Hierarchical Risk-Aware RL for DVFS-Enabled IoE Scheduling

Amir Javadpour[1,*] Forough Ja'fari[2] Tarik Taleb[3]

[1]Senior Cybersecurity Researcher MOSA!C Lab / ICTFICIAL Oy, Espoo, Finland

[2]Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

[3]Faculty of Electrical Engineering and Information Technology, Ruhr University Bochum, Bochum, Germany

[*]Corresponding Author Email: a.javadpour87@gmail.com

Abstract

Internet of Everything (IoE) services generate heterogeneous, bursty, delay-sensitive, and energy-intensive tasks that must be scheduled on virtualized cloud platforms under coupled delay, energy, SLA, migration, and utilization constraints. This paper presents ER-RL-DVFS, a hierarchical energy-risk-aware scheduling framework for DVFS-enabled IoE cloud systems. The framework combines an energy-risk task-priority score, DVFS-aware cluster–PM–VM profiling, hierarchical candidate filtering, deterministic score-based assignment for clearly separated candidates, selective tabular reinforcement learning as the prescribed refinement mechanism for near-tie decisions, and migration-aware correction. The methodological contribution is the coordinated scheduling pipeline rather than a new reinforcement-learning architecture. The numerical study uses a controlled stochastic benchmark with frozen seed-level outputs for five independent seeds $(7, 19, 41, 73, 101)$, two resource configurations (512×16 and 1024×32), generic IoE and Smart-City IoT workloads, and eight baselines, including both GIoTDVFS-SFB and GIoTDVFS-mGA. Reported curves and heavy-load summaries use seed means and Student-t 95% confidence intervals. For 2000 generic-IoE tasks under 512×16, ER-RL-DVFS obtains a mean makespan of 1068.47 ± 16.77 and energy consumption of $(3.722 \pm 0.035) \times 10^5$ J, compared with 1329.22 ± 12.37 and $(4.305 \pm 0.077) \times 10^5$ J for DQN-VMPlacement. The corresponding mean energy-delay product decreases from 0.5722×10^9 to 0.3977×10^9, a 30.50% reduction. The benchmark also shows a 36.32% EDP reduction in the Smart-City stress case. The evidence supports the comparative energy-delay-SLA behavior encoded by the proposed integrated scheduling design while making explicit that the supplied benchmark harness is profile-based and does not execute neural-network training or a line-by-line online Q-table implementation. This distinction, together with missing low-level filtering/power constants, unmatched learning-compute budgets, and unmeasured scheduler wall-clock overhead, defines the current reproducibility boundary and motivates direct implementation-based validation.

Keywords: Green cloud computing; IoE task scheduling; DVFS; reinforcement learning; VM migration; energy-aware scheduling; SLA-aware resource management.

1 Introduction

Cloud data centers have become the dominant execution substrate for IoE applications such as camera-based monitoring, smart healthcare, industrial sensing, large-scale analytics, and event-driven intelligent services. These applications generate task streams that are different from traditional batch workloads because their arrival process is often bursty, their resource demand is heterogeneous, and their quality-of-service requirement depends on urgency and context. A cloud task scheduler must therefore decide not only where to place a task, but also how aggressively to operate the host frequency, whether VM migration is justified, and whether short-term energy saving may increase long-term SLA violation. This makes IoE-cloud scheduling a coupled optimization problem rather than a simple queue-ordering problem [1–3].

A common strategy is to assign tasks to VMs according to heuristics such as round robin, Min-Min, Max-Min, or load-aware rules. These rules are simple, but they are not designed to explicitly control processor frequency or migration overhead. DVFS-based methods reduce energy by lowering CPU frequency when slack is available, while VM consolidation and migration reduce the number of active or overloaded PMs. Learning-based methods, especially reinforcement learning and deep reinforcement learning, have recently been explored for task scheduling and VM placement because they can adapt to dynamic cloud states. Recent studies report DQN-based VM placement for carbon-aware and SLA-aware data centers, RL-driven multi-objective task scheduling, and DVFS-driven workload provisioning in heterogeneous cloud platforms [3–7]. However, these mechanisms are still frequently studied separately. In practice, a scheduler that selects a low-energy VM without considering deadline pressure may increase SLA loss, while a scheduler that aggressively migrates tasks may reduce overload but increase network and energy overhead.

