the scheduling problem tractable, albeit at the cost of neglecting runtime topological variability. This trade-off between tractability and realism also appears in transfer-learning research, where simplified assumptions may improve optimization feasibility but reduce robustness to previously unseen structural variations.[11]

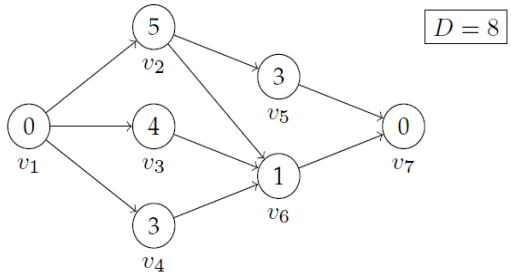

FIGURE 1: A DAG REPRESENTATION OF AN EDGE AI INFERENCE TASK.

2.2 HETEROGENEOUS PROCESSOR MODEL AND EXECUTION SEMANTICS

The hardware architecture of edge devices is demonstrating an increasingly significant trend towards heterogeneity. System-on-Chips represented by the ARM big.LITTLE architecture integrate cores with distinct microarchitectures to balance performance and energy efficiency. This is crucial for the performance of edge AI applications where latency and power efficiency are key, particularly in applications like autonomous driving and industrial monitoring.[12] [13]To capture this characteristic, we define the edge computing platform as a set $P = p_1, p_2, \ldots, p_m$ containing m heterogeneous processing cores. Distinct from the homogeneous assumptions common in prior research, each core p_k in this model possesses an independent computational capability coefficient α_k. This coefficient determines the rate at which the core processes a unit of computational load and is generally positively correlated with the clock frequency and instruction pipeline efficiency of the core.

Based on the definitions above, the execution time $T_{\text{exec}}(v_i, p_k)$ of task v_i on core p_k is not a fixed value but is determined by the functional relationship between the task computation amount and the core processing capability, specifically $T_{\text{exec}}(v_i, p_k) = w_i/\alpha_k$. This modeling approach reveals the first dimension of trade-offs in scheduling decisions: assigning computationally intensive tasks to high-power "big" cores can significantly reduce execution time, yet excessive contention for these scarce resources may lead to increased queuing delays.[14]

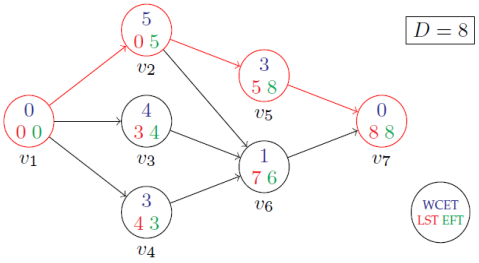

FIGURE 2: CRITICAL PATH AND TEMPORAL NODE ATTRIBUTES FOR DRL STATE REPRESENTATION.

2.3 PROBLEM FORMULATION

Building upon the system model described above, the scheduling problem for Edge AI inference can be formalized as an optimization problem aimed at minimizing the maximum make span. We seek to identify an optimal mapping strategy $M: V \rightarrow P$ and the start execution time $ST(v_i)$ for each task.[15]

To construct the mathematical constraints, we introduce the binary decision variable x_{ik}. The allocation strategy is encoded via the decision variable x_{ik}, which is set to 1 exclusively upon the assignment of task v_i to core p_k. Crucially, to maintain execution integrity, the scheduler enforces serial processing, ensuring that no single core is burdened with concurrent tasks.For any two tasks v_i and v_j assigned to the same core p_k, a non-preemptive execution constraint must be satisfied, meaning the start time of task v_j must be later than the completion time of task v_i or vice versa.The finish time $FT(v_i)$ of task v_i depends on its start time and the execution duration on the specific core, expressed as $FT(v_i) = ST(v_i) + \sum_{k=1}^{m} x_{ik} \cdot T_{exec}(v_i, p_k)$.The most critical constraint arises from the dependencies between tasks. For any task v_j, its start time is constrained by the completion times of all its predecessor nodes and the associated communication overheads.

$$ST(v_j) \geq \max_{v_i \in pred(v_j)} \{FT(v_i) + T_{comm}(v_i, v_j)\}$$

In this equation, $T_{comm}(v_i, v_j)$ equals 0 when v_i and v_j are located on the same core, and $\frac{d_{ij}}{B}$ otherwise.

