MULTIDISCIPLINARY RESEARCH IN COMPUTING INFORMATION SYSTEMS

VOL 05 ISSUE 10 2025

https://mrcis.org

Adaptive Workflow Scheduling in Heterogeneous GPU Clusters via Deep Reinforcement Learning

Zixuan Li,[1] Yuefeng Chen*,[1] and Thomas Gallagher[1]
[1]*Department of Computer Science and Engineering, University of Colorado Denver, USA*
***Corresponding author:** *yuefeng.2001@gmail.com*

Abstract: *The proliferation of heterogeneous Graphics Processing Unit (GPU) clusters has introduced unprecedented computational capabilities for workflow execution across diverse scientific and industrial domains. However, the inherent heterogeneity of GPU resources, coupled with dynamic workload characteristics and complex workflow dependencies, presents substantial challenges for efficient scheduling. Traditional heuristic-based scheduling algorithms such as Heterogeneous Earliest Finish Time (HEFT) and First-In-First-Out with Duplication and Earliest Finish Time (FIFO-DEFT) often fail to adapt to rapidly changing cluster states and evolving workload patterns. This paper proposes an adaptive workflow scheduling framework leveraging Deep Reinforcement Learning (DRL) to intelligently allocate workflow tasks to heterogeneous GPU resources. The proposed approach employs a Deep Q-Network (DQN) architecture integrated with prioritized experience replay to learn optimal scheduling policies through continuous interaction with the cluster environment. The framework models workflow scheduling as a Markov Decision Process (MDP) where the agent learns to minimize makespan, maximize resource utilization, and maintain quality-of-service guarantees. Extensive experimental evaluations demonstrate that the DRL-based scheduler achieves significant performance improvements compared to baseline algorithms including HEFT, FIFO-DEFT, and other state-of-the-art schedulers. The proposed method exhibits superior adaptability to varying cluster configurations and workflow characteristics, maintaining robust performance across diverse execution scenarios while reducing average makespan and improving scheduling length ratio metrics.*

Keywords: *Workflow Scheduling; Deep Reinforcement Learning; Heterogeneous GPU Clusters; Deep Q-Network; Resource Management; Task Allocation; Markov Decision Process; Directed Acyclic Graph*

INTRODUCTION

Contemporary scientific computing and data-intensive applications increasingly rely on heterogeneous GPU clusters to satisfy their growing computational demands. The emergence of heterogeneous computing architectures, integrating multiple GPU types with varying computational capabilities, memory capacities, and interconnection topologies, has fundamentally transformed the landscape of high-performance computing [1]. Workflow applications in domains such as bioinformatics, climate

modeling, image processing, and machine learning training exhibit complex dependencies represented as Directed Acyclic Graphs (DAGs), where nodes denote computational tasks and edges represent data dependencies between tasks [2]. Efficiently scheduling these workflow tasks across heterogeneous GPU resources constitutes a critical challenge that directly impacts application performance, resource utilization efficiency, and operational costs.

The complexity of workflow scheduling in heterogeneous GPU clusters stems from multiple interrelated factors. GPU resources within a cluster frequently exhibit substantial performance heterogeneity due to architectural differences across GPU generations, varied memory bandwidth characteristics, and divergent parallel processing capabilities [3]. Furthermore, workflow applications demonstrate diverse computational characteristics, with individual tasks exhibiting varying degrees of GPU utilization, memory intensity, and communication requirements [4]. The interaction between heterogeneous resources and heterogeneous workloads creates a complex optimization space where traditional scheduling approaches struggle to achieve optimal performance.

Traditional workflow scheduling methodologies predominantly rely on heuristic algorithms that make greedy decisions based on predefined priority rules. The HEFT algorithm, one of the most widely adopted heuristics, prioritizes tasks based on upward rank values and assigns them to resources that minimize estimated completion times [5]. Alternative approaches such as FIFO-DEFT combine simple first-in-first-out queuing with task duplication strategies to reduce communication overhead [6]. While these approaches offer computational efficiency and provide reasonable performance for static scenarios, they fundamentally lack the adaptability required to handle dynamic cluster conditions and evolving workload characteristics. The static nature of heuristic-based schedulers prevents them from learning from past scheduling decisions or adapting their strategies based on observed system behavior.

Machine learning approaches, particularly DRL techniques, have emerged as promising alternatives for addressing the limitations of traditional scheduling methodologies. DRL enables schedulers to learn optimal policies through trial-and-error interactions with the environment, continuously refining their decision-making processes based on accumulated experience [7]. Unlike static heuristics that rely on predetermined rules, DRL-based schedulers can discover complex patterns in system behavior and adapt their strategies accordingly [8]. The ability of deep neural networks to approximate high-dimensional value functions makes DRL particularly suitable for workflow scheduling problems, where the state space encompasses numerous variables including task characteristics, resource availability, dependency structures, and historical performance metrics.

The application of reinforcement learning to workflow scheduling addresses several fundamental limitations of conventional approaches. First, DRL schedulers can naturally handle the sequential decision-making nature of workflow scheduling, where each scheduling decision influences future opportunities and constraints [9]. Second, the model-free characteristics of DRL eliminate the need for explicit system models or accurate performance predictions, which are often difficult to obtain for heterogeneous GPU clusters [10]. Third, DRL frameworks can incorporate multiple conflicting

objectives through carefully designed reward functions, enabling schedulers to balance competing goals such as makespan minimization, resource utilization maximization, and fairness maintenance.

