Journal of Computing and Electronic Information Management
ISSN: 2413-1660 | Vol. 23, No. 2, 2026

Energy-Aware Task Scheduling in Distributed Cloud Systems Using Graph-Based Reinforcement Learning

Felix Neumann*

Department of Computer Science, University of Freiburg, Freiburg, Germany

*** Corresponding author**

Abstract: This paper studies compact reinforcement learning for scheduling precedence-constrained directed acyclic graph (DAG) workloads across heterogeneous cloud nodes. A graph-conditioned tabular Q-learning controller selects among four transparent energy–finish-time dispatch rules, using a temporal-difference update with discounted next-state value. An always-on active/idle server-power model accounts for both execution and waiting intervals. The scheduler is evaluated on 120 held-out synthetic workflows after 800 training episodes against random, active-energy-only, greedy modeled-energy, earliest-finish-time, fixed-weight, and resource-state-only Q-learning comparators. Graph-Q yields 22717.4 J mean modeled energy and 68.48 s mean makespan; relative to fixed weighting, it reduces makespan by 5.31% while increasing modeled energy by 1.20%. Paired confidence intervals quantify these differences. The resource-state-only ablation achieves comparable modeled energy and slightly shorter completion time, delineating the value of compact graph features within this controlled simulation.

Keywords: Cloud computing; Directed acyclic graph; Workflow scheduling; Tabular Q-learning; Energy model; Reproducible simulation.

1. Introduction

Dependency-constrained workflows are an important class of workloads on distributed cloud systems. Their tasks must respect predecessor completion, while individual task duration and communication time depend on the assigned computing node. In an always-on cluster, machine selection also affects total active and idle energy. Existing scheduling research spans heterogeneous DAG list scheduling [1], workflow surveys [4], [6], [34], power modeling [9], [10], and reinforcement learning [2], [7], [11]–[16].

A conventional rule can minimize predicted finish time or prioritize instantaneous computational energy, but these two targets may disagree. A slow processor can be energy-efficient per task yet stretch the workflow horizon, increasing idle consumption on the other machines. Conversely, selecting the fastest available resource may shorten the horizon while increasing the power of busy intervals. This suggests a practical setting for learning when to emphasize energy versus completion time rather than attempting unconstrained policy search.

We investigate Graph-Q, a lightweight controller that uses task-graph features and machine-load information to select an interpretable scheduling rule. It is graph-feature-based, rather than a graph neural network (GNN): the method neither learns graph embeddings nor performs message passing. The research objective is to determine the actual energy–makespan behavior of a compact, trainable scheduler under fully specified simulation assumptions.

The main contributions are threefold: (1) a graph-conditioned Q-learning controller with a four-action rule set and explicit discounted temporal-difference updates; (2) a consistent busy/idle energy ledger and reproducible heterogeneous DAG simulator; and (3) paired benchmark comparisons, confidence intervals, and a resource-state-only Q-learning ablation. These components provide a transparent evaluation of low-overhead learning in a small distributed-cloud model.

2. Related work

Topcuoglu et al. [1] proposed HEFT, which prioritizes DAG tasks by upward rank and uses an insertion-based earliest-finish allocation. Our EFT comparator shares rank prioritization and earliest-finish selection but does not implement insertion into machine idle gaps; thus, it is explicitly a rank-ordered EFT heuristic and not full HEFT. Reviews of scheduling and cloud resource management [3]–[6], [30], [33], [34] describe the breadth of workflow constraints and optimization objectives.

Energy-aware studies have considered server power efficiency [9], cloud power-model taxonomies [10], adaptive consolidation [36], DVFS [37], thermal prediction [28], and evolutionary or swarm-based approaches [18], [38], [40]. In contrast, our hardware model is intentionally fixed: no dynamic voltage or frequency states are introduced, so any scheduling difference comes from task placement and the resulting active/idle intervals.

Learning-based approaches include Q-learning scheduling [2], [7], actor–critic resource control [13], hierarchical control [12], imitation-aided scheduling [11], and deep policies for workflow or cluster resources [8], [14]–[17], [20], [21], [43]. Recent surveys summarize the potential and tradeoffs of learning in cloud scheduling [3], [45]. Unlike those larger systems, Graph-Q limits the learned action to a dispatch-rule weight and retains fixed, auditable workload and power assumptions [46-50].

