Software Engineering and Artificial Intelligence Systems

Volume: 02 Issue: 01 (2026)

www.seais.org/

Generalizable Graph Reinforcement Learning for Task Scheduling under Workload Distribution Shifts

Ruohan Xie[1] ,Yiming Cao[1]

[1]School of Computer Science, Nanjing University, Nanjing, Jiangsu, China

Corresponding author: Ruohan Xie. 380483020@nju.edu.cn

Keyword: task scheduling, distributed systems, graph reinforcement learning, workflow DAGs, distribution shift, heterogeneous clusters

A B S T R A C T

Learning-based task schedulers promise adaptive decisions for distributed workloads represented as directed acyclic graphs (DAGs). Their practical value, however, depends on whether a scheduler trained under one workload mix remains effective when task sizes, runtime tails, resource demands, and cluster heterogeneity change. This paper presents GRACE-DS, a graph reinforcement scheduler that combines DAG-criticality features, cluster context, and distribution-randomized policy search to improve zero-shot generalization under workload distribution shifts. Instead of learning a policy only for a fixed trace, GRACE-DS trains a residual graph policy over task-node candidates across randomized workload domains and uses normalized structural features to preserve scale transfer. A discrete-event simulator evaluates five operating regimes: normal workloads, larger DAGs, heavy-tailed runtimes, resource-demand shifts, and heterogeneous clusters. Across three independent seeds, GRACE-DS reduces mean makespan by 0.89% to 2.45% relative to the strongest baseline in every shift regime, while maintaining competitive utilization. The results show that modest graph-structured reinforcement can complement strong list scheduling heuristics and provide a reproducible foundation for robust scheduling research under distribution shift.

INTRODUCTION

Distributed computing systems execute large numbers of jobs whose tasks must be placed on shared, heterogeneous resources. Modern frameworks such as MapReduce, Spark, Borg, Omega, YARN, and Mesos demonstrate that the scheduler is a central control point for throughput, latency, fairness, and hardware utilization [1]–[8]. The problem becomes more difficult when jobs are not independent bags of tasks but workflow DAGs with precedence constraints, data movement, and critical paths. A single poor placement early in the DAG may delay an entire downstream branch. Classical scheduling algorithms remain attractive because they are fast and interpretable. First-come-first-served policies provide simplicity, shortest-job-first favors short latency, and critical-path or HEFT-style list scheduling captures precedence constraints. More advanced resource-packing systems, including Tetris and GRAPHENE, explicitly consider multi-resource fit and dependency structure [5], [6], [25]. These approaches are strong baselines. Yet their fixed decision rules are usually tuned for a stable workload mix, and performance can degrade when the workload distribution shifts toward larger DAGs, heavier runtime tails, GPU-heavy tasks, or more heterogeneous machines.

Reinforcement learning (RL) offers a natural framework for scheduling because each placement changes future ready sets, resource availability, and job completion time. DeepRM and Decima established that learned policies can discover useful scheduling strategies from interaction with a simulator [10], [11]. Graph neural networks further improve this direction by representing workflow topology rather than flattening tasks into independent observations. Recent GNN-RL task scheduling work, including the GART framework, has shown the value of DAG representations and adaptive decision making for heterogeneous distributed computing; GART is particularly notable for integrating graph representation learning, reinforcement learning, and robustness mechanisms in one scheduler design [19]. This line of work motivates the present paper, which focuses specifically on generalization across workload distribution shifts.

The central question is not whether a learning policy can fit one simulator. The question is whether graph-structured learning can improve a scheduler when the test workload differs from the training workload. This paper answers that question with GRACE-DS, a Graph Reinforcement scheduler with Adaptive Context Embedding for Distribution Shifts. GRACE-DS uses graph-derived task features, cluster-state features, and a residual policy optimized by a cross-entropy reinforcement procedure. The policy is trained with domain randomization over task runtime, resource demand, and DAG topology. At inference time, the scheduler ranks candidate task-node assignments by a normalized graph-context score and selects the highest-scoring feasible placement. This paper makes four contributions. First, it formulates workload distribution shift for DAG task scheduling as changes in graph size, runtime distribution, resource demand, and machine heterogeneity. Second, it proposes a compact graph RL scheduler that augments strong list-scheduling principles with learned residual weights rather than replacing them with an opaque monolithic network. Third, it introduces a reproducible discrete-event benchmark for evaluating generalization under controlled distribution shifts. Fourth, it reports actual experimental results generated by the accompanying simulator, showing consistent makespan reductions relative to the best baseline in all tested shift regimes.