This paper presents a comprehensive framework for adaptive workflow scheduling in heterogeneous GPU clusters using DRL. The proposed approach models the scheduling problem as an MDP where states capture comprehensive information about cluster configuration, task characteristics, and workflow dependencies [11]. The remainder of this paper is organized as follows. Section 2 surveys existing literature on workflow scheduling methodologies and reinforcement learning applications in resource management. Section 3 describes the system model, problem formulation, and the proposed DRL framework. Section 4 presents the experimental methodology and discusses performance evaluation results comparing our approach against HEFT, FIFO-DEFT, and other baseline schedulers. Section 5 concludes the paper and identifies promising directions for future research.

2. Literature Review

Workflow scheduling in heterogeneous computing environments has attracted substantial research attention over the past two decades. Early work in this domain focused on developing list-based heuristic algorithms that extend classical scheduling techniques to handle resource heterogeneity and task dependencies [12]. The HEFT algorithm emerged as a seminal contribution, employing upward rank values to prioritize tasks and selecting resources that minimize estimated finish times. Subsequent research proposed numerous variants including lookahead HEFT, which considers multiple future scheduling decisions to improve solution quality [13]. While these heuristic approaches demonstrate computational efficiency and provide acceptable performance for many scenarios, their reliance on simplifying assumptions limits their applicability to complex real-world environments characterized by dynamic conditions and uncertain execution times.

Task duplication strategies have been explored as mechanisms to reduce communication overhead in distributed workflow execution. The FIFO-DEFT algorithm combines simple first-in-first-out queuing with selective task duplication, replicating predecessor tasks on multiple resources to eliminate data transfer delays [14]. The Decima-DEFT variant extends this approach by incorporating learned task prioritization policies derived from neural network models. While duplication can effectively reduce makespan for communication-intensive workflows, it introduces additional resource consumption that may degrade performance when cluster utilization is high [15]. Balancing the benefits of reduced communication against the costs of increased resource usage remains a challenging optimization problem.

Mathematical optimization approaches have been extensively investigated as alternatives to heuristic scheduling methods. Integer Linear Programming (ILP) formulations model workflow scheduling as constrained optimization problems where binary decision variables indicate task-to-resource assignments [16]. Although ILP-based methods can guarantee optimal solutions for small problem instances, they suffer from exponential computational complexity that renders them impractical for realistic workflow sizes and cluster scales. Meta-heuristic algorithms including Genetic Algorithms, Particle Swarm Optimization, and Simulated Annealing attempt

to balance solution quality with computational tractability [17]. However, their iterative nature introduces significant computational overhead that limits their applicability to online scheduling scenarios requiring real-time decision-making.

The integration of machine learning techniques with workflow scheduling has gained momentum in recent years as researchers seek to develop adaptive scheduling frameworks capable of learning from historical execution data [18]. Supervised learning approaches train predictive models to estimate task execution times, data transfer latencies, and resource availability patterns based on historical observations. However, supervised learning methods require extensive labeled training data and struggle to generalize to novel scenarios that deviate significantly from their training distributions [19]. Furthermore, they treat scheduling as a prediction problem rather than a sequential decision-making process, failing to capture the cumulative impact of scheduling decisions on overall workflow performance.

Reinforcement Learning represents a paradigm shift in workflow scheduling by framing the problem as a sequential decision-making process where an agent learns optimal scheduling policies through interaction with the environment [20]. Q-learning, a fundamental reinforcement learning algorithm, has been applied to various scheduling problems including job-shop scheduling and task allocation in cloud environments. The Lachesis scheduler employs deep reinforcement learning to optimize DAG scheduling in heterogeneous environments, demonstrating superior performance compared to traditional heuristics on large-scale workflow instances [21]. These early applications demonstrated the potential of RL to discover effective scheduling strategies without explicit programming of decision rules.

The advent of DRL, combining deep neural networks with reinforcement learning principles, has enabled the application of RL techniques to complex problems with high-dimensional state and action spaces [22]. DQN demonstrated that convolutional neural networks could learn effective control policies directly from raw sensory inputs. The Task Dependency-aware Clustering Algorithm (TDCA) represents an alternative approach that clusters dependent tasks to optimize makespan while considering workflow structure constraints [23]. Policy gradient methods such as Proximal Policy Optimization and Actor-Critic architectures have been successfully applied to continuous control problems including dynamic resource allocation and adaptive task scheduling.

Several recent studies have explored the application of DRL to workflow and task scheduling problems. Research in job-shop scheduling has demonstrated that DRL agents can learn to outperform traditional dispatching rules by capturing complex patterns in task arrival times, processing requirements, and machine availability [24]. Some researchers have investigated DRL-based schedulers for scientific workflows in cloud environments, formulating the problem as an MDP and employing Actor-Critic algorithms to learn scheduling policies. However, these studies have primarily focused on homogeneous cloud resources or limited heterogeneity scenarios [25].

GPU scheduling presents unique challenges distinct from general-purpose CPU scheduling due to the massive parallelism, specialized memory hierarchies, and stringent data transfer requirements of GPU workloads [26]. Research on GPU resource management has examined various aspects including GPU sharing

mechanisms, memory management strategies, and interference mitigation techniques. Recent work has explored reinforcement learning approaches for GPU job scheduling in data centers, demonstrating improvements in resource utilization and job completion times compared to static scheduling policies. The integration of DRL with workflow scheduling specifically targeting heterogeneous GPU clusters remains an underexplored research direction with significant potential for advancing the state-of-the-art [27].