3. System model

3.1. Precedence, resources, and time

A workflow is a directed acyclic graph G=(V,E). Task j has positive work w_j; an edge (i,j) expresses precedence and a transfer delay d_ij when the tasks are assigned to different machines. Four heterogeneous nonpreemptive computing nodes have execution speeds s_h, active powers P_h, and idle powers I_h. The processing time is t_jh=w_j/s_h. Given

machine availability A_h and finish times F_i of already scheduled predecessors, a candidate start and completion time are

$$EST(j,h)=\max\{A_h, \max_{(i\in pred(j))}[F_i + d_{ij}\cdot 1(x_i\neq h)]\}; \quad EFT(j,h)=EST(j,h)+t_{jh}. \quad (1)$$

Tasks become eligible once all predecessors have been allocated; their earliest feasible start remains constrained by actual predecessor finish times. Eligible tasks are prioritized by a deterministic upward rank calculated from task work and longest successor contributions. This identical ordering is used by all comparators. A node processes at most one task at a time, without preemption or gap insertion.

3.2. Busy and idle energy

Let B_h denote the sum of processing durations actually allocated to machine h and T=max_j F_j the workflow makespan. Each machine is powered from time zero through T, including intervals before its first task and gaps while waiting for dependencies. The exact modeled energy under these assumptions is

$$E=\Sigma_h[P_h B_h + I_h(T-B_h)]=\Sigma_h(P_h-I_h)B_h + T\Sigma_h I_h. \quad (2)$$

For a tentative placement on node h, let $\Delta T_{jh}=\max(T_{current},EFT(j,h))-T_{current}$. Its marginal model-energy increment is

$$\Delta E_{jh}=(P_h-I_h)t_{jh} + (\Sigma_k I_k)\Delta T_{jh}. \quad (3)$$

Equation (3) includes the additional idle-energy exposure of all nodes when the global horizon grows. It also avoids treating a machine as continuously busy between time zero and its last scheduled completion. These are model-based joules rather than energy measurements from real servers; separate network electrical energy, startup cycles, and thermal variation are not modeled.

4. Graph-conditioned q-learning

4.1. Compact states and transparent actions

The state has four discretized entries: (i) task upward-rank ratio, 4 bins; (ii) number of incoming dependency edges, 3 bins; (iii) relative task work, 3 bins; and (iv) imbalance of the four machine-availability times, 3 bins. The resulting table has up to 4×3×3×3=108 states. The first three entries are DAG-derived, while the fourth describes resource pressure. This representation reduces the learning problem to a compact set of repeated scheduling situations.

An action chooses $\beta \in \{0,0.30,0.65,1.20\}$. The selected rule examines all four candidate machines and assigns task j to

$$h^*=\arg\min_h \{\Delta E_{jh} + 65\beta\cdot EFT(j,h)\}. \quad (4)$$

The coefficient 65 specifies a fixed score conversion between modeled joules and seconds for the tested configuration; it is a heuristic scaling parameter, not a physical conversion factor. A higher β generally favors shorter candidate completion times. Choosing a rule instead of a specific machine keeps every action feasible and reduces the policy search space.

4.2. Temporal-difference update

For each task decision, the simulator obtains the candidate marginal energy and makespan increase from (3). To keep reward magnitudes local, the immediate reward is defined relative to a counterfactual fixed-weight (β=0.48) placement evaluated at the same partial schedule:

$$r=-[(\Delta E_{selected}-\Delta E_{fixed})/1000 + 0.25(\Delta T_{selected}-\Delta T_{fixed})]. \quad (5)$$

The reference placement in (5) is not executed; it is used only to normalize the immediate learning signal. The controller updates its state-action table using the next observed dispatch state s′:

$$Q(s,a)\leftarrow Q(s,a)+\alpha[r+\gamma \max_{a'}Q(s',a')-Q(s,a)], \quad (6)$$

where α=0.13 and γ=0.90. The final task has terminal next-state value zero. An epsilon-greedy strategy linearly decreases exploration from 0.55 to a minimum of 0.05 over 800 complete training workflows. All transitions are collected from the actual chosen schedule and updated once per episode in their observed order. During testing, epsilon is set to zero and Q is fixed.