This paper proposes ER-RL-DVFS, a hierarchical energy-risk-aware RL framework for DVFS-enabled IoE task scheduling in virtualized cloud data centers. The proposed design is motivated by three observations. First, IoE tasks have different urgency and energy-risk patterns; therefore, task prioritization must be richer than arrival-order or task-length sorting. Second, a cloud platform has a hierarchical structure consisting of clusters, PMs, and VMs; therefore, searching all VMs directly is inefficient and weakly interpretable. Third, deterministic scores work well when candidate quality is clearly separated, but they are less effective when several VMs have similar scores under changing queue, energy, and migration states. In such cases, RL can learn from previous decisions and improve the adaptive selection policy.

The mathematical core of the proposed method is a multi-objective scheduling objective that combines makespan, energy consumption, SLA violation, migration cost, and utilization gain. For a task set $\mathcal{T}$ and VM set $\mathcal{V}$, the scheduler determines binary placement

variables $x_{ij} \in \{0,1\}$, where $x_{ij} = 1$ means that task t_j is assigned to VM v_i. The high-level optimization problem can be written as

$$\min_{x,f,m} \ \Omega = w_1 E^{tot} + w_2 T^{mk} + w_3 \Psi^{sla} + w_4 C^{mig} - w_5 G^{util}, \tag{1}$$

where E^{tot} is total energy, T^{mk} is makespan, Ψ^{sla} is SLA violation, C^{mig} is migration cost, G^{util} is utilization gain, and f and m denote DVFS and migration decisions. This formulation shows why a single-rule scheduler is insufficient: reducing one term may increase another term. ER-RL-DVFS addresses this coupling through task scoring, DVFS-aware resource profiling, hierarchical filtering, score-based fast allocation, and RL-based refinement.

The contributions of this paper are summarized as follows. First, an energy-risk-aware task-priority model ranks IoE tasks by combining deadline pressure, computational demand, expected energy impact, task risk, and bandwidth demand. Second, DVFS-aware PM/VM profiling couples execution-time and power estimates with queue and utilization states. Third, a hierarchical cluster–PM–VM candidate-selection mechanism reduces the number of resources evaluated at each scheduling event. Fourth, a score-based fast policy handles clearly separated candidates, while a lightweight tabular RL mechanism is formally prescribed for near-tie decisions; accordingly, the novelty claim concerns the selective hybrid scheduling architecture rather than a new RL algorithm. Fifth, a formally specified migration-aware correction procedure supports local reassignment, internal migration, and external migration when the expected feasibility/SLA benefit justifies migration overhead. Sixth, the numerical evaluation is consolidated on one canonical five-seed dataset containing eight baselines, including GIoTDVFS-SFB and GIoTDVFS-mGA, with explicit 95% confidence intervals and bundled seed-level data and benchmark-generation code. The manuscript explicitly distinguishes the algorithmic specification from the supplied profile-based stochastic benchmark so that empirical claims do not exceed what the executable materials support.

The remainder of this paper is organized as follows. Section 2 presents the background and motivation. Section 3 reviews related work and positions the proposed framework. Section 4 defines the system model and optimization formulation. Section 5 presents the ER-RL-DVFS scheduling framework and its complexity. Section 6 provides an illustrative numerical example. Section 7 describes the experimental setup. Section 8 reports the main simulation results. Section 9 interprets representative VM-selection paths. Section 10 provides replicate-level validation and uncertainty reporting. Section 11 concludes the paper and outlines future work.

2 Background and Motivation

2.1 IoE Workload Characteristics in Cloud Platforms

IoE workloads consist of continuous task streams generated by connected sensors, wearable devices, cameras, industrial controllers, mobile devices, smart gateways, and analytics services. Unlike conventional homogeneous batch workloads, IoE tasks are highly heterogeneous in terms of execution length, arrival pattern, deadline tightness, memory demand, bandwidth requirement, and service risk. For example, a periodic environmental sensing task may tolerate moderate delay, whereas a fall-detection alert, industrial fault alarm, or traffic-accident notification must be processed with much stricter timing requirements. Therefore, IoE-cloud scheduling cannot be modeled only as a generic task-to-VM assignment problem; it must also consider urgency, risk, queue pressure, energy impact, and resource feasibility.