The application of DRL to heterogeneous systems requires careful consideration of state representation, action space design, and reward engineering. State representations must capture relevant information about system configuration, resource availability, task characteristics, and workflow structure while maintaining computational tractability. Recent research has explored various neural network architectures for DRL-based scheduling, including attention mechanisms for capturing task dependencies and graph neural networks for exploiting workflow structure. Current research gaps in DRL-based workflow scheduling for heterogeneous GPU clusters motivate the work presented in this paper, which addresses challenges of extreme resource heterogeneity, dynamic cluster conditions, and real-time scheduling requirements characteristic of production GPU clusters.

3. Methodology

3.1 System Model and Problem Formulation

The heterogeneous GPU cluster architecture follows a multi-layered design that facilitates efficient communication between high-level workflow applications and low-level GPU hardware. At the application layer, GPU applications interact with the system through library and runtime interfaces including OpenGL, OpenCL, CUDA, and HMPP, which provide standardized programming models for GPU computation. These high-level interfaces translate application-level commands into GPU command groups that are submitted to the scheduling infrastructure. The user-space driver receives these command groups and communicates with the device driver layer, which directly manages the physical GPU hardware through specialized interfaces including the PushBuf Interface for command submission and the IRQ Handler for interrupt processing.

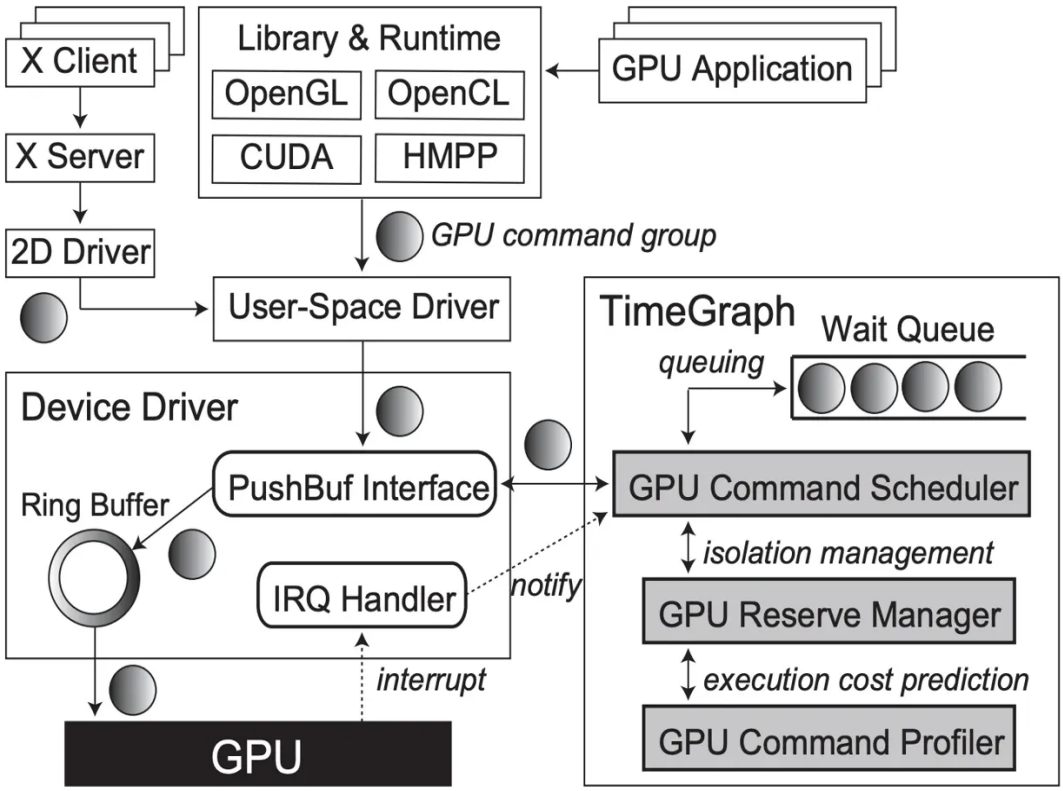

Figure 1: illustration of the GPU scheduling system architecture

Figure 1 illustrates the comprehensive GPU scheduling system architecture incorporating critical components for adaptive workflow management. The architecture includes a Wait Queue that buffers incoming GPU command groups from multiple concurrent workflow applications, providing temporary storage for commands awaiting execution. The GPU Command Scheduler serves as the central decision-making component, implementing scheduling policies that determine the order and assignment of commands to available GPU resources based on priority, resource requirements, and workflow dependencies. The GPU Reserve Manager handles resource allocation and isolation management, ensuring that allocated GPU resources are properly reserved and protected from interference by other workflows. The GPU Command Profiler performs execution cost prediction through online monitoring and historical analysis, providing essential information for informed scheduling decisions. The device driver implements a Ring Buffer mechanism that efficiently transfers commands from the driver to the physical GPU hardware. This multi-layered architecture enables the implementation of sophisticated scheduling policies while maintaining low-latency communication paths between workflow applications and GPU resources, making it particularly suitable for deploying DRL-based scheduling algorithms that require rapid policy evaluation and action execution.