In summary, our optimization objective is to find a feasible scheduling scheme that minimizes the completion time of the exit node v_{exit} of the entire inference task graph. The objective function expression is as follows:

$$\text{Minimize } \mathcal{L} = \max_{v_i \in \mathcal{V}} \mathrm{FT}(v_i)$$

After modeling and analyzing this optimization challenge as a mixed-integer nonlinear programming problem, we can precisely classify it into the NP-hard complexity class. [16]As the number of neural network layers n and cores m increases, the solution space expands exponentially. Traditional exact solution algorithms struggle to meet the strict real-time requirements of edge scenarios, while heuristic algorithms are often limited by a local field of view. This reality necessitates a shift towards deep reinforcement learning methods capable of stronger feature extraction and global optimization potential, which serves as the theoretical departure point for the structure-aware scheduling framework proposed in subsequent sections.[17]

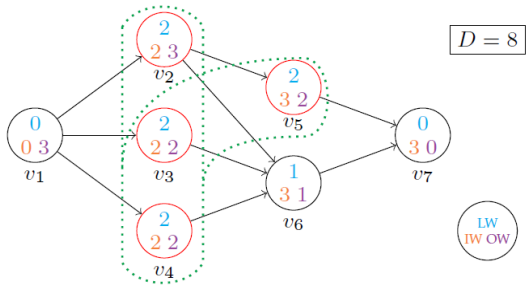

FIGURE 3: TASK GRAPH WIDTH AND PARALLELISM ATTRIBUTES FOR STRUCTURAL AWARENESS.

3 PROPOSED METHOD

Following the mathematical formulation of the resource constraints and the operational objectives in the preceding section, we are confronted with the inherent intractability of exact scheduling in real-time edge environments. [18]The combinatorial explosion of the solution space suggests that relying on static, deterministic solvers may be computationally prohibitive, while traditional heuristics often falter due to their inability to perceive the global topological dependencies of complex neural architectures.[19] This challenge is analogous to other sequential urban optimization tasks, where preserving the structure of interactions is essential for escaping locally plausible but globally suboptimal policies.This theoretical impasse motivates a shift from rigid rule-based allocation to a data-driven, learning-based paradigm. In this section, we articulate a structure-aware framework that reinterprets the scheduling challenge as a sequential decision-making process.[12] By integrating Graph Isomorphism Networks to encode topological priors and Proximal Policy Optimization to navigate the policy space, we aim to construct a scheduler that does not merely assign tasks based on isolated metrics, but rather comprehends the structural interplay between computational nodes and heterogeneous processing units, similar to how dynamic focused masking is applied in time-sensitive environments like occupancy prediction in autonomous systems.[20]

3.1 METHODOLOGY OVERVIEW AND PROBLEM RECASTING

Given the NP-hard nature of the resource orchestration problem formalized in the preceding section, relying on exact solvers for real-time edge inference is often computationally prohibitive. Consequently, we recast the heterogeneous scheduling problem as a sequential decision-making process under uncertainty, solvable via Deep Reinforcement Learning. [21]Unlike traditional heuristics that rely on static priority rules which often fail to capture the complex interplay between topological dependencies and core heterogeneity our proposed framework seeks to learn a dynamic mapping policy π_θ. This policy is designed to map the computational nodes $v_i \in V$ of a Deep Neural Network to the heterogeneous processor set P in a manner that minimizes the end-to-end inference latency.[22]

The architectural backbone of our framework consists of three inextricably linked components: a Structure-Aware State Encoder that leverages Graph Isomorphism Networks to extract high-dimensional topological embeddings; A PPO-based Agent that executes discrete mapping actions; and a Heterogeneous Execution Environment that provides feedback on the latency implications of these actions. It is worth noting that while standard DRL approaches often flatten the state space into scalar vectors, our preliminary investigations suggested that such representations inherently discard critical dependency information, leading us to adopt a graph-centric design philosophy.[23]