4.3. Baselines and graph-feature ablation

Random chooses a node uniformly. Active-only minimizes P_h t_jh without considering waiting energy; it is a diagnostic comparator. Energy-greedy minimizes the incremental total model energy ΔE_jh. EFT minimizes earliest finish time. Fixed weighted applies (4) with β=0.48. Resource-Q uses the same Q-learning procedure, rewards, β actions, and generated training DAGs as Graph-Q but conditions its action on machine-availability imbalance alone. This is a controlled ablation of the graph-derived state dimensions.

4.4. Computation procedure

ALGORITHM 1. Graph-Q dispatch with episodic Q updates

Input: DAG (work and transfer edges); node speeds and power; learned or empty Q table. 1) Determine upward ranks and initialize machine availability and busy-time ledgers. 2) Select the highest-rank ready task. 3) Enumerate candidate EST, EFT, ΔE, and ΔT for each node. 4) Discretize graph/resource state; choose β by epsilon-greedy Q. 5) Assign the node minimizing (4), update predecessor finish, machine availability and busy time. 6) Repeat to finish the workflow. 7) During training, apply (5)–(6) to the recorded transitions. Output: assignments, modeled energy and makespan.

For n tasks and m=4 machines, each dispatch tests m candidate placements and calculates predecessor-dependent delays. Rank generation uses the DAG edges. The small state-action table requires at most 108×4 floating-point entries, so policy lookup is constant time after the graph features have been computed. These advantages apply to the fixed-size environment studied here; the benchmark does not measure accelerator requirements or production-scale online overhead.

5. Experiment design

TABLE I. Fully specified simulation configuration. Powers are assumptions, not instrumented data-center observations.

Parameter	Configuration
Compute nodes	4 heterogeneous nodes
Node speeds	1.00, 1.35, 1.80, 2.30
Active power (W)	42, 65, 103, 165
Idle power (W)	5.04, 7.80, 12.36, 19.80
DAG sizes	20, 30, 40, 50 tasks
Edge probability	0.06, 0.10, 0.14
Work units	Lognormal (μ=2.25, σ=0.55)
Transfer times	Uniform (0.3, 2.5) s
Training / testing	800 DAGs per Q policy / 120 DAGs
Seed	2026; test generator offset +10000

DAGs are procedurally generated in topological index order, with additional predecessor edges sampled according to the selected probability. If a task has no sampled parent, an extra parent is introduced with probability 0.75. Node processing rates, communication times, and power draws remain constant across all methods. Both Q variants are trained on the same generated sequence of 800 DAGs but maintain separate Q tables. Testing uses an independent pseudorandom stream and 120 shared held-out DAGs across all seven methods.

The raw experiment file records workflow ID, size, graph-edge probability, scheduler, modeled active energy, modeled idle energy, modeled total energy, and makespan. Means and sample standard deviations are computed over held-out DAGs. Since every method uses the same DAGs, paired differences provide a direct comparison; we report normal-approximation 95% confidence intervals (mean difference ±1.96 paired standard errors). These intervals quantify variation across the synthetic test workflows, not server-meter uncertainty or variation over independent training seeds.

6. Results and discussion

TABLE II. Held-out simulated means (n=120 paired DAGs). The raw CSV includes per-workflow measurements and sample standard deviations.

Method	Mean E (J)	Mean T (s)
Random	24,416.9	132.35
Active-only	31,286.7	381.73
Energy-greedy	22,035.3	90.98
EFT	22,801.3	66.83
Fixed weighted	22,447.1	72.32
Resource-Q	22,719.2	67.01
Graph-Q	22,717.4	68.48

Graph-Q yields 22717.4 J and 68.48 s on the 120 held-out workflows. Relative to the fixed-weight policy, its average makespan is 5.31% shorter, at a 1.20% modeled-energy increase. The same Graph-Q schedule consumes 0.37% less modeled energy than EFT, while taking 2.46% longer. These values place Graph-Q between faster, higher-energy allocations and more energy-conservative strategies in the simulated tradeoff space.