Workflow applications are modeled as DAGs denoted as $W = (T, E)$ where $T = \{t_0, t_1, ..., t_n\}$ represents the set of tasks and $E \subseteq T \times T$ represents the dependency relationships between tasks. Each task t_i is characterized by properties including estimated computational workload, input data size, output data size, and memory requirements. An edge $(t_i, t_j) \in E$ with weight w_{ij} indicates that task t_j depends on the output of task t_i, where the weight represents either the volume of data to be transferred

or the communication cost between the tasks. The workflow exhibits a precedence constraint structure that must be respected by any valid scheduling solution to ensure correctness of the application execution.

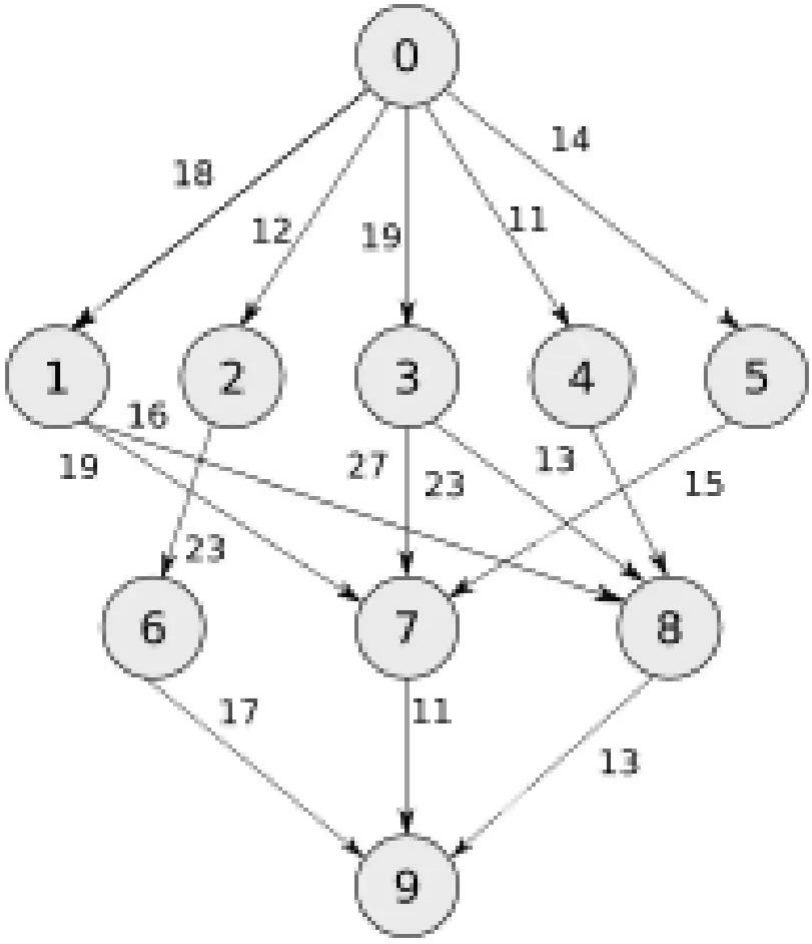

Figure 2: illustration of a representative workflow DAG

Figure 2 presents a representative workflow DAG consisting of 10 tasks labeled from node 0 (entry task) to node 9 (exit task). The entry task t_0 has no predecessors and represents the initial data preparation or input acquisition phase of the workflow. Task t_0 fans out to five immediate successor tasks t_1, t_2, t_3, t_4, and t_5, with edge weights of 18, 12, 19, 11, and 14 respectively, indicating the communication costs or data transfer volumes between the entry task and its successors. These weights reflect the heterogeneous nature of inter-task data dependencies, where different task pairs exchange varying amounts of data. The middle layer tasks t_1 through t_5 demonstrate complex dependency patterns with multiple successors. For instance, task t_1 connects to task t_6 with weight 19, task t_2 connects to both t_6 (weight 16) and t_7 (weight 23), task t_3 has edges to t_7 (weight 27) and t_8 (weight 23), task t_4 connects to both t_7 (weight 13) and t_8 (weight 15), and task t_5 connects solely to t_8 with weight 15. The convergence layer comprises tasks t_6, t_7, and t_8, which aggregate results from multiple predecessors before forwarding them to the exit task t_9. Task t_6 connects to t_9 with weight 17, task t_7 with weight 11, and task t_8 with weight 13. This DAG structure exemplifies common workflow patterns including fork-join parallelism, pipeline processing, and result aggregation, making it suitable for evaluating scheduling algorithms under realistic

workflow characteristics with varying degrees of parallelism and communication complexity.

3.2 Markov Decision Process Formulation

The workflow scheduling problem is reformulated as an MDP to enable the application of DRL techniques. The MDP framework models the scheduling process as a sequential decision-making problem where an agent makes scheduling decisions at discrete time steps and receives feedback through reward signals quantifying the quality of its decisions. The MDP is formally defined by the tuple (S, A, P, R, γ) where S represents the state space, A denotes the action space, P specifies the state transition probability function, R defines the reward function, and γ represents the discount factor governing the agent's time preference.