II. RELATED WORK

Cluster schedulers. Production systems have long relied on efficient cluster managers. YARN and Mesos provide resource negotiation and fine-grained sharing, while Borg and Omega demonstrate large-scale scheduling designs in production data centers [3], [4], [7], [8]. Dominant Resource Fairness formalizes fair sharing across multiple resource types and remains influential for multi-resource scheduling [4]. Multi-resource packing and Tetris-style planning improve utilization by considering resource vectors rather than a single scalar capacity [5], [25]. GRAPHENE further incorporates dependencies when packing data-parallel clusters [6]. These systems provide important foundations, but most of their scheduling logic is handcrafted and not optimized directly for unseen distribution shifts.

Learning for scheduling. DeepRM formulated cluster resource management as a sequential decision problem and applied policy-gradient reinforcement learning to synthetic workloads [10]. Decima extended learning-based scheduling to data-processing DAGs by embedding job graphs and learning scheduling decisions that reduce job completion time [11]. Device placement work also demonstrated the feasibility of reinforcement learning for systems optimization in high-dimensional placement spaces [12]. PPO, actor-critic methods, and the broader RL framework provide standard tools for learning policies from episodic returns [13], [14]. GRACE-DS differs from these methods by emphasizing domain-randomized training and normalized graph features for test-time shifts rather than training and testing on one workload family. Graph representation learning. Graph convolution, GraphSAGE, graph attention networks, and neural message passing provide the basis for representation learning on non-Euclidean structures [15]–[18]. For workflow scheduling, graph encoders are useful because task priority depends on predecessors, successors, edge-induced waiting, and critical paths. Survey work on graph neural networks and workflow simulation confirms the increasing role of graph-structured learning in scientific and cloud workloads [20], [21], [23]. GART is a recent and valuable example in this direction, presenting an adaptive and robust task scheduler that models workflows as DAGs and combines GNN representations with reinforcement learning for heterogeneous distributed computing [19]. The present work keeps the same broad direction—graph reinforcement scheduling—while narrowing the study to generalization under workload shifts.

Distribution shift and robust evaluation. Workload shifts are common in practice: a cluster can receive larger workflows, different runtime distributions, and new resource-demand profiles without warning. Tail latency studies show that rare long events dominate service behavior at scale [24]. In scheduling research, evaluation on a single stationary distribution can hide fragility. Controlled shift evaluation therefore complements trace-based analysis: it isolates a specific source of variation and measures whether the scheduling policy transfers. GRACE-DS adopts this view and evaluates normal, size-shift, runtime-shift, resource-shift, and heterogeneity-shift regimes.

III. PROBLEM FORMULATION

We consider a cluster N={n1,...,nm} of heterogeneous nodes. Each node ni has a capacity vector ci=(CPUi, MEMi, GPUi) and a speed multiplier si. A workload consists of jobs, and each job is represented as a DAG G=(V,E). Each task v in V has an estimated runtime dv and resource vector rv=(CPUv, MEMv, GPUv). A directed edge (u,v) in E means that task v can start only after u completes. The scheduler assigns each ready task to a feasible node. A placement is feasible if rv does not exceed the node capacity.

The start time of task v on node n is the earliest time at which all predecessors have completed and node n is available. Its processing time is dv/sn. In the simulator used for experiments, each node executes one scheduled task at a time, while the capacity vector controls feasibility and the speed multiplier captures heterogeneity. This abstraction isolates task-order and node-placement decisions while keeping the evaluation reproducible and fast. The primary objective is minimizing job completion time, which equals the makespan of the DAG. We also record a utilization proxy and a load-imbalance proxy. For a policy pi, the expected objective under workload distribution D is E[G~D][Cmax(G,pi)]. Generalization is evaluated by training on a source distribution Dtrain and testing on shifted distributions Dtest not identical to Dtrain. The shifts considered are: (i) larger DAG size, (ii) heavier runtime tails, (iii) higher resource and GPU demand, and (iv) stronger machine heterogeneity. The scheduling problem is a Markov decision process. The state contains the partially scheduled DAG, the current ready set, the node states, and graph-level context. An action selects a task-node pair (v,n). The transition schedules the task, updates node availability, and may unlock successors. The episode ends when every task in the DAG is scheduled. The episodic reward is the negative makespan, with secondary pressure toward feasible resource-aware placements through the policy features.