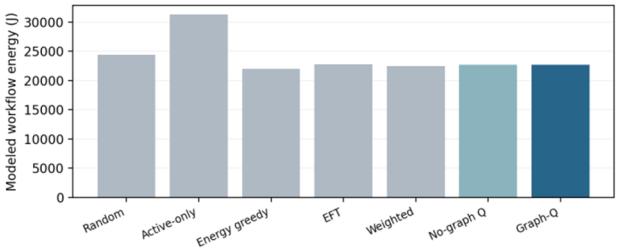


FIG. 1 Mean modeled total energy on the shared test DAGs.

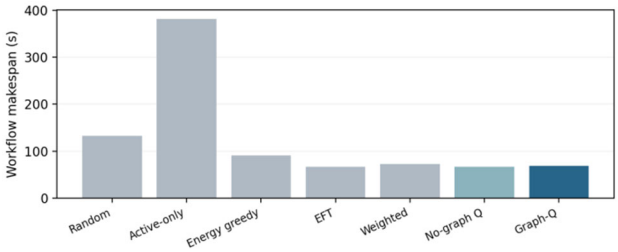


FIG. 2 Mean workflow makespan on the shared test DAGs.

The active-only rule has the largest mean simulated energy because using a slow, low-power machine can extend the time for which all four machines remain powered. The energy-greedy method accounts for this idle contribution and reduces the average total energy relative to the active-only rule. EFT produces the shortest mean makespan, whereas fixed weighting is a compromise. This ordering illustrates why active power alone should not be used as total workflow energy.

6.1. Paired confidence intervals

For Graph-Q minus fixed weighting, the mean paired energy difference is +270.3 J with a 95% interval [218.5, 322.0] J; the corresponding makespan difference is -3.84 s [-4.61, -3.08] s. The intervals exclude zero in this test set, indicating that the observed tradeoff is consistent across the paired synthetic workflows.

6.2. Graph-feature ablation

Resource-Q, which observes only resource imbalance, obtains 22719.2 J and 67.01 s. Graph-Q differs by -1.8 J and +1.47 s on average; the paired differences have 95% intervals of [-29.5, 25.8] J and [1.08, 1.86] s, respectively. Thus, the extra DAG state dimensions do not improve the two test metrics simultaneously in this compact simulator. This result is useful when judging whether additional graph conditioning justifies its state space.

The resource-only Q policy uses three availability-imbalance states, while Graph-Q expands this to 108 graph-conditioned states. Because all methods already share the same upward-rank task priority, extra DAG information can overlap with structural information that is implicitly present in the scheduling sequence. In a small stationary workload generator, a smaller table can learn reliable rule preferences more quickly. The ablation therefore distinguishes the presence of graph-derived state information from its measured utility for this configuration; it does not conflate Graph-Q with a trained GNN or imply a universal advantage from adding features.

6.3. Stratification by workflow size

TABLE III. Test-set breakdown by workflow size (T in seconds; E in modeled joules). Counts are 26, 41, 29, and 24 for 20-, 30-, 40-, and 50-task DAGs, respectively.

Tasks	Weighted T	Graph-Q T	Graph-Q E
20	45.18	42.57	13,464
30	65.41	61.60	20,263
40	84.19	79.41	26,478
50	99.21	95.10	32,391

The workflow-size breakdown uses exactly the same held-out DAGs as the main comparison. It separates the influence of increasing task count from the overall mean. The state discretization and Q table are shared across all DAG sizes; no size-specific model is fitted. Additional per-workflow details are preserved in the accompanying raw data.

6.4. Scope of inference

All energy values are produced from the constant-power formula in (2), not hardware probes. The workload distribution is synthetic and no production trace, network electrical load, server boot cycle, multi-workflow queue, or DVFS policy is simulated. Comparisons therefore support conclusions about this specified small heterogeneous-cluster

model. The fixed-weight and resource-only baselines are informative checks on learned-policy complexity; the rank-ordered EFT result should not be interpreted as a comparison against full insertion-based HEFT.

6.5. Sensitivity to idle-power assumptions

We additionally vary the assumed idle-to-active power ratio from 0.06 to 0.24, while preserving active powers, test DAGs, and the Q tables trained at the nominal 0.12 setting. Consequently this is a transfer evaluation of the nominal policy rather than a retraining experiment. Candidate placement costs are recalculated at each fraction using (3).