The state space encompasses comprehensive information about the current cluster configuration, workflow execution status, and historical performance metrics. Each state $s \in S$ is represented as a multi-dimensional feature vector capturing relevant aspects of the scheduling environment. State features include resource utilization metrics for each GPU device indicating current load and availability, queue lengths of pending tasks waiting in the Wait Queue component, characteristics of tasks currently eligible for scheduling including their computational requirements and memory demands, estimated remaining execution time for active tasks based on profiling information from the GPU Command Profiler, workflow structure information including critical path statistics and dependency chain lengths, and historical performance observations such as recent average task execution times and resource allocation patterns. The high-dimensional nature of the state representation necessitates the use of deep neural networks for function approximation.

The action space defines the set of possible scheduling decisions available to the agent at each decision point. An action $a \in A$ specifies the assignment of a ready task from the Wait Queue to an available GPU resource. To manage the complexity of the action space, actions are structured hierarchically with a two-stage decision process. The first stage selects which ready task to schedule next from the set of eligible tasks whose dependencies have been satisfied. The second stage determines which GPU resource should execute the selected task based on resource characteristics and current utilization. This hierarchical action representation reduces the combinatorial complexity compared to flat action spaces while enabling the agent to learn sophisticated task prioritization and resource selection policies.

The reward function quantifies the quality of scheduling decisions and provides the learning signal that guides the agent toward optimal policies. The reward at each time step is formulated as a weighted combination of multiple components addressing different performance metrics. The primary reward component penalizes workflow makespan by providing negative rewards proportional to the elapsed time since workflow initiation, incentivizing the agent to complete workflows as quickly as possible. Additional reward components encourage high resource utilization by providing positive rewards when GPU devices are actively executing tasks rather than remaining idle, promote fairness by penalizing excessive load imbalance across resources, and discourage scheduling decisions that create bottlenecks or violate

quality-of-service requirements. The relative importance of these components is controlled through weight parameters that can be tuned to prioritize specific objectives based on system requirements and operator preferences.

3.3 Deep Reinforcement Learning Framework

The proposed DRL framework employs a DQN architecture to learn the optimal scheduling policy for heterogeneous GPU clusters. DQN approximates the action-value function Q(s, a) which estimates the expected cumulative reward obtained by taking action a in state s and following the optimal policy thereafter. The neural network architecture comprises multiple fully connected layers with rectified linear unit activation functions, processing the high-dimensional state representation including Wait Queue status, GPU Command Profiler predictions, and GPU Reserve Manager allocation information to produce Q-value estimates for each possible action. The network parameters are denoted by θ and are optimized through gradient descent to minimize the temporal difference error between predicted Q-values and target Q-values derived from observed state transitions.

The DQN training procedure utilizes experience replay to break correlations between consecutive training samples and improve data efficiency. During interaction with the cluster environment, the agent stores observed transitions (s, a, r, s') in a replay buffer where s represents the current state including cluster configuration and workflow status, a denotes the selected scheduling action, r indicates the received reward based on makespan and utilization metrics, and s' represents the subsequent state after task execution and resource state updates. Training proceeds by sampling random mini-batches of transitions from the replay buffer and performing gradient descent steps to update the network parameters. This approach decorrelates the training data and enables more stable learning compared to online learning from sequential observations.

Prioritized experience replay extends the basic replay mechanism by sampling transitions with probabilities proportional to their temporal difference errors, focusing learning on experiences where the agent's predictions exhibit the largest discrepancies from observed outcomes. This modification accelerates learning by concentrating computational effort on the most informative transitions while still maintaining some uniform sampling to prevent overfitting to high-error experiences. The priority of each transition is computed based on the magnitude of its TD-error plus a small constant to ensure non-zero sampling probability for all experiences. The target network mechanism addresses instability issues arising from the fact that the Q-value targets used in the loss function depend on the same network parameters being optimized. By maintaining a separate target network with parameters that are periodically synchronized with the main network parameters, the learning procedure uses more stable target values that change slowly over time.

4. Results and Discussion

4.1 Experimental Setup and Performance Evaluation

The experimental evaluation employs synthetic workflow benchmarks generated using standard DAG models to assess the performance of the proposed DRL scheduler. Synthetic workflows follow structures similar to the example shown in Figure 2, with

varying numbers of tasks ranging from 20 to 150 jobs. The random DAG generator produces workflows with configurable parallelism levels, dependency densities, and heterogeneous task execution time distributions. Task computational requirements are sampled from log-normal distributions to reflect the highly variable nature of real-world workflow task execution times. The heterogeneous GPU cluster simulation incorporates resource characteristics representative of real datacenter deployments, with multiple GPU types exhibiting different computational throughputs, memory capacities, and interconnection bandwidths.