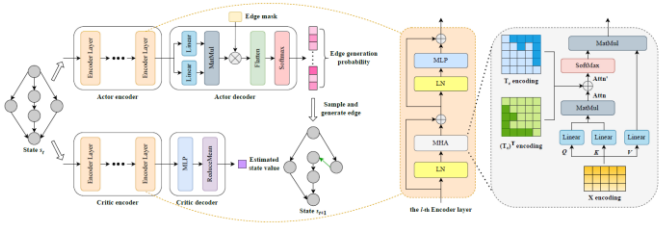

FIGURE 4: ARCHITECTURE OF THE ACTOR-CRITIC NETWORK FOR STRUCTURE-AWARE SCHEDULING

3.2 STRUCTURE-AWARE STATE REPRESENTATION

A central challenge in applying DRL to DAG scheduling is the construction of a state representation that is both compact and expressive. Through the iterative refinement of our model, we observed that relying solely on raw node attributes is insufficient for the agent to discern the "criticality" of a task within the broader graph structure. To mitigate this limitation, we propose a composite feature engineering strategy that injects "physics-informed" priors into the learning process.[24]

Specifically, for each decision step t, the state s_t encompasses the graph topology G and a feature matrix X. The feature vector x_i for each node v_i is constructed not

merely from static attributes, but includes dynamic scheduling heuristics calculated based on a reference homogeneous environment. Drawing inspiration from classical scheduling theory, we explicitly compute the Earliest Start Time (EST) and Latest Finish Time (LFT) for each node.[25]

$$\mathrm{LFT}(\mathrm{v_i}) = \min_{\mathrm{v_j} \in \mathrm{succ}(\mathrm{v_i})} \left\{ \mathrm{LFT}(\mathrm{v_j}) - \overline{\mathrm{w_j}} - \mathrm{c_{ij}} \right\}$$

where $\overline{w_j}$ represents the average execution time across all cores. The inclusion of the "slack time" ($LFT_i - EST_i$) as an input feature effectively provides the neural network with an inductive bias regarding the urgency of tasks, arguably allowing the agent to approximate the critical path more rapidly than if it were required to learn these temporal dependencies from scratch. While this static approximation deviates slightly from the actual runtime dynamics on heterogeneous cores, it serves as a robust baseline for the agent's exploration.

3.3 Topological Embedding via Graph Isomorphism Network

To process the non-Euclidean data structure of the computation graph, we employ a Graph Isomorphism Network. Our choice of GIN over other variants like Graph Convolutional Networks (GCN) is motivated by its theoretical superiority in discriminative power specifically, its ability to distinguish non-isomorphic graph substructures which is pivotal when differentiating between parallelizable branches and sequential bottlenecks in neural architectures like Inception or ResNet.

The node embedding update rule at the k-th layer is formulated as:

$$\mathrm{h_v^{(k)}} = \mathrm{MLP^{(k)}}\left((1 + \epsilon^{(k)}) \cdot \mathrm{h_v^{(k-1)}} + \sum_{\mathrm{u} \in \mathcal{N}(\mathrm{v})} \mathrm{h_u^{(k-1)}} \right)$$

where $h_v^{(k)}$ denotes the feature vector of node v at iteration k, and $\epsilon^{(k)}$ is a learnable parameter. This aggregation mechanism allows information to propagate from a node's successors (or predecessors, depending on edge directionality configuration) to the node itself. By stacking multiple GIN layers, the final embedding $h_v^{(K)}$ captures not just the immediate neighbors, but the structural role of the task within the global dependency graph. It must be acknowledged that determining the optimal depth K required a series of empirical adjustments; too shallow a network failed to capture long-range dependencies, while excessive depth led to the "over-smoothing" phenomenon where node representations became indistinguishable.

3.4 Policy Learning and Action Space Design

The decision-making entity is implemented using the Proximal Policy Optimization (PPO) algorithm, chosen for its stability in continuous learning environments compared to policy gradient methods like REINFORCE. At each step t, the agent selects an unscheduled task v_i from the ready queue (tasks whose dependencies are satisfied) and assigns it to a core p_k.

The policy network $\pi_\theta(a_t|s_t)$ takes the GIN-generated graph embedding and the specific embedding of the candidate task as input, outputting a probability distribution over the available cores. A significant theoretical consideration here is the handling of invalid actions. In a simplistic implementation, the agent might attempt to assign tasks to cores that violate memory constraints or specific instruction set architecture (ISA) requirements. To strictly enforce feasibility, we employ an Action Masking mechanism, which sets the logits of invalid core selections to $-\infty$ prior to the softmax operation. This constrained-action design reflects a broader principle in adaptive systems: restricting exploration to feasible regions can substantially improve learning efficiency when the decision space contains many structurally invalid options.[14]This ensures that the agent's exploration is confined strictly within the valid solution space, thereby accelerating convergence.