TABLE IV. Held-out power-model sensitivity, mean E (J) / T (s). Policies are trained only at nominal idle fraction 0.12.

Idle/active	Weighted E / T	Graph-Q E / T
0.06	22,092 / 74.27	22,119 / 72.99
0.12	22,447 / 72.32	22,717 / 68.48
0.24	23,101 / 68.11	23,073 / 67.12

Graph-Q has shorter mean makespan than fixed weighting at each tested idle fraction. At 0.06, its mean energy is 27.1 J higher; at 0.24, its mean energy is 27.8 J lower. For the 0.24 case, the paired 95% confidence interval for the energy difference is [−62.6, 7.1] J, while the interval for makespan is [−1.37, −0.61] s. Thus, the apparent high-idle energy advantage is small relative to workflow variation, whereas the completion-time difference remains consistently negative in the synthetic test set. The underlying per-workflow sensitivity records are included in the accompanying results [51-56].

7. Conclusion

A compact graph-conditioned Q-learning scheduler for heterogeneous cloud DAGs was implemented with an exact busy/idle simulation ledger and standard discounted temporal-difference training. On 120 paired held-out synthetic workflows, Graph-Q recorded 22717.4 J mean modeled energy and 68.48 s mean makespan. Compared with fixed weighting, it shortened completion time by 5.31% for 1.20% additional modeled energy. A resource-state-only ablation achieved a slightly shorter makespan, showing that the benefit of richer graph state is not automatic. The released experiment scripts, per-workflow outcomes, and assumptions allow the energy–latency comparisons to be reproduced and extended to more realistic server traces [57-60].

References

[1] Topcuoglu, H., Hariri, S., & Wu, M.-Y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Transactions on Parallel and Distributed Systems, 13(3), 260-274. https://doi.org/10.1109/71.993206

[2] Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. Machine Learning, 8, 279-292. https://doi.org/10.1007/BF00992698

[3] Zhang, F., & Yang, J. (2024). Robustness-aware model compression for multimodal perception under challenging conditions. AI and Data Science Journal, 1(1), 103-111.

[4] 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.

[5] 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.

[6] 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, 12, 116546-116557.

[7] Wang, G., Li, P., Feng, E., Shen, A., & Rhee, M. (2026). A dual-process memory-augmented large language model agent for explainable sequential recommendation. IEEE Access.

[8] Guo, Z., Chen, T., & Shang, W. (2022). Class-conditional intermediate-domain adversarial adaptation for cross-device acoustic scene classification. Journal of Computer Science and Electrical Engineering, 4(1), 26-35.

[9] Jiao, Y., Fan, H., Yue, X., Ping, W., Sun, T., & Wang, J. (2026). Dynamic heterogeneous graph contrastive learning for uncovering collusive financial fraud. Scientific Reports.

[10] Liang, Y., Jiao, Y., Ping, W., Fan, H., & Han, X. (2026). Adaptive event-driven labeling: A neuro-symbolic multiagent framework for causal inference in non-stationary time series. IEEE Access.

[11] 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.

[12] Yue, X., Yang, J., & Liu, W. (2026). FraudDebate-Agent: A multi-agent LLM framework with an evidence-based debate mechanism for financial statement fraud detection. Mathematics, 14(15), 2695.

[13] Mo, T., Wang, G., Feng, E., & Li, P. (2023). Execution-feedback-guided context selection for large language model-based software testing and debugging. World Journal of Information and Knowledge Management, 1(1), 55-64.

[14] Chen, J., Zhang, X., & Han, X. (2023). Explainable identification of tax-related financial reporting anomalies in U.S. public companies: Cross-fitted tax-consistency residuals and Owen-value Shapley propagation. World Journal of Economics and Business Research, 1(1), 33-45.

[15] Feng, E., Li, P., Mo, T., & Wang, G. (2022). PRISM: Prediction-guided risk-aware deep reinforcement learning for energy-aware dynamic resource scheduling in distributed cloud data centers. Journal of Computer Science and Electrical Engineering, 4(1), 53-61.