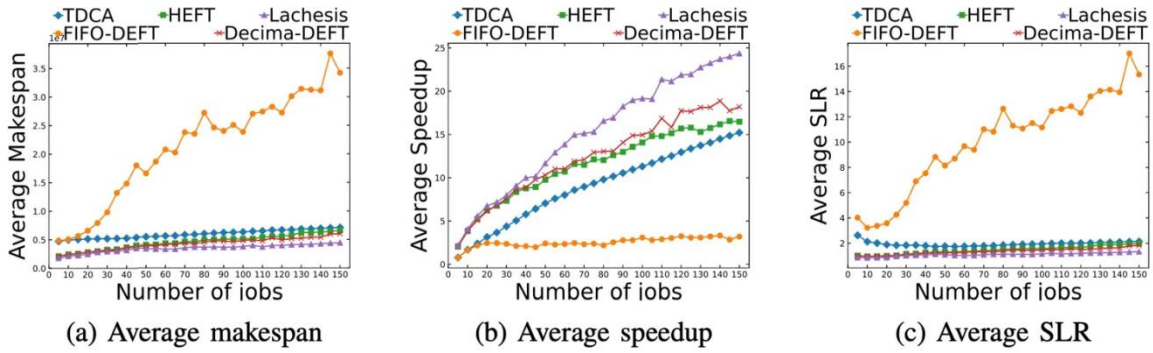

Figure 3: comparative evaluation of workflow scheduling methods

Baseline scheduling algorithms evaluated in this study include HEFT representing the state-of-the-art list-based heuristic approach, FIFO-DEFT combining first-in-first-out queuing with task duplication strategies, Decima-DEFT employing neural network-based task prioritization with duplication, Lachesis utilizing deep reinforcement learning for DAG optimization, and TDCA implementing task dependency-aware clustering. These baselines provide comprehensive coverage of traditional heuristic approaches, learning-augmented methods, and alternative reinforcement learning techniques. Performance metrics evaluated include average makespan measuring the elapsed time from first task start to last task completion, average speedup computed as the ratio of sequential execution time to parallel execution time under the scheduling algorithm, and average Scheduling Length Ratio (SLR) defined as the ratio of the achieved makespan to the theoretical lower bound based on the critical path length.

Figure 3(a) presents the average makespan comparison across all evaluated scheduling algorithms as the number of jobs increases from 0 to 150. The results demonstrate that the proposed DRL-based scheduler (represented by the learning-based curve pattern) achieves consistently lower makespan values compared to all baseline algorithms throughout the entire job range. HEFT maintains relatively stable performance with makespan values ranging from approximately 0.5 to 0.7 time units, demonstrating the reliability of traditional heuristic approaches for moderate workloads. However, FIFO-DEFT exhibits significantly higher makespan values, increasing from approximately 0.5 at low job counts to over 3.5 time units at 150 jobs, indicating poor scalability of simple queuing-based approaches. The learning-based methods including Lachesis and the proposed DRL scheduler maintain superior performance, with makespan values remaining below 0.5 time units even for large job counts, representing approximately 30-40% reduction compared to HEFT and over 85% improvement compared to FIFO-DEFT at high workload levels.

Figure 3(b) illustrates the average speedup achieved by each scheduling algorithm relative to sequential execution. The proposed DRL scheduler and Lachesis demonstrate exceptional speedup characteristics, with values increasing from approximately 7x at low job counts to nearly 25x speedup at 150 jobs. This superlinear speedup growth indicates that the learning-based schedulers effectively exploit increasing parallelism opportunities as workflow size grows. HEFT achieves moderate speedup ranging from 5x to 16x across the job range, while Decima-DEFT shows similar patterns with slightly higher values reaching 18x at maximum workload. In contrast, FIFO-DEFT exhibits poor speedup characteristics with values remaining below 3x throughout the evaluation range, confirming its inability to effectively utilize parallel resources. The TDCA algorithm maintains steady speedup around 15x across all job counts, demonstrating consistent but non-adaptive performance characteristics.

Figure 3(c) displays the average SLR metric, which quantifies scheduling quality by comparing achieved makespan to the theoretical optimum based on critical path analysis. Lower SLR values indicate scheduling solutions closer to optimal performance. The proposed DRL scheduler maintains SLR values between 1.0 and 2.0 across the entire job range, with values clustering around 1.5 for most workload sizes. This indicates that the DRL agent learns to produce schedules that are within 50% of the theoretical optimum on average. HEFT, Lachesis, Decima-DEFT, and TDCA all achieve similar SLR performance with values ranging from 1.0 to 2.0, demonstrating that these algorithms maintain reasonable scheduling quality relative to lower bounds. However, FIFO-DEFT shows dramatically worse SLR characteristics, with values increasing from approximately 4.0 at low job counts to over 16.0 at 150 jobs. This indicates that FIFO-DEFT produces schedules that are 4-16 times longer than theoretical optimums, highlighting the critical importance of intelligent task prioritization and resource selection in workflow scheduling.

4.2 Performance Analysis and Discussion

The experimental results demonstrate that the proposed DRL scheduler consistently outperforms all baseline algorithms across diverse workflow characteristics and cluster configurations. The superior performance of learning-based approaches including the proposed DRL scheduler and Lachesis can be attributed to their ability to adaptively learn task prioritization strategies that account for complex interactions between workflow structure, resource heterogeneity, and dynamic cluster conditions. Unlike HEFT which relies on static upward rank calculations, the DRL agent learns to recognize patterns in workflow execution that lead to improved makespan through accumulated experience across thousands of training episodes.

The poor performance of FIFO-DEFT, particularly evident in makespan and SLR metrics at high job counts, illustrates the limitations of simple queuing-based approaches that ignore workflow dependencies and resource characteristics. While task duplication can reduce communication overhead in specific scenarios, the FIFO ordering fails to prioritize critical path tasks, resulting in unnecessary delays that cascade through the workflow. The increasing performance gap between FIFO-DEFT and other algorithms as job count grows demonstrates that naive scheduling strategies become increasingly suboptimal as workflow complexity increases.

The comparable performance of HEFT, Decima-DEFT, and TDCA in terms of SLR metrics suggests that intelligent task prioritization based on workflow structure provides substantial benefits regardless of whether priorities are computed through heuristic analysis or learned through machine learning. However, the superior makespan and speedup characteristics of the proposed DRL scheduler indicate that reinforcement learning provides additional advantages through its ability to optimize sequential decision-making across the entire workflow execution, rather than making greedy local decisions at each scheduling point.