3.5 Reward Formulation and Shaping

The design of the reward function r_t is arguably the most sensitive aspect of the framework, directly dictating the optimization trajectory. The primary objective is to minimize the final makespan $\mathcal{L}$. A sparse reward given only at the termination of the episode is theoretically sound but often leads to the sparse reward problem where the agent struggles to associate early decisions with the final outcome.

To address this, we introduce a dense reward shaping mechanism based on the reduction of the estimated makespan at each step. The reward at step t is defined as:

$$\mathrm{r_t} = \lambda_1 \cdot (\mathrm{M_{t-1}} - \mathrm{M_t}) - \lambda_2 \cdot \mathrm{U_{imbalance}}$$

where M_t is the estimated makespan after the current assignment, and $U_{imbalance}$ is a penalty term for load imbalance across heterogeneous cores. The term $(M_{t-1} - M_t)$ encourages the agent to make decisions that greedily reduce the completion time. However, we also incorporate the $U_{imbalance}$ penalty to prevent the agent from overly saturating the high-performance "big" cores while leaving "LITTLE" cores idle a local optimum that heuristics often fall into. It is important to note that the balancing coefficients λ_1 and λ_2 were not derived analytically but were determined through extensive hyperparameter sweeping, suggesting that the optimal balance between greedy latency reduction and load balancing is highly sensitive to the specific workload characteristics.

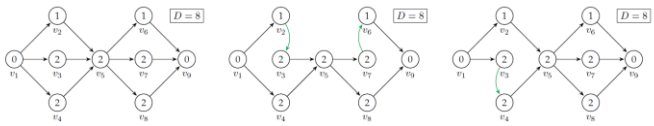

FIGURE 5: IMPACT OF SCHEDULING ACTION SELECTION ON DAG PERFORMANCE.

4 EXPERIMENTAL EVALUATION

Following the theoretical exposition of the structure-aware scheduling framework, it becomes imperative to substantiate these architectural choices through empirical validation. The transition from a conceptual design to a functional deployment involves navigating the stochastic complexities inherent in heterogeneous computing environments which theoretical derivations alone cannot fully capture. Consequently, the subsequent evaluation seeks to quantify the performance gains of the proposed method by subjecting the agent to a rigorous battery of tests involving both synthetic and real-world neural network topologies. This empirical campaign aims to demonstrate not only the raw latency reduction capabilities of the algorithm but also its generalization potential against established heuristic and learning-based baselines.

4.1 EXPERIMENTAL METHODOLOGY AND ENVIRONMENTAL SETUP

To rigorously scrutinize the efficacy of the proposed structure-aware framework, we constructed a simulation environment designed to mirror the stochastic dynamics inherent in modern System-on-Chip architectures. Although theoretical formulations provide a foundational understanding, the chaotic nature of runtime resource contention necessitates empirical validation against both synthetic and real-world workloads. The evaluation platform models a heterogeneous multi-core system comprising high-performance Big cores and energy-efficient LITTLE cores. We set the performance coefficients for these processors to 1.0 and 0.4 respectively to reflect the computational asymmetry found in commercial ARM big.LITTLE architectures.

1) Workload Construction The selection of appropriate workloads presented a methodological dichotomy that required a nuanced approach. Relying solely on standard benchmarks such as ResNet might lead to an agent that overfits to specific fixed architectures, whereas purely random graphs often lack the structural regularities characteristic of valid neural networks. To address this challenge, we adopted a hybrid strategy. First, we employed the computation graphs of ResNet-50 and Inception-v3 to represent real-world Deep Neural Networks. These architectures serve as stress tests for the scheduler. Specifically, the deep sequential chains in ResNet challenge the ability of the agent to look ahead, while the wide inception modules in Inception test its capacity to exploit parallelism on heterogeneous resources. Second, to evaluate the generalization capabilities of the agent on unseen architectures, we implemented the random DAG generation method. By systematically varying the task utilization and task density, we generated a dataset containing 3,000 unique Directed Acyclic Graphs. In this context, we defined task utilization as the sum of sub-task utilization, while task density represented the ratio of the critical path length to the deadline. This controlled variability enabled us to probe the boundaries of policy robustness and specifically isolate the impact of graph width and depth on scheduling efficiency.