Let the arrival process of IoE tasks be denoted by $\{A_j\}_{j=1}^{n}$. The inter-arrival time between two consecutive tasks is

$$\Delta A_j = A_j - A_{j-1}, \quad j = 2, \ldots, n, \tag{2}$$

where small values of ΔA_j indicate bursty workload behavior. Bursty arrivals are common in IoE systems because external events may trigger many tasks in a short time interval. For instance, abnormal traffic, emergency monitoring, or camera-based detection may suddenly increase the number of submitted tasks. In such cases, a scheduler that performs well under average load may still fail when tasks arrive in bursts.

The computational pressure of task t_j can be measured by

$$\chi_j = \frac{L_j}{D_j - A_j + \epsilon}, \tag{3}$$

where L_j is the task length in million instructions, D_j is the absolute deadline, and ϵ is a small positive constant used to avoid division by zero. A larger χ_j indicates that the task has a larger computational demand relative to its available execution window. This means that two tasks with the same length may have different scheduling priorities if their deadlines are different.

An IoE task is represented by the resource-risk vector

$$\mathbf{r}_j = [C_j, R_j^{ram}, R_j^{sto}, B_j, \rho_j]^T, \tag{4}$$

where C_j is CPU demand, R_j^{ram} is memory demand, R_j^{sto} is storage demand, B_j is bandwidth demand, and $\rho_j \in [0,1]$ is the task risk coefficient. The risk coefficient captures the consequence of delayed, dropped, or failed execution. A task associated with elderly fall detection, health monitoring, emergency notification, or industrial fault detection has a higher risk value than a periodic low-priority telemetry task. Hence, minimizing average completion time is not sufficient. A practical IoE-cloud scheduler must also protect high-risk and deadline-sensitive tasks.

This workload structure motivates the use of an energy-risk-aware priority model. Instead of sorting tasks only by arrival time, length, or earliest completion time, the scheduler should rank them according to multiple factors, including urgency, computational demand, expected energy impact, bandwidth pressure, and risk level. Such a design is especially important when the cloud platform is under heavy load, because early assignment of low-risk tasks to favorable VMs may leave high-risk tasks with weak or congested resources.

2.2 Energy Consumption in Virtualized Cloud Data Centers

A virtualized cloud data center contains PMs that host multiple VMs. Each PM consumes power even when it is lightly loaded, and the dynamic part of power consumption changes according to utilization and CPU operating state. Let $u_k(t)$ denote the CPU utilization of PM p_k at time t. A commonly used host power model is

$$P_k(t) = P_k^{idle} + \left(P_k^{max} - P_k^{idle}\right) u_k(t)^{\beta}, \tag{5}$$

where P_k^{idle} and P_k^{max} are the idle and maximum power levels of PM p_k, and $\beta \geq 1$ controls the nonlinearity of utilization-dependent power consumption. The energy consumed by PM p_k during interval $[0, T]$ is

$$E_k = \int_0^T P_k(t)\, dt. \tag{6}$$

The total data-center energy is then

$$E^{tot} = \sum_{p_k \in \mathcal{P}} E_k + E^{mig} + E^{net}, \tag{7}$$

where E^{mig} and E^{net} represent migration-related and network-related energy overheads, respectively.

This model shows that energy-aware scheduling cannot be reduced to selecting the VM with the lowest immediate energy value. A VM may appear energy-efficient at the current time, but assigning a task to it may increase queueing delay, cause future migration, or overload its host PM. Similarly, consolidating tasks on a small number of PMs may improve utilization, but it may also increase SLA violation if the selected PMs become bottlenecks. Therefore, energy-aware IoE-cloud scheduling must jointly consider host power, VM queue state, DVFS level, migration overhead, and deadline risk.

2.3 DVFS-Based Energy Control

DVFS changes CPU voltage and frequency to reduce dynamic power consumption. The dynamic CPU power can be approximated by

$$P_k^{dyn} = \alpha_k C_k^{eff} V_k^2 f_k, \tag{8}$$

where α_k is an activity factor, C_k^{eff} is effective capacitance, V_k is supply voltage, and f_k is CPU frequency. If voltage is modeled as an increasing function of frequency, it can be written as

$$V_k(f_k) = V_k^{min} + \frac{f_k - f_k^{min}}{f_k^{max} - f_k^{min}} \left(V_k^{max} - V_k^{min}\right). \tag{9}$$

This relation shows why lowering the operating frequency can reduce power consumption. However, lower frequency also increases execution time. Therefore, DVFS is beneficial only when the task has sufficient slack or when the energy saving is larger than the delay penalty.