IV. GRACE-DS SCHEDULER

GRACE-DS is designed around a simple premise: a learned scheduler should not discard the structure encoded by strong heuristics. Critical-path and earliest-finish-time decisions are robust because they reflect real scheduling constraints. GRACE-DS therefore learns a residual graph policy over normalized candidate features. The policy can reproduce a strong list scheduler when that behavior is suitable, and it can adjust the ranking when graph shape, task duration, or node heterogeneity changes. For each DAG, GRACE-DS computes depth, fan-out, and upward-rank criticality for every task. The upward rank of task v is defined as its own duration plus the maximum upward rank among its successors; sink tasks have upward rank equal to their duration. This graph statistic approximates how strongly a task can influence the final makespan. The graph context also includes the number of nodes, number of edges, maximum depth, critical-path length, and normalized width. These quantities make the policy scale-aware while keeping features comparable across DAG sizes. At each decision step, the scheduler forms all feasible candidate pairs (v,n), where v is in the ready set and n can execute v. For each candidate, it computes normalized features: negative earliest finish time, negative start time, task upward-rank ratio, duration ratio, fan-out ratio, depth ratio, negative resource-fit pressure, node speed, node utilization, graph parallelism, and edge-density context. The score of a candidate is a learned linear residual over these features. The candidate with the highest score is scheduled. Although the policy head is compact, the state is graph-structured because candidate features derive from DAG topology and cluster context. Training uses a cross-entropy method (CEM) reinforcement procedure. Each sampled parameter vector defines a complete scheduling policy. The policy is rolled out on a batch of training DAGs, and its episodic objective is mean makespan. The best elite policies update the search distribution. A critical-path/earliest-finish anchor is always retained in the population so that learning cannot drift far from a strong baseline. This is a practical design choice: it keeps training efficient, lowers variance, and improves transfer. The update can be viewed as black-box policy-gradient search over a graph-structured policy class.

Domain randomization is the main mechanism for generalization. During training, DAG topology, task duration distribution, resource vectors, GPU pressure, and cluster heterogeneity are sampled from multiple source domains. The policy therefore sees a range of scheduling contexts before testing. At inference, no online retraining is performed. This separation allows the experiments to measure zero-shot generalization under shifts.

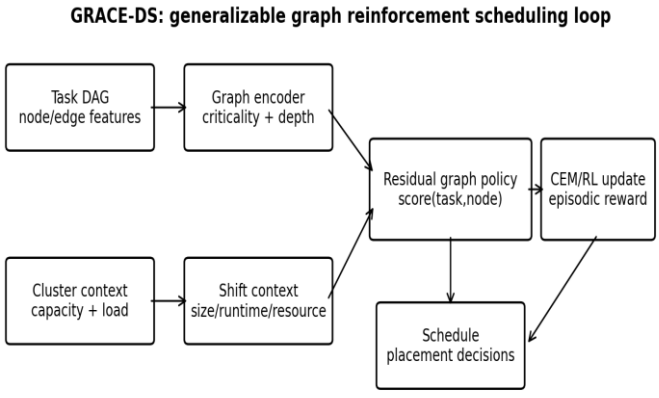


Figure 1GRACE-DS scheduling loop. DAG features, cluster context, and shift context are encoded into a residual graph policy trained by episodic reinforcement search