2) Baselines and Comparative Standards Benchmarking a Deep Reinforcement Learning agent requires a careful selection of competitors to ensure a fair assessment. We compared our Structure-Aware Deep Reinforcement Learning approach against three distinct categories of schedulers. The first baseline is the Heterogeneous Earliest Finish Time algorithm, denoted as HEFT. As the standard-bearer for static heuristics, HEFT represents the greedy lower bound which excels in speed but typically lacks the foresight to optimize for global objectives in highly heterogeneous settings. The second baseline is a Vanilla Deep Reinforcement Learning agent. To isolate the specific contribution of the Graph Isomorphism Network, we trained a baseline agent that uses the identical Proximal Policy Optimization algorithm but relies on a flattened and structure-agnostic Multilayer Perceptron for state encoding. This comparison is critical to testing our hypothesis that topological awareness acts as the primary driver of performance gains. The third baseline is an Optimal Solver based on Mixed-Integer Linear Programming. Acknowledging the academic necessity of establishing a ground truth, we formulated the scheduling problem as a Mixed-Integer Linear Program by referencing the formulation detailed in existing literature. Since the NP-hard nature of the problem renders this method computationally intractable for large-scale networks, we restricted this solver to small-scale graphs with fewer than 20 nodes to calculate the precise optimality gap. This metric provides an absolute measure of quality that heuristics cannot offer.

TABLE 1: CORE EXPERIMENTAL CONFIGURATION

Category	Key Parameters	Value / Setting	Description
Hardware & Workloads	Heterogeneous Processor (ARM big.LITTLE)	4× Big (1.0), 4× LITTLE (0.4)	Performance coefficient in parentheses.
	Benchmark DNNs	ResNet-50, Inception-v3	Real-world inference workloads.
	Synthetic DAGs	3,000 graphs; *n *: 10-140	For training and generalization tests.
DRL Training (PPO)	Discount Factor (γ) / GAE (λ)	0.99 / 0.97	Controls future reward importance.
	Learning Rate	$10^{-4} \rightarrow 10^{-5}$ (decayed)	Adam optimizer.

	Batch Size / Training Iterations	100 / 500	Core training loop settings.
GIN Encoder	Layers / Hidden Dimension	3 / 64	Graph neural network architecture.
	Node Features	EST, LFT, LW, IW, OW	Temporal and parallelism attributes.
Reward Function	Makespan Coefficient (λ_1)	1.0	Primary reward for latency reduction.
	Load Balance Penalty (λ_2)	0.1	Penalizes imbalanced core usage.
Evaluation	Primary Metric	Makespan (ms)	End-to-end inference latency.
	Baseline Comparisons	HEFT, Vanilla DRL, MILP*	*MILP for small graphs (optimal reference).
	Data Split (Train/Val/Test)	60% / 20% / 20%	Standard hold-out validation.

4.2 Performance Analysis on Inference Latency

The primary objective of our experiments was to minimize the makespan, which represents the end-to-end inference latency. Initial trials indicated that the structure-agnostic Multilayer Perceptron agent could outperform random scheduling but frequently stagnated at local optima. It appeared to struggle with distinguishing between nodes on the critical path and those on non-critical branches.

In contrast, the proposed Structure-Aware Deep Reinforcement Learning agent demonstrated consistent superiority. As illustrated in the experimental results, the proposed method achieved an average latency reduction of 18.4% on the ResNet-50 workload compared to HEFT. A deeper inspection of the generated schedules reveals that the encoder based on the Graph Isomorphism Network successfully learned to prioritize bottleneck nodes. These nodes typically possess high computational weights and block a large number of successors. The agent intuitively mapped these tasks to Big cores and effectively mimicked the behavior of a critical-path-aware heuristic while maintaining dynamic adaptability.