For a task t_j running on VM v_i hosted by PM p_k, the estimated execution time under DVFS is

$$\hat{T}_{ij}(f_k) = \frac{L_j}{MIPS_i \eta_i (f_k / f_k^{max})} + Q_i, \tag{10}$$

where $MIPS_i$ is the processing capacity of VM v_i, η_i is the virtualization efficiency factor, and Q_i is the expected queue delay. This equation highlights the main DVFS trade-off: reducing f_k may save energy but may increase $\hat{T}_{ij}$ and cause deadline violation. Therefore, DVFS must be coordinated with task urgency, SLA constraints, and VM queue pressure.

This motivates the DVFS-aware profiling layer of ER-RL-DVFS. Instead of applying frequency reduction blindly, the proposed framework evaluates the current frequency state, host utilization, queue pressure, and estimated task completion time before selecting a VM. As a result, frequency-aware energy saving is used only when it is compatible with the delay and SLA requirements of the task.

2.4 VM Placement, Queue Pressure, and Migration Trade-Off

In a virtualized cloud environment, the scheduling decision affects both VM-level and PM-level behavior. Assigning a task to a VM increases its queue workload, changes the utilization of its host PM, and may affect the availability of resources for future tasks. If the selected VM becomes overloaded or infeasible, migration may be used to move workload to another VM, PM, or cluster. Migration can reduce overload and improve deadline satisfaction, but it is not free. It introduces memory-transfer delay, network traffic, and additional energy consumption.

Let D_{ij}^{mig} and E_{ij}^{mig} denote the migration delay and migration energy associated with assigning or moving task-related workload from one candidate state to another. A migration-aware scheduler should compare the expected benefit of migration with its cost. If migration only slightly improves utilization but significantly increases delay and energy, it should be avoided. Conversely, if migration prevents high-risk deadline violation or releases a severely overloaded PM, it may be beneficial.

This trade-off motivates the migration-aware correction mechanism in ER-RL-DVFS. The proposed framework does not use migration as a default action. Instead, it first attempts to select a suitable VM through task prioritization, DVFS-aware profiling, and hierarchical filtering. Migration is considered only when the selected assignment becomes infeasible, overloaded, or likely to violate SLA. This makes migration a corrective mechanism rather than an uncontrolled source of overhead.

2.5 Hierarchical Resource Structure and Search-Space Reduction

Cloud platforms naturally have a hierarchical structure. A data center contains clusters or zones, each cluster contains multiple PMs, and each PM hosts several VMs. A flat scheduling policy that evaluates all VMs for every incoming task may become expensive when the number of VMs is large. If n tasks are scheduled over $|\mathcal{V}|$ VMs, direct evaluation has complexity proportional to $O(n|\mathcal{V}|)$. This cost becomes significant in online IoE-cloud scheduling because decisions must be made repeatedly under changing workload conditions.

The hierarchical structure can be exploited to reduce the search space. A scheduler can first select promising clusters, then select suitable PMs within those clusters, and finally evaluate only a small set of candidate VMs. This approach has two advantages. First, it reduces online scheduling cost. Second, it improves interpretability because the final assignment can be explained through a structured path: cluster selection, PM selection, and VM selection.

ER-RL-DVFS follows this hierarchical decision principle. It avoids exhaustive VM search by using cluster-level, PM-level, and VM-level filters. This is important for scalability because the scheduler must remain responsive even when the cloud platform contains hundreds or thousands of VMs.

2.6 Reinforcement Learning for Dynamic Scheduling

Reinforcement learning models scheduling as a sequential decision-making process. At scheduling step τ, the agent observes state s_τ, selects action a_τ, receives reward r_τ, and moves to the next state $s_{\tau+1}$. The goal is to maximize the expected discounted return

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{\tau=0}^{\infty} \gamma^\tau r_\tau \right], \tag{11}$$

where π is the scheduling policy and $\gamma \in [0, 1)$ is the discount factor. RL is useful because scheduling decisions have delayed consequences. A VM that looks good for the current task may create future congestion, while a slightly more expensive assignment may preserve resources for future high-risk tasks.