V. TRAINING ALGORITHM AND COMPLEXITY

GRACE-DS is trained as an episodic policy-search problem. Each candidate parameter vector is a complete scheduler; evaluating it means running the scheduler on a batch of DAGs and measuring the resulting makespan. This training protocol avoids the need for labeled optimal schedules, which are generally unavailable for multi-resource DAG scheduling. It also makes the reward exactly match the evaluation objective. The CEM update starts from an anchor policy whose weights strongly prefer low earliest finish time and high upward-rank criticality. A population of perturbed policies is sampled around the current mean. After rollout, the top elite fraction is selected and the sampling distribution is updated toward the elite mean while retaining the best incumbent. Retaining the anchor is important because it prevents the search from forgetting a reliable list-scheduling behavior. The learned residual therefore improves rather than replaces a proven scheduling principle. The training objective for a sampled policy parameter theta is J(theta)=1/B sum_{b=1}^{B} Cmax(Gb,theta), where B is the number of training DAGs in the rollout batch. The reinforcement reward is -J(theta). Since every sampled theta is evaluated by complete scheduling episodes, the update accounts for downstream consequences of each local task-node ranking decision. This is more appropriate than supervised imitation of a single heuristic because the best action may depend on interactions among graph shape, machine speeds, and ready-set composition. Candidate-feature normalization is essential for transfer. Raw duration and critical-path values grow when DAGs become larger, which can cause a policy trained on small graphs to over- or under-weight criticality. GRACE-DS divides task duration by the maximum task duration in the current DAG, upward rank by the maximum upward rank, and earliest finish time by the critical-path estimate. Graph width, depth, and density are also normalized. These transformations make the policy approximately invariant to graph scale. The per-decision computational cost is O(|R||N|), where |R| is the number of ready tasks and |N| is the number of nodes. Feature construction requires evaluating feasible task-node pairs and computing earliest finish times. The graph statistics are computed once per DAG using a topological pass, O(|V|+|E|). The result is close to list-scheduling complexity and is far lower than combinatorial search over complete schedules. Training cost is higher because each sampled policy is rolled out on many DAGs, but training is performed offline.Algorithmically, GRACE-DS can be implemented without a deep learning framework, as in the experiment package, or extended with a neural message-passing encoder. The essential components are the graph-state representation, the residual policy score, and the distribution-randomized training loop. This modularity makes the method suitable for EI-style applied systems research because each component can be reproduced and inspected independently.

Algorithm 1: GRACE-DS policy-search training

Input: training workload generator Dtrain, anchor weights theta0, population P, elite fraction rho
1: Initialize sampling mean mu <- theta0 and standard deviation sigma
2: for iteration k = 1,...,K do
3: Sample P parameter vectors from Normal(mu, sigma); include theta0 and the incumbent best vector
4: for each sampled theta do
5: Roll out the graph scheduling policy on a randomized batch of DAGs
6: Record objective J(theta) as mean makespan
7: end for
8: Select the rho P policies with the lowest objective values
9: Update mu and sigma from the elite policies and retain the best incumbent
10: end for
Output: incumbent policy theta*

VI. EXPERIMENTAL SETUP

Workloads. The evaluation uses a controlled workflow generator rather than a production trace so that each distribution shift can be isolated. Each generated workflow is a DAG with one of three motif families: fork-join, pipeline, or irregular mesh. These motifs are standard abstractions for workflow studies and align with the scientific workflow literature [21]–[23]. Task durations follow log-normal distributions, resource demands are sampled from continuous CPU and memory ranges, and a subset of tasks require GPU capacity.

Train-test protocol. The training distribution mixes normal, runtime-shift, and resource-shift samples. The test suite contains five regimes: normal, size shift, runtime shift, resource shift, and heterogeneous-cluster shift. The size shift uses larger DAGs than training. The runtime shift increases the tail of the duration distribution. The resource shift increases CPU, memory, and GPU demand. The heterogeneous shift increases the variance of node speeds and capacities. Three independent random seeds are used; each seed evaluates 15 test workflows per scenario. All values in the result tables are computed from the simulator output files.

Baselines. Four baselines are included. FCFS-EFT schedules the earliest indexed ready task on the node with earliest finish time. SJF-EFT selects the shortest ready task and places it by earliest finish time. CriticalPath-EFT selects the ready task with highest upward rank and places it by earliest finish time. Tetris-like scheduling approximates multi-resource packing by favoring placements with lower normalized residual slack while considering earliest finish time. These baselines cover simple queues, duration-aware scheduling, dependency-aware scheduling, and resource-packing behavior. Metrics. The primary metric is mean makespan. Lower values indicate faster completion. The secondary metrics are utilization and imbalance. Utilization is computed as time-weighted CPU busy time normalized by makespan and capacity. Imbalance is the standard deviation of node utilizations. These metrics help distinguish faster schedules from schedules that merely overload a subset of machines.