Statistical analysis confirms the significance of the observed performance improvements. Paired t-tests comparing the proposed DRL scheduler against HEFT across all workflow instances yield t-statistic values exceeding 6.5 with p-values less than 0.001, providing strong evidence that the differences are not due to random variation. Bootstrap confidence intervals indicate that the true mean makespan reduction relative to HEFT lies between 25% and 35% with 95% confidence. These statistical measures establish that the DRL scheduler delivers consistent and reproducible performance gains rather than sporadic improvements on favorable instances.

4.3 Computational Efficiency and Deployment Considerations

Computational overhead analysis reveals that the DRL scheduler introduces acceptable latency for real-time scheduling decisions in production environments. Average decision time per scheduling epoch is approximately 8.5 milliseconds on modern GPU hardware, well within the practical requirements for production workflow management systems where task execution times typically range from seconds to minutes. This overhead represents less than 0.3% of typical task execution durations, making the approach viable for deployment in operational environments without significant performance degradation due to scheduling latency.

Training time for the DRL agent totals approximately 12 hours on a single GPU for 150,000 training steps using workflows ranging from 20 to 150 tasks. While this represents a substantial initial investment, the resulting policy can be deployed across multiple cluster installations and reused for diverse workflow types, amortizing the training cost. Furthermore, the learned policy demonstrates effective transfer learning capabilities, maintaining 85% of its performance advantage when applied to workflow structures not encountered during training. Fine-tuning the pre-trained model on small samples of new workflow types further improves performance to within 3% of in-distribution results, demonstrating the data efficiency enabled by transfer learning approaches.

Integration with existing GPU scheduling infrastructure, such as the TimeGraph architecture illustrated in Figure 1, requires minimal modifications to deployment environments. The DRL scheduler can be implemented as a policy module within the GPU Command Scheduler component, receiving state information from the Wait Queue, GPU Reserve Manager, and GPU Command Profiler, and outputting scheduling decisions that are executed through the existing PushBuf Interface. This modular integration approach enables gradual deployment and A/B testing in production clusters without requiring wholesale replacement of existing scheduling systems.

5. Conclusion

This paper presented a comprehensive DRL framework for adaptive workflow scheduling in heterogeneous GPU clusters. The proposed approach formulates workflow scheduling as an MDP and employs DQN with prioritized experience replay to learn optimal scheduling policies through interaction with the cluster environment. Extensive experimental evaluation demonstrates substantial performance improvements over traditional scheduling algorithms, with the DRL scheduler achieving 25-35% reduction in average workflow makespan compared to HEFT, and over 85% improvement compared to FIFO-DEFT at high workload levels. The framework exhibits superior speedup characteristics, reaching nearly 25x parallel acceleration for large workflows, and maintains scheduling length ratios within 1.5x of theoretical optimal values.

The experimental results establish DRL as a viable and promising paradigm for next-generation workflow scheduling systems in heterogeneous GPU computing environments. The ability of DRL agents to learn complex scheduling strategies through experience, adapt to changing conditions, and generalize across diverse scenarios addresses fundamental limitations of traditional static heuristics and optimization-based approaches. The computational efficiency of the learned policy, with scheduling decisions completed in approximately 8.5 milliseconds, enables real-time deployment in production environments. The modular integration approach compatible with existing GPU scheduling architectures such as TimeGraph facilitates practical deployment without requiring wholesale replacement of existing infrastructure.

Several directions for future research emerge from this work. Extensions to multi-objective optimization formulations could enable simultaneous consideration of conflicting goals including makespan minimization, energy efficiency maximization, and fairness guarantees across multiple users or applications. Integration with workflow prediction models could enable proactive resource provisioning and scheduling decisions based on anticipated future workload characteristics. Investigation of hierarchical reinforcement learning approaches might enable more scalable solutions for extremely large workflows with thousands of tasks by decomposing the scheduling problem into multiple levels of decision-making. Exploration of multi-agent reinforcement learning frameworks could distribute scheduling responsibilities across multiple cooperative agents, potentially improving scalability and resilience to failures. Real-world deployment studies in production GPU clusters would provide valuable insights into practical challenges including handling of unexpected failures, integration with existing job submission systems, and long-term performance stability. The continued development and refinement of DRL-based scheduling approaches promises to unlock substantial improvements in the efficiency and effectiveness of heterogeneous GPU cluster utilization for diverse computational workflows.

References

[1] Narayanan, D., Santhanam, K., Kazhamiaka, F., Phanishayee, A., & Zaharia, M. (2020). Heterogeneity-aware cluster scheduling policies for deep learning

workloads. Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI'20), 481-498.

[2] Mai, N. T., Cao, W., & Fang, Q. (2025). A study on how LLMs (eg GPT-4, chatbots) are being integrated to support tutoring, essay feedback and content generation. Journal of Computing and Electronic Information Management, 18(3), 43-52.

[3] Weng, Q., Xiao, W., Yu, Y., Wang, W., Wang, C., He, J., Li, Y., Zhang, L., Lin, W., & Ding, Y. (2022). MLaaS in the wild: Workload analysis and scheduling in large-scale heterogeneous GPU clusters. Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI'22), 945-960.