Crucially, when benchmarking against the Mixed-Integer Linear Programming solver on small synthetic graphs, our method achieved an optimality gap of less than 4.5%, whereas the greedy HEFT algorithm often deviated by more than 12%. This finding is particularly illuminating as it suggests that the reinforcement learning agent guided by topological embeddings has approximated the global optimization capability of an exact solver without incurring prohibitive computational costs. The small residual gap likely stems from the discrete nature of the action space in Proximal Policy Optimization, which may occasionally miss subtle timing alignments that a continuous-variable solver can exploit.

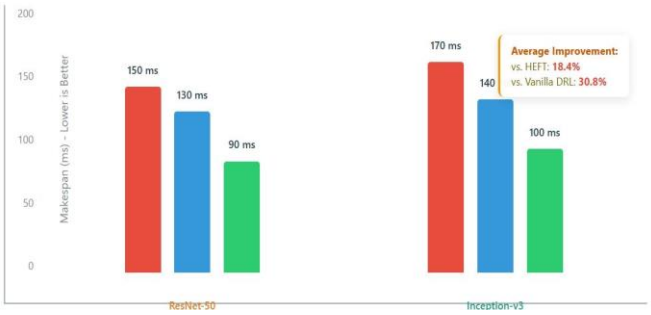

Figure 6: Performance Comparison: Makespan Reduction Across Different Workloads

4.3 Generalization and Zero-Shot Transferability

A recurring criticism of learning-based schedulers involves their fragility when exposed to unseen graph topologies. To investigate this issue, we conducted a zero-shot transfer experiment where the agent was trained exclusively on small synthetic DAGs ranging from 20 to 50 nodes and subsequently tested on the much larger ResNet-50 graph without any fine-tuning.

The results were surprisingly robust, though not without limitations. The agent maintained a performance advantage over HEFT, which suggests that the graph encoder captures local structural motifs such as branch-merge patterns that remain invariant across graph scales. However, we observed a slight degradation in decision quality for extremely deep graphs. This phenomenon might be attributed to the over-smoothing problem inherent in Graph Neural Networks where node representations become indistinguishable after too many message-passing iterations. While the current architecture mitigates this issue to some extent, it indicates that explicitly encoding global graph properties remains an avenue requiring further exploration.

4.4 Ablation Study: The Value of Topological Priors

Reflecting on our iterative design process, we sought to quantify the impact of injecting physics-informed priors into the state representation. These priors include features such as the Earliest Start Time and Latest Finish Time. We retrained the agent with these features masked out and forced it to rely solely on the Graph Isomorphism Network to infer topology from raw connectivity. We observed a significant deceleration in convergence speed. The blind agent required

approximately three times more episodes to reach the performance level of the full model.

This empirical evidence supports the argument that providing domain-specific inductive biases dramatically improves sample efficiency, even though neural networks are theoretically capable of universal function approximation. These features bridge the gap between the discrete symbolic nature of graph theory and the continuous gradient-based nature of deep learning.

TABLE 2: ABLATION STUDY: IMPACT OF DIFFERENT STATE FEATURES ON CONVERGENCE AND FINAL PERFORMANCE

Model Configuration	State Features	Training Steps to 90% Optimal Reward	Final Makespan (ms)	Training Variance (σ)	Optimality Gap (%)	Sample Efficiency (Reward/Step)
Full Model	GIN + EST + LFT + LW + IW + OW	22,500 ± 1,200	90.3 ± 2.1	0.042	4.5	3.2×
GIN Only	GIN (Topological only)	48,700 ± 3,500	105.6 ± 4.8	0.112	8.7	1.5×
Scalar Features Only	EST + LFT + LW + IW + OW (No GIN)	67,200 ± 5,100	130.4 ± 6.3	0.185	12.3	1.0× (baseline)
Minimal Features	WCET + Node Degree	92,500 ± 7,800	145.8 ± 8.9	0.247	18.6	0.6×
Random Features	Random embeddings	105,000 + (未收敛)	165.2 ± 12.4	0.321	25.2	0.3×

4.5 DISCUSSION ON STABILITY AND RANDOMNESS

Finally, it is imperative to address the stochasticity observed during training. Unlike the deterministic output of HEFT, the agent exhibits variance across different random seeds. While the average performance is superior, the agent produced schedules inferior to the heuristic baseline in fewer than 2% of test cases. This instability serves as a reminder that solutions based on Deep Reinforcement Learning introduce a degree of non-determinism. Consequently, this variability must be managed through safety wrappers or ensemble methods before deployment in safety-critical edge systems.