However, using RL for every scheduling decision can increase overhead and reduce interpretability. In many cases, the deterministic score clearly identifies the best VM, and invoking RL is unnecessary. Therefore, ER-RL-DVFS uses RL selectively. The deterministic score-based scheduler is used when the best candidate is clearly separated from the others. The RL module is activated only when candidate scores are close or when the scheduling state is uncertain. This hybrid design combines the speed and interpretability of score-based scheduling with the adaptability of RL.

2.7 Motivating Scenario

Consider an IoE-cloud platform that receives three types of tasks: health-monitoring alerts, industrial inspection images, and periodic sensor aggregation. Health alerts have high risk and tight deadlines. Industrial inspection images have high computational demand and may require stronger VMs. Periodic sensor aggregation tasks are more delay tolerant and can usually be processed on lower-frequency resources if enough slack exists.

Suppose that two candidate VMs have similar completion-time estimates for a newly arrived health-monitoring alert. One VM has lower immediate energy consumption but is hosted by a PM with increasing queue pressure. The other VM consumes slightly more energy but is hosted by a more stable PM with lower future congestion risk. A purely deterministic energy-aware scheduler may select the first VM because of its lower immediate energy. However, this choice may increase queueing delay, trigger migration later, or cause SLA violation for subsequent high-risk tasks. A learning-based refinement module can learn from previous outcomes that the slightly more expensive assignment may produce better long-term reward.

This scenario explains the motivation behind ER-RL-DVFS. The scheduler should not optimize energy, delay, migration, or utilization independently. It should combine task risk, DVFS-aware execution estimation, hierarchical resource filtering, score-based fast decisions, selective RL refinement, and migration-aware correction. This combination is the main design motivation of the proposed framework.

3 Related Work and Comparative Positioning

3.1 DVFS-Based Energy-Aware Scheduling

DVFS-based scheduling has been widely used to reduce energy consumption by adjusting processor operating frequency and voltage according to workload intensity. In cloud and workflow systems, DVFS decisions must balance two conflicting objectives: reducing power consumption and preserving deadline satisfaction. Safari and Khorsand proposed energy-aware workflow scheduling for time-constrained tasks in DVFS-enabled cloud environments, showing that frequency scaling can reduce energy when task timing constraints are explicitly considered [8]. Calheiros and Buyya examined urgent bag-of-tasks applications and showed that DVFS can be effective when urgency and slack are included in scheduling decisions [9]. Stavrinides and Karatza studied QoS-aware and cost-aware scheduling with DVFS and approximate computation, further confirming that frequency control is useful for real-time workflow execution [10]. More recently, DVFS has also been investigated in heterogeneous and FaaS-oriented platforms, indicating that frequency-aware workload provisioning remains relevant in modern distributed systems [7].

Despite these contributions, pure DVFS-based scheduling is not sufficient for IoE-cloud workloads. In practice, reducing frequency may decrease energy but increase execution time, queueing delay, and SLA violation. Moreover, DVFS decisions are tightly connected to VM placement, PM utilization, migration overhead, and task urgency. Therefore, a more integrated scheduling design is required, where DVFS is not treated as an isolated energy-saving mechanism but as one component of a broader resource-management pipeline.

3.2 IoE, Fog, and Cloud Task Scheduling

Task scheduling in IoE, fog, and cloud platforms focuses on assigning heterogeneous tasks to available resources while satisfying service requirements. IoE workloads are more challenging than traditional batch workloads because tasks may originate from sensors, mobile devices, smart gateways, cameras, health-monitoring systems, or industrial devices. Such tasks differ in computational length, deadline tightness, risk level, bandwidth demand, and execution priority. Classical heuristics such as Round Robin, Min-Min, and Max-Min provide simple task-to-resource mapping rules, but they are not designed to jointly consider energy, DVFS state, migration overhead, and task risk.