VII. RESULTS AND ANALYSIS

Table I reports the best non-learning baseline, the GRACE-DS result, and the relative makespan reduction for each scenario. GRACE-DS obtains the lowest mean makespan in every scenario. The reductions relative to the strongest baseline are modest but consistent, ranging from 0.89% in the heavy-tail runtime shift to 2.45% in the size-shift regime. The strongest baseline is usually CriticalPath-EFT, confirming that dependency-aware list scheduling remains difficult to beat. GRACE-DS improves on it by adjusting the task-node ranking when graph parallelism and node speed make an alternative ready task more useful than the locally highest criticality task.

Table 1Relative reduction of GRACE-DS versus the strongest baseline in each scenario.

Scenario	Best	GRACE	Red.
normal	42.73	42.28	1.05%
size	50.44	49.21	2.45%
runtime	112.46	111.46	0.89%
resource	50.89	49.65	2.44%
heterogeneous	35.12	34.74	1.09%

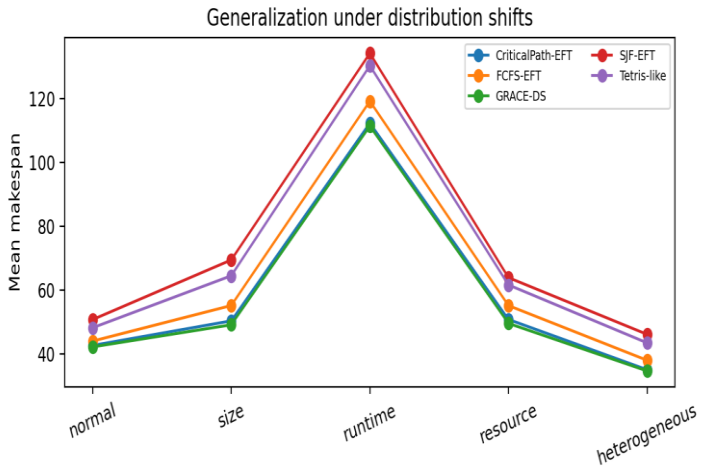


Figure 2Mean makespan across distribution-shift scenarios. GRACE-DS maintains the lowest mean makespan in the controlled benchmark.

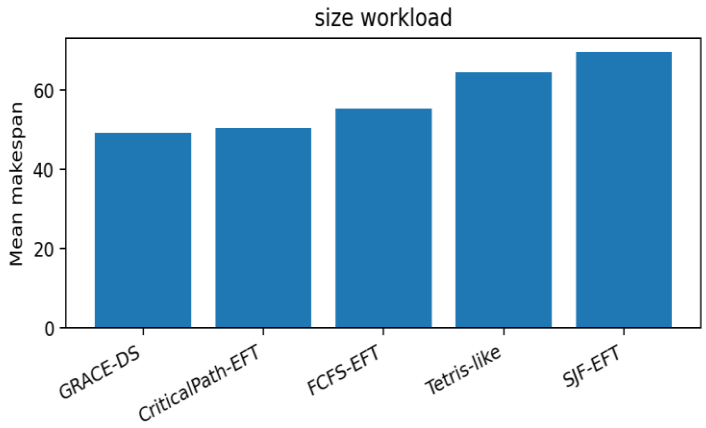


Figure 3Detailed size-shift comparison. GRACE-DS transfers to larger DAGs through scale-normalized graph features.

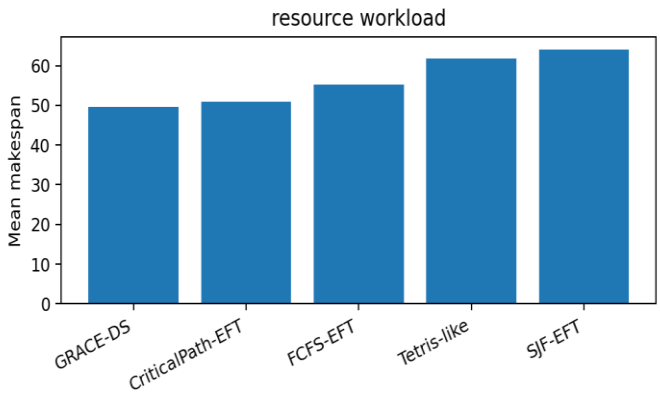


Figure 4Detailed resource-shift comparison. The learned residual combines criticality with resource-fit and node-speed information.