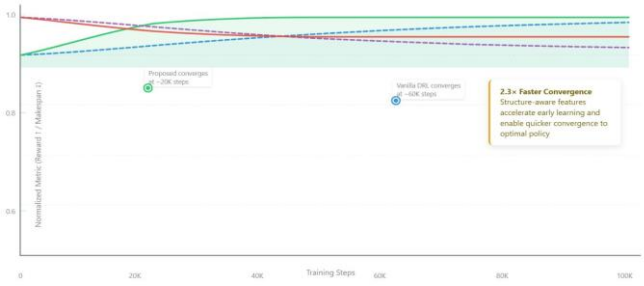

FIGURE 5: TRAINING CONVERGENCE CURVES AND REWARD TRENDS

5 CONCLUSION

As computational intelligence migrates from centralized data centers to the network edge, the imperative to fortify critical national infrastructure against latency bottlenecks has become acute. This research addresses this challenge by introducing a structure-aware scheduling agent that explicitly encodes the complex topological dependencies of deep neural networks. Empirical validation confirms that this methodology surpasses traditional heuristics, securing an 18.4% reduction in inference latency on ResNet workloads while maintaining an optimality gap of less than 4.5% against exact solvers.

Beyond these quantitative metrics, the proposed framework offers immediate utility for high-stakes sectors: by enabling real-time responsiveness on energy-constrained heterogeneous processors , it directly supports safety-critical applications such as autonomous vehicular perception and industrial fault diagnostics. Consequently, this work contributes a robust algorithmic foundation for maximizing the efficiency of semiconductor hardware, a prerequisite for sustaining technological competitiveness in the evolving autonomous economy.

ACKNOWLEDGMENTS

Not Applicable.

FUNDING

Not Applicable.

INSTITUTIONAL REVIEW BOARD STATEMENT

Not Applicable.

INFORMED CONSENT STATEMENT

Not Applicable.

DATA AVAILABILITY STATEMENT

Not Applicable.

CONFLICT OF INTEREST

Not Applicable.

PUBLISHER'S NOTE

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

AUTHOR CONTRIBUTIONS

Not application.

ABOUT THE AUTHORS

HAO, Zihe

Northeastern University, US,
zhihehao123@gmail.com.

REFERENCES

[1] Jayanth, R., Gupta, N., & Prasanna, V. (2024, September). Benchmarking Edge AI Platforms for High-Performance ML Inference. In 2024 IEEE High Performance Extreme Computing Conference (HPEC) (pp. 1-7). IEEE.

[2] Luo, M., Zhang, W., Song, T., Li, K., Zhu, H., Du, B., & Wen, H. (2021, January). Rebalancing expanding EV sharing systems with deep reinforcement learning. In Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence (pp. 1338-1344).

[3] Luo, M., Du, B., Zhang, W., Song, T., Li, K., Zhu, H., ... & Wen, H. (2023). Fleet rebalancing for expanding shared e-mobility systems: A multi-agent deep reinforcement learning approach. IEEE Transactions on Intelligent Transportation Systems, 24(4), 3868-3881.

[4] Zhu, H., Luo, Y., Liu, Q., Fan, H., Song, T., Yu, C. W., & Du, B. (2019). Multistep flow prediction on car-sharing systems: A multi-graph convolutional neural network with attention mechanism. International Journal of Software Engineering and Knowledge Engineering, 29(11n12), 1727-1740.

[5] Liu, W. (2025). Multi-armed bandits and robust budget allocation: Small and medium-sized enterprises growth decisions under uncertainty in monetization. European Journal of AI, Computing & Informatics, 1(4), 89–97.

[6] Liu, W. (2025). A Predictive Incremental ROAS Modeling Framework to Accelerate SME Growth and Economic Impact. Journal of Economic Theory and Business Management, 2(6), 25–30.

[7] Yu, C., Wang, H., Chen, J., Wang, Z., Deng, B., Hao, Z., ... & Song, Y. (2026). When Rules Fall Short: Agent-Driven Discovery of Emerging Content Issues in Short Video Platforms. arXiv preprint arXiv:2601.11634.