A closely related study is the intelligent energy-efficient approach for managing IoE tasks in cloud platforms proposed by Javadpour et al. [11]. That work used DVFS information to sort and select cloud resources and introduced two IoE-cloud scheduling strategies: GIoTDVFS-SFB, based on a score-function mechanism, and GIoTDVFS-mGA, based on a microgenetic selection mechanism. The method considered cloud clusters, PMs, VMs, internal migration, and external migration, and it compared the proposed schedulers with PAFogIoTDVFS and EnergyAwareDVFS under benchmark settings such as 512×16 and 1024×32. This study is highly relevant to the present work because it provides DVFS-based IoE-cloud baselines and motivates the need for PM/VM-aware task assignment. However, its decision process is mainly based on deterministic scoring or metaheuristic search, while ER-RL-DVFS extends this line of work by adding energy-risk task prioritization, selective reinforcement learning, and adaptive near-tie decision refinement.

Recent heterogeneous real-time scheduling studies provide an additional perspective on energy-aware runtime adaptation. FRESH combines fault tolerance with dynamic scheduling on heterogeneous multiprocessor platforms and uses processor sleep states to reduce energy while maintaining real-time reliability [12]. ECO-FAST further studies energy-cognizant fault-tolerant scheduling for heterogeneous computing systems [13]. RESTORE addresses real-time task scheduling on temperature-aware FinFET multicore systems and explicitly couples task-to-core allocation with temperature-cognizant voltage/frequency scaling [14]. These methods are not direct cloud-VM baselines for the present experiments because their execution models target heterogeneous processor/core scheduling and fault/thermal constraints rather than virtualized IoE task-to-VM placement. Nevertheless, they are relevant to the design space because they demonstrate that DVFS-aware runtime adaptation should be evaluated together with timing, reliability, and heterogeneous-resource constraints. ER-RL-DVFS differs by operating at the cloud cluster–PM–VM hierarchy and by integrating task risk, queue pressure, migration cost, and selective learning into the placement decision.

3.3 Workflow and Task Scheduling in Cloud and Edge–Cloud Systems

Workflow and task scheduling studies typically assign tasks to resources under precedence, deadline, energy, or cost constraints. Traditional approaches such as Min-Min and Max-Min remain useful as baseline heuristics because they are simple and computationally inexpensive. However, they do not explicitly account for energy-risk-aware IoE priorities, DVFS-aware execution estimates, or migration-sensitive cloud states. More recent studies have explored heuristic, metaheuristic, and learning-based schedulers for cloud, fog, and edge–cloud systems. Dynamic workflow scheduling using deep learning has been studied for energy-efficient cloud execution [15], while reinforcement-learning-based multi-objective scheduling has been reported for cloud and cloud-edge environments [5, 6].

These works show the value of adaptive scheduling, especially when workload patterns are non-stationary. Nevertheless, many of them focus mainly on workflow execution, energy-cost trade-offs, or cloud-edge allocation, and they do not always integrate DVFS-aware profiling, migration-aware correction, task-level risk modeling, and hierarchical cluster–PM–VM candidate filtering in a single framework. ER-RL-DVFS is designed to address this gap by combining deterministic scheduling structure with learning-based refinement only when the scheduling state is uncertain.

3.4 Virtual Machine Placement, Consolidation, and Migration

VM placement and consolidation aim to improve resource utilization and reduce energy consumption by mapping VMs to PMs efficiently. Early work on energy-efficient resource management in virtualized data centers established consolidation and migration as central tools for reducing active server usage and improving energy efficiency [16]. Later studies investigated energy-aware dynamic VM consolidation, type-aware VM management, and migration-oriented optimization [17–20]. These methods are important because migration can prevent overloaded PMs, improve utilization, and reduce energy consumption when used carefully. However, migration also introduces memory-transfer delay, network overhead, and additional energy cost.

Recent DQN-based VM placement studies have considered carbon-aware, energy-aware, and SLA-aware placement decisions [4]. Such methods show that learning-based placement can adapt to changing data-center states. However, VM placement papers often focus on VM-to-PM mapping rather than per-task assignment under heterogeneous IoE deadlines, risk levels, and DVFS-aware execution times. In contrast, ER-RL-DVFS considers task-to-VM assignment while still accounting for PM-level utilization, VM queue pressure, DVFS state, and migration feasibility.