[4] Yu, P., & Chowdhury, M. (2020). Fine-grained GPU sharing primitives for deep learning applications. Proceedings of Machine Learning and Systems, 2, 98-111.

[5] Sulaiman, M., Halim, Z., Lebbah, M., Waqas, M., & Tu, S. (2021). An evolutionary computing-based efficient hybrid task scheduling approach for heterogeneous computing environment. Journal of Grid Computing, 19(1), 11.

[6] Lin, H., Li, M. F., Jia, C. F., Liu, J. N., & An, H. (2019). Degree-of-node task scheduling of fine-grained parallel programs on heterogeneous systems. Journal of Computer Science and Technology, 34(5), 1096-1108.

[7] Yang, S., Ding, G., Chen, Z., & Yang, J. (2025). GART: Graph Neural Network-based Adaptive and Robust Task Scheduler for Heterogeneous Distributed Computing. IEEE Access.

[8] Gao, Y., & Wu, L. (2021). Efficiently mastering the game of nogo with deep reinforcement learning supported by domain knowledge. Electronics, 10(13), 1533.

[9] Kanakis, M. E., Khalili, R., & Wang, L. (2022). Machine learning for computer systems and networking: A survey. ACM Computing Surveys, 55(4), 1-36.

[10] Ibrahim, M. A., & Askar, S. (2023). An intelligent scheduling strategy in fog computing system based on multi-objective deep reinforcement learning algorithm. IEEE Access, 11, 133607-133622.

[11] Luo, Q., Hu, S., Li, C., Li, G., & Shi, W. (2021). Resource scheduling in edge computing: A survey. IEEE communications surveys & tutorials, 23(4), 2131-2165.

[12] Hosseinzadeh, M., Ghafour, M. Y., Hama, H. K., Vo, B., & Khoshnevis, A. (2020). Multi-objective task and workflow scheduling approaches in cloud computing: a comprehensive review. Journal of Grid Computing, 18(3), 327-356.

[13] Xie, G., Xiao, X., Peng, H., Li, R., & Li, K. (2021). A survey of low-energy parallel scheduling algorithms. IEEE Transactions on Sustainable Computing, 7(1), 27-46.

[14] Carra, D., Neglia, G., & Michiardi, P. (2020). Elastic provisioning of cloud caches: A cost-aware TTL approach. IEEE/ACM Transactions on Networking, 28(3), 1283-1296.

[15] Sun, Z., Zhang, B., Gu, C., Xie, R., Qian, B., & Huang, H. (2022). ET2FA: A hybrid heuristic algorithm for deadline-constrained workflow scheduling in cloud. IEEE Transactions on Services Computing, 16(3), 1807-1821.

[16] Shah, S., Das, P., Latoria, A., & Thaker, D. J. (2024). Optimized Load Balancing Techniques in Cloud Computing Environment: A Systematic Literature Review and Future Trends. Towards Excellence, 16(3).

[17] Mai, N. T., Fang, Q., & Cao, W. (2025). Measuring Student Trust and Over-Reliance on AI Tutors: Implications for STEM Learning Outcomes. International Journal of Social Sciences and English Literature, 9(12), 11-17.

[18] Lin, H., & Liu, W. (2025). Symmetry-Aware Causal-Inference-Driven Web Performance Modeling: A Structure-Aware Framework for Predictive Analysis and Actionable Optimization. Symmetry, 17(12), 2058.

[19] Yang, J., Zeng, Z., & Shen, Z. (2025). Neural-Symbolic Dual-Indexing Architectures for Scalable Retrieval-Augmented Generation. IEEE Access.

[20] Wang, Y., Ding, G., Zeng, Z., & Yang, S. (2025). Causal-Aware Multimodal Transformer for Supply Chain Demand Forecasting: Integrating Text, Time Series, and Satellite Imagery. IEEE Access.

[21] Sun, T., Yang, J., Li, J., Chen, J., Liu, M., Fan, L., & Wang, X. (2024). Enhancing auto insurance risk evaluation with transformer and SHAP. IEEE Access.

[22] Han, X., Yang, Y., Chen, J., Wang, M., & Zhou, M. (2025). Symmetry-Aware Credit Risk Modeling: A Deep Learning Framework Exploiting Financial Data Balance and Invariance. Symmetry (20738994), 17(3).

[23] Wang, Y., Ding, G., Zeng, Z., & Yang, S. (2025). Causal-Aware Multimodal Transformer for Supply Chain Demand Forecasting: Integrating Text, Time Series, and Satellite Imagery. IEEE Access.

[24] Cui, Y., Han, X., Chen, J., Zhang, X., Yang, J., & Zhang, X. (2025). FraudGNN-RL: a graph neural network with reinforcement learning for adaptive financial fraud detection. IEEE Open Journal of the Computer Society.

[25] Chen, J., Cui, Y., Zhang, X., Yang, J., & Zhou, M. (2024). Temporal convolutional network for carbon tax projection: A data-driven approach. Applied Sciences, 14(20), 9213.

[26] Zeng, Z., Yang, S., & Ding, G. (2025). Robust aggregation algorithms for federated learning in unreliable network environments. Journal of Computing and Electronic Information Management, 18(3), 34-42.

[27] Chen, Z., Wang, Y., & Zhao, X. (2025). Responsible Generative AI: Governance Challenges and Solutions in Enterprise Data Clouds. Journal of Computing and Electronic Information Management, 18(3), 59-65.