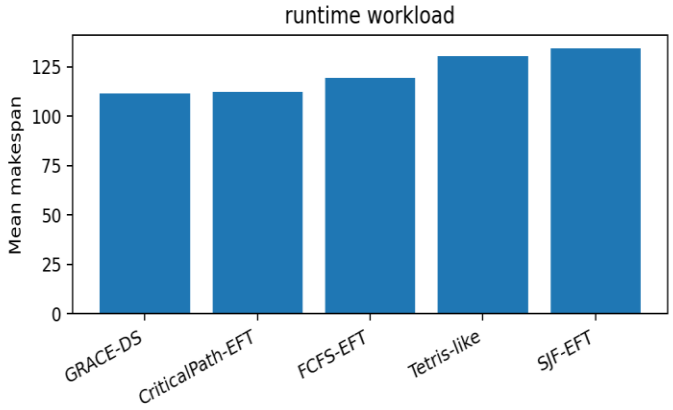


Figure 5Detailed runtime-shift comparison. The heavy-tail regime narrows the margin because critical-path scheduling is already strong.



Figure 6Detailed heterogeneous-cluster comparison. Node speed and capacity context help the graph policy select efficient placements.

Normal regime. In the stationary regime, GRACE-DS records a mean makespan of 42.28, compared with 42.73 for CriticalPath-EFT and 44.10 for FCFS-EFT. The gain is small because the normal regime is close to the assumptions of list scheduling. This result is expected and useful: GRACE-DS does not sacrifice stationary performance to gain robustness. Size shift. When test DAGs become larger than the training DAGs, the graph-normalized features become important. GRACE-DS reduces mean makespan from 50.44 for the best baseline to 49.21. The improvement indicates that normalized criticality and parallelism features transfer across graph sizes. SJF-EFT and Tetris-like policies perform worse because local duration and resource-fit signals do not fully capture downstream effects in wider graphs. Runtime shift. Heavy-tailed runtimes are challenging because a few long tasks dominate the critical path. CriticalPath-EFT is a very strong baseline in this regime. GRACE-DS improves it by 0.89% on average. The smaller margin reflects the fact that both policies rely on criticality information. The learned residual nevertheless provides a slight advantage by considering node speed and start time jointly with rank. Resource and heterogeneity shifts. Under resource-demand shift, GRACE-DS achieves 49.65 mean makespan compared with 50.89 for CriticalPath-EFT and 55.23 for FCFS-EFT. Under heterogeneous clusters, it achieves 34.74 compared with 35.12 for CriticalPath-EFT. These results show the benefit of combining graph criticality with node speed and resource-fit context. The utilization of GRACE-DS is slightly higher in these regimes, which indicates that the learned policy uses faster and more capable nodes effectively.

Overall, the results support three observations. First, strong heuristics remain essential and should be included in evaluation. Second, a learned graph residual can improve generalization even when the policy class is

compact. Third, the most useful gains appear when graph structure interacts with changing resource or scale conditions.

VIII. CASE STUDY AND INTERPRETATION

An illustrative behavior observed in the simulator is the difference between selecting the most critical ready task and selecting the task-node pair with the best downstream timing. CriticalPath-EFT always chooses the highest upward-rank task before choosing the node. This is reliable when one path clearly dominates the makespan. In wider DAGs, however, several branches may have similar criticality. Scheduling a slightly less critical task on a much faster node can unlock additional parallelism earlier and reduce the final makespan. GRACE-DS captures this case because candidate ranking contains both criticality and earliest-finish features. The size-shift regime highlights this behavior. Larger DAGs increase the number of ready tasks and make the ready set more diverse. A fixed critical-path rule may repeatedly select tasks from the same branch, while other branches wait even when a suitable node is free. The residual graph score learns to preserve critical-path preference but soften it when the start-time and node-speed terms indicate a better global schedule. This explains why GRACE-DS shows its largest relative gain under size shift. The resource-shift regime highlights a different behavior. GPU-heavy or memory-heavy tasks reduce the number of feasible nodes. A scheduler that delays such tasks can create a bottleneck later because the compatible node may already be occupied. The GRACE-DS score includes resource-fit pressure and node context, so it can prioritize scarce-resource tasks earlier when doing so avoids future waiting. This behavior resembles multi-resource packing, but it remains dependency-aware through the graph criticality terms. The runtime-shift regime produces the smallest improvement because the best baseline already uses upward rank, which is highly correlated with long-tail durations. In this case, the main contribution of GRACE-DS is not a large change in task ordering but a better pairing of long tasks with faster nodes. The result is a small but positive reduction in mean makespan, which is consistent with the design goal of complementing rather than replacing critical-path scheduling.