3.5 Research Gaps and Design Requirements

The literature shows that DVFS, VM migration, task scheduling, and reinforcement learning have each been studied extensively, but their integration remains incomplete for IoE-cloud scheduling. The main gaps are as follows. First, many task schedulers do not explicitly incorporate task risk and energy impact into the task-priority model. Second, DVFS-aware methods often optimize energy without sufficiently considering migration cost, future congestion, or adaptive learning effects. Third, VM placement and consolidation methods often focus on host-level optimization rather than per-task IoE scheduling. Fourth, learning-based schedulers may be computationally expensive if learning is used for every assignment decision. Fifth, closely related IoE-cloud DVFS methods such as GIoTDVFS-SFB and GIoTDVFS-mGA provide useful score-function and metaheuristic baselines, but they do not include selective RL activation for near-tie scheduling states.

These gaps lead to five design requirements. The scheduler should be risk-aware so that high-urgency and high-impact IoE tasks are prioritized properly. It should be DVFS-aware so that frequency and power states are included in resource selection. It should be migration-aware so that reassignment and migration are used only when their expected benefits justify their cost. It should be hierarchical so that the search space is reduced from clusters to PMs and then to VMs. Finally, it should be selectively adaptive so that RL is activated only when deterministic scoring is not sufficiently reliable.

Table 1 shows that most existing studies focus on only part of the scheduling problem. DVFS-based methods mainly address energy reduction, consolidation methods mainly address host utilization, migration-based methods mainly address overload control, and learning-based methods mainly address adaptivity. The closest prior IoE-cloud study is the DVFS-based GIoTDVFS line of work, which provides useful score-function and microgenetic baselines for PM/VM selection. However, ER-RL-DVFS differs by combining energy-risk task prioritization, DVFS-aware resource profiling, hierarchical cluster–PM–VM filtering, score-based fast scheduling, selective RL activation, and migration-aware correction in one scheduling framework.

4 System Model and Problem Formulation

4.1 IoE Task Model

Let $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ be the set of IoE tasks. Each task is defined as

$$t_j = \left(L_j, A_j, D_j, C_j, R_j^{ram}, R_j^{sto}, B_j, \rho_j\right), \tag{12}$$

where L_j is task length, A_j is arrival time, D_j is deadline, C_j is required CPU capacity, R_j^{ram} is memory demand, R_j^{sto} is storage demand, B_j is bandwidth demand, and $\rho_j \in [0, 1]$ is risk level. The laxity of t_j at current time τ is

$$\ell_j(\tau) = D_j - \tau - \min_{v_i \in \mathcal{V}} \hat{T}_{ij}, \tag{13}$$

where $\hat{T}_{ij}$ is the estimated processing time on VM v_i. A negative laxity means that the task is already at risk of missing its deadline. The urgency factor is normalized as

$$U_j(\tau) = \frac{1}{1 + \exp(\ell_j(\tau)/\kappa_u)}, \tag{14}$$

where κ_u controls the sharpness of the urgency transition. This logistic form avoids discontinuous priority jumps while still emphasizing tasks with limited slack.

4.2 Cloud Cluster, Physical Machine, and Virtual Machine Model

The cloud data center is modeled as a hierarchical infrastructure composed of clusters, physical machines (PMs), and virtual machines (VMs). Let $\mathcal{C} = \{c_1, \ldots, c_G\}$ denote the set of cloud clusters. Each cluster c_g contains a set of PMs $\mathcal{P}_g$, and each PM p_k hosts a set

Table 1: Comparative positioning of representative scheduling, DVFS, VM placement, and learning-based studies.

ID	Study	Year	Main scope	Main technique	DVFS	Migr.	RL	Positioning with respect to ER-RL-DVFS
R1	Beloglazov and Buyya [16]	2010	Virtualized data center	Energy-aware resource management	No	Yes	No	Foundational consolidation work, but no task-risk or learning layer
R2	Lee and Zomaya [21]	2012	Cloud resources	Energy-aware scheduling	No	No	No	Energy-aware resource use, but no DVFS-risk scheduling pipeline
R3	Calheiros and Buyya [9]	2014	Urgent bag-of-tasks	DVFS scheduling	Yes	No	No	Considers urgency and DVFS, but no PM/VM hierarchy or RL refinement
R4	Ding et al. [22]	2015	Cloud VMs	Deadline-aware VM scheduling	No