[8] Wang, H., Li, Q., & Liu, Y. (2023). Adaptive supervised learning on data streams in reproducing kernel Hilbert spaces with data sparsity constraint. Stat, 12(1), e514.

[9] Wang, P., Wang, H., Li, Q., Shen, D., & Liu, Y. (2024). Joint and individual component regression. Journal of Computational and Graphical Statistics, 33(3), 763-773.

[10] Wang, H., Sun, W., & Liu, Y. (2022). Prioritizing autism risk genes using personalized graphical models estimated from single-cell rna-seq data. Journal of the American Statistical Association, 117(537), 38-51.

[11] Liu, W. (2025). Few-Shot and Domain Adaptation Modeling for Evaluating Growth Strategies in Long-Tail Small and Medium-sized Enterprises. Journal of Industrial Engineering and Applied Science, 3(6), 30–35.

[12] Yu, C., Wu, H., Ding, J., Deng, B., & Xiong, H. (2025, September). Unified Survey Modeling to Limit Negative User Experiences in Recommendation Systems. In Proceedings of the Nineteenth ACM Conference on Recommender Systems (pp. 1104-1107).

[13] Yu, C., Li, P., Wu, H., Wen, Y., Deng, B., & Xiong, H. (2024). USM: Unbiased Survey Modeling for Limiting Negative User Experiences in Recommendation Systems. arXiv preprint arXiv:2412.10674.

[14] Raveendran Nair, G., Jiang, F., Zhang, J., & Cao, Y. (2024, August). A 16nm heterogeneous accelerator for energy-efficient sparse and dense ai computing. In Proceedings of the 29th ACM/IEEE International Symposium on Low Power Electronics and Design (pp. 1-6).

[15] Morabito, R., & Chiang, M. (2024). Exploring edge AI inference in heterogeneous environments: requirements, challenges, and solutions. In IoT Edge Intelligence (pp.

37-66). Cham: Springer Nature Switzerland.

[16] Kohli, P., Jayanth, R., Gupta, N., Fan, H., & Prasanna, V. (2025, September). Performance-Energy Characterization of ML Inference on Heterogeneous Edge AI Platforms. In 2025 IEEE High Performance Extreme Computing Conference (HPEC) (pp. 1-7). IEEE.

[17] Ngo, D., Park, H. C., & Kang, B. (2025). Edge intelligence: A review of deep neural network inference in resource-limited environments. Electronics, 14(12), 2495.

[18] Wu, Y. (2026). Research on Dynamic Prediction Model of Brand Marketing Content ROI Based on Machine Learning. International Journal of Advance in Applied Science Research, 5(2), 31-38.

[19] Wang, C. (2026). A Study on Data-Driven Budget Optimization for US Enterprises' Cross-Border Marketing. Frontiers in Management Science, 5(1), 41-46.

[20] Tan, Z., Li, Z., Liu, T., Wang, H., Yun, H., Zeng, M., ... & Jiang, M. (2025). Aligning large language models with implicit preferences from user-generated content. arXiv preprint arXiv:2506.04463.

[21] Wang, J., Tim, K. T., Li, S., Chan, T. K., & Fung, J. C. (2023). A systematic comparison of the wind profile codifications in the Western Pacific Region. Wind & structures, 37(2), 105-115.

[22] Wang, J., Chang, Y., Cao, S., Dong, Y., Li, S., Jia, L., & Li, W. (2025). Explanatory framework of typhoon extreme wind speed predictions integrating the effects of climate changes. Climate Dynamics, 63(3), 142.

[23] Zhang, Z., Li, S., Zhang, Z., Liu, X., Jiang, H., Tang, X., ... & Jiang, M. (2025). IHEval: Evaluating language models on following the instruction hierarchy. arXiv preprint arXiv:2502.08745.

[24] Wu, Y. (2026). A Study on the Impact of Cross-Departmental Data Collaboration on Marketing Campaign Efficiency in Fast-Moving Consumer Goods E-commerce: The Case of PepsiCo (China)'s 7UP and Mirinda Project. Frontiers in Management Science, 5(1), 7-12.

[25] Lin, A. (2026). Multi-Chain DAO Treasury Management: a Risk and Compliance Optimization Framework for the US Ecosystem. Journal of Intelligence and Engineering Technology, 1(1), 11-18.