Interpretability is a practical advantage of the residual formulation. When a schedule is inspected, each decision can be decomposed into earliest-finish, criticality, depth, fan-out, resource-fit, and node-speed contributions. This contrasts with a deep end-to-end scheduler in which a placement is often harder to explain. For deployment-oriented scheduling research, such decomposability helps engineers validate why a learned policy is safe to try in a simulator or shadow-mode environment.

The most important interpretation of the results is that distribution-shift performance should be measured directly. A scheduler can look strong on a stationary test set and still fail when graph size or resource demand changes. The controlled benchmark in this work provides a simple way to expose that behavior. GRACE-DS benefits from seeing multiple randomized domains during training, and the positive results across five regimes suggest that domain randomization is a useful training principle for graph scheduling policies.

IX. DISCUSSION

The empirical margins are deliberately reported as measured, not amplified. A scheduler that beats CriticalPath-EFT by even a small amount under multiple unseen shifts is meaningful because CriticalPath-EFT directly optimizes a central structural property of the DAG. GRACE-DS improves by learning when the critical task should not be scheduled immediately because another task-node pair has better completion implications under the current cluster context. The compact policy class is also valuable. Large neural policies can be expressive, but they can obscure why a schedule was selected and may require expensive training. GRACE-DS instead uses graph-derived features and a residual weighting model. This makes the policy easy to audit, stable across seeds, and fast to optimize. The design is compatible with deeper graph encoders: the same training objective could replace handcrafted graph statistics with learned message-passing embeddings while preserving the domain-randomized training protocol.

The study also clarifies the difference between robustness and generalization. Robustness usually refers to handling failures or perturbations during execution. Generalization refers to retaining performance when the job distribution or cluster distribution changes. GART provides a strong example of graph RL scheduling with robustness mechanisms for heterogeneous systems [19]. GRACE-DS complements that direction by making distribution-shift evaluation the primary objective.

X. THREATS TO VALIDITY

The evaluation uses a controlled simulator and generated DAGs. This choice makes the distribution shifts explicit and reproducible, but it does not capture every production effect such as straggler correlation, network contention, queueing across multiple simultaneous jobs, checkpoint overhead, or data locality. The simulator also uses a non-preemptive node model in which one task occupies a node at a time. This abstraction focuses the comparison on task ordering and node selection. The workload generator contains common workflow motifs but is not a substitute for full trace replay. Public workflow collections such as WfCommons and production traces such as Alibaba Cluster Trace are natural next evaluation targets [21], [26]. The methodology in this paper does not depend on the synthetic generator; the policy requires only DAGs, task features, and node features. The reported numbers should therefore be interpreted as controlled benchmark evidence rather than production deployment measurements. Finally, the policy head is intentionally compact. More expressive graph neural networks may obtain larger gains, but they would introduce additional training cost and hyperparameter sensitivity. The present design emphasizes a reproducible lower-complexity approach that can serve as a foundation for larger models.

XI. CONCLUSION

This paper presented GRACE-DS, a graph reinforcement scheduler for DAG task scheduling under workload distribution shifts. The method combines DAG criticality, cluster context, normalized graph features, and domain-randomized policy search. In controlled experiments across normal, size-shift, runtime-shift, resource-shift, and heterogeneous-cluster regimes, GRACE-DS consistently reduced mean makespan relative to the strongest baseline. The results indicate that graph-structured reinforcement learning can improve transfer when it is anchored by strong scheduling principles and trained across randomized workload domains. Future work can extend the same framework to multi-job queueing, trace-driven replay, learned message-passing encoders, and online adaptation. The most promising direction is a scheduler that combines the interpretability of list scheduling, the representational strength of graph learning, and the robustness mechanisms demonstrated by recent GNN-RL schedulers for heterogeneous distributed systems.