[16] Teng, D. (2025). RSRBench: Reproducibility and performance benchmarking for regional seismic risk workflows. Academic Journal of Architecture and Civil Engineering, 3(1), 40-52.

[17] Chen, J., Liang, Y., Liu, J., & Zhou, M. (2026). Temporal transformer with conditional tabular GAN for credit card fraud detection: A sequential deep learning approach. Mathematics, 14(7), 1183.

[18] 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.

[19] Lin, Y., Yue, X., Bian, M., Zhu, R., & Teng, D. (2026). Subgraph-level temporal graph transformer with laundering-motif prototypes for anti-money laundering on blockchains. IEEE Access.

[20] Teng, D., Rhee, M., Qin, Y., Zi, B., & Liu, W. (2026). SW-SpeedDLM: Sliding window speculative decoding for diffusion language models under long context constraints. Mathematics, 14(12), 2137.

[21] Wang, Y., Bian, M., & Lin, Y. (2022). An explainable temporal graph neural network for disruption prediction and risk propagation in multi-tier supply chains. Journal of Computer Science and Electrical Engineering, 4(1), 17-25.

[22] Fan, H., Liang, Y., Jiao, Y., Ping, W., & Lin, Y. (2026). An explainable temporal machine learning framework for financial fraud risk prediction in listed institutions. Highlights in Business, Economics and Management, 70, 173-185.

[23] Wang, J., Liu, J., Ge, Y., & Jiang, B. (2023). Risk-aware ensemble learning for credit default prediction under economic uncertainty. World Journal of Economics and Business Research, 1(1), 18-32.

[24] Ping, W., & Fan, H. (2024). Disclosure quality and valuation risk in level 3 investments: A disclosure-implied risk framework for U.S. business development companies. Social Science and Management, 1(1), 78-87.

[25] Lin, Y., Wang, Y., & Bian, M. (2024). XOR-EW: An explainable and cost-aware artificial intelligence framework for operational risk early warning in financial institutions. AI and Data Science Journal, 1(1), 94-102.

[26] Zhao, X., Liu, J., Wang, Y., & Wang, J. (2026). CryptoMamba-SSM: Linear complexity state space models for cryptocurrency volatility prediction. IEEE Open Journal of the Computer Society, 7, 226-243.

[27] Fang, Q., & Liu, W. (2025). HARLA-ED: Resolving information asymmetry and enhancing algorithmic symmetry in intelligent educational assessment via hybrid reinforcement learning. Symmetry, 18(1), 58.

[28] Wang, J., Liu, J., Zheng, W., & Ge, Y. (2025). Temporal heterogeneous graph contrastive learning for fraud detection in credit card transactions. IEEE Access.

[29] 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.

[30] Wang, B., Zhao, W., & Wang, Z. (2024). DAS-GNN: A scalable disagreement-aware graph neural framework for anomaly detection in large-scale networks. AI and Data Science Journal, 1(1), 67-75.

[31] 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.

[32] Guo, Z., Fang, Q., Yang, J. S., & Rhee, M. (2026). Closed-loop audio neuroregulation for ADHD: Graph-based attention decoding and rhythm entrainment. IEEE Access.

[33] Chen, J., Han, X., Zhang, X., & Yang, Y. (2023). Exact concept-level explanations for early detection of financial misstatements. Eurasia Journal of Science and Technology, 5(2), 34-48.

[34] Zhang, C., & Wang, Y. (2024). A data-driven multi-objective production scheduling framework for battery pack manufacturing under material arrival uncertainty. AI and Data Science Journal, 1(1), 86-93.

[35] Ping, W., & Fan, H. (2023). Beyond isolated ratios: Accrual-revenue-cash flow indicators and interaction-aware learning for detecting financial reporting irregularities in U.S. public companies. World Journal of Economics and Business Research, 1(1), 46-55.

[36] Zhang, F., & Yang, J. (2022). Accuracy-aware model compression for real-time visual perception under resource constraints. Journal of Computer Science and Electrical Engineering, 4(1), 45-52.

[37] Zhu, R., & Yue, X. (2024). An explainable machine learning framework for predicting dividend sustainability and financial risk of US REITs under interest-rate shocks. Journal of Trends in Finance and Economics, 1(2), 48-57.