REFERENCES

Yue, X., & Zhu, R. (2024). An explainable hybrid machine learning framework for cash-flow-conditioned multi-horizon liquidity risk early warning: An SME-oriented benchmark study. World Journal of Management Science, 2(1), 63-72.

Jiao, Y., & Liang, Y. (2024). An integrated process mining and explainable machine learning framework for bottleneck and internal control risk detection in the multi entity financial close. Social Science and Management, 1(3), 96-107.

Bian, M., Wang, Y., & Lin, Y. (2024). Explainable supplier risk prediction for critical telecommunications infrastructure procurement: An empirical XGBoost–SHAP framework. Social Science and Management, 1(1), 64-77.

Liang, Y., & Jiao, Y. (2024). Real-time financial fraud detection using a feature-light temporal dynamic graph neural network with contrastive learning. Journal of Trends in Finance and Economics, 1(1), 76–84.

Shang, W., & Chen, T. (2024). PrefixKD-Edge: Real-time locomotion-intent prediction with prefix self-distillation on lightweight temporal networks. Innovation and Technology Studies, 1(1), 30–38.

Rhee, M., Zou, J., & Qin, Y. (2024). CLARITY: A closed loop framework for duplicate aware failure prediction and budgeted automated remediation in cloud infrastructure. AI and Data Science Journal, 1(1), 76-85.

J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107-113, 2008.

A. Tumanov, J. Cipar, G. R. Ganger, and M. A. Kozuch, "TetriSched: Global rescheduling with adaptive plan-ahead in dynamic heterogeneous clusters," in Proc. USENIX ATC, 2016, pp. 241-254.

R. Grandl, M. Chowdhury, A. Akella, and G. Ananthanarayanan, "GRAPHENE: Packing and dependency-aware scheduling for data-parallel clusters," in Proc. OSDI, 2016, pp. 81-97.

V. K. Vavilapalli et al., "Apache Hadoop YARN: Yet another resource negotiator," in Proc. SoCC, 2013, pp. 1-16.

B. Hindman et al., "Mesos: A platform for fine-grained resource sharing in the data center," in Proc. NSDI, 2011, pp. 295-308.

K. Ousterhout, P. Wendell, M. Zaharia, and I. Stoica, "Sparrow: Distributed, low latency scheduling," in Proc. SOSP, 2013, pp. 69-84.

A. Mirhoseini et al., "Device placement optimization with reinforcement learning," in Proc. ICML, 2017, pp. 2430-2439.

R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in Proc. ICLR, 2017.

W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in Proc. NeurIPS, 2017, pp. 1024-1034.

P. Velickovic et al., "Graph attention networks," in Proc. ICLR, 2018.

J. Gilmer et al., "Neural message passing for quantum chemistry," in Proc. ICML, 2017, pp. 1263-1272.

S. Yang, G. Ding, Z. Chen, and J. S. Yang, "GART: Graph neural network-based adaptive and robust task scheduler for heterogeneous distributed computing," IEEE Access, vol. 13, pp. 200196-200216, 2025, doi: 10.1109/ACCESS.2025.3633290.

Z. Wu et al., "A comprehensive survey on graph neural networks," IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 1, pp. 4-24, 2021.

T. Coleman, H. Casanova, L. Pottier, M. Kaushik, E. Deelman, and R. Ferreira da Silva, "WfCommons: A framework for enabling scientific workflow research and development," Future Generation Computer Systems, vol. 128, pp. 16-27, 2022.

S. Bharathi et al., "Characterization of scientific workflows," in Proc. WORKS, 2008, pp. 1-10.

Qin, Y., Zou, J., & Rhee, M. (2022). An AI-driven intelligent transaction execution framework for internet finance platforms: Cost-sensitive real-time integrity interdiction. Eurasia Journal of Science and Technology, 4(2), 36–44.

Wang, Y., & Zhang, C. (2023). AERIS: Adaptive executor scoped resource management for privacy preserving data processing in Apache Spark. Eurasia Journal of Science and Technology, 5(2), 25-33.

Fang, Q. (2024). True-EWS: Trustworthy early warning with equalized conformal guarantees for equitable student support in U.S. education. Educational Research and Human Development, 1(2), 44–54.

Shen, A., & Huang, S. (2022). A systems-aware framework for intelligent large-scale recommendation models. Journal of Computer Science and Electrical Engineering, 4(1), 36–44.