[38] Fang, Q. (2024). Bias-aware large language models for equitable educational feedback and assessment. Trends in Social Sciences and Humanities Research, 2(1), 41-50.

[39] 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.

[40] Chen, J., Liu, J., Liang, Y., & Zhou, M. (2026). HeteroGCL: A heterogeneous graph contrastive learning framework for scalable and sustainable cryptocurrency AML. Applied Sciences, 16(6), 2860.

[41] Fan, H., Jiao, Y., Wang, M., Chen, J., & Yang, S. (2026). Fraud learns too: Continual graph learning under strategic adversarial drift in dynamic networks. Scientific Reports.

[42] Teng, D. (2025). TRIAD: Deadline- and reliability-aware scheduling for post-earthquake structural assessment workflows. World Journal of Engineering Research, 3(6), 55-66.

[43] Li, P., Feng, E., Wang, G., & Mo, T. (2024). Hybrid deep reinforcement learning and integer optimization for resource scheduling in heterogeneous AI computing clusters. Innovation and Technology Studies, 1(1), 48-56.

[44] Mai, N. T., Cao, W., & Fang, Q. (2025). A study on how LLMs (e.g., 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.

[45] Wang, G., Li, P., Feng, E., & Mo, T. (2024). Uncertainty-aware LLM-as-a-judge evaluation for reliable search and recommendation systems: Calibrated confidence and impact-aware selective human review. AI and Data Science Journal, 1(1), 112-121.

[46] Wang, J., Ge, Y., Liu, J., & Jiang, B. (2024). Early warning of liquidity stress in U.S. banks with seasonally aligned frailty hazards and delayed-feedback conformal alarms. Journal of Trends in Finance and Economics, 1(1), 85-100.

[47] Qin, Y., Rhee, M., Teng, D., Zhang, C., & Zi, B. (2026). Neuro-symbolic constraint verification for LLM-driven internet finance transaction execution. Scientific Reports.

[48] Cao, J., Zheng, W., Ge, Y., & Wang, J. (2025). DriftShield: Autonomous fraud detection via actor-critic reinforcement learning with dynamic feature reweighting. IEEE Open Journal of the Computer Society, 6, 1166-1177.

[49] 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.

[50] 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.

[51] Shen, A. (2023). Compute-efficient scaling of large sequential recommendation models under dynamic user behavior. World Journal of Information Technology, 1(1), 62-71.

[52] 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.

[53] Zou, J., Qin, Y., & Rhee, M. (2024). Beyond model scale: A task-aware reliability benchmark for large language models in clinical decision support. Innovation and Technology Studies, 1(1), 39-47.

[54] Shen, A., & Liu, W. (2026). Budget-aware adaptive test-time compute with hierarchical self-verification for semantic ID generative recommendation. Computers.

[55] Zhang, F., Shen, A., & Huang, S. (2026). TRIAD: Co-designing mixed-precision quantization, operator fusion, and on-chip memory reuse for real-time vision-language-action inference on automotive NPUs. Results in Engineering, 113673.

[56] Zhou, F. (2026). Moment-constrained side-force allocation for wind-assisted merchant ships: Hydrodynamic performance of retractable anti-leeway fins and flap rudders. Innovation and Technology Studies, 3(1), 27-41.

[57] Chen, T., Shang, W., & Guo, Z. (2024). XSLA: An SLA-aware heterogeneous XPU scheduling framework for edge AI inference. World Journal of Information Technology, 2(2), 48-59.

[58] 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.

[59] Mo, T., Zhang, C., Zou, J., Guo, Z., & Rhee, M. (2026). Self-evolving AI agents with dual memory for automated software testing and bug localization. IEEE Access.

[60] Zhang, H. (2024). Graph-aware multi-task learning for early prediction of timing violations and routing congestion in digital IC physical design. World Journal of Information Technology, 2(1), 13-20.

[61] Wang, J., Tan, Y., Jiang, B., Wu, B., & Liu, W. (2025). Dynamic marketing uplift modeling: A symmetry-preserving framework integrating causal forests with deep reinforcement learning for personalized intervention strategies. Symmetry, 17(4), 610.

[62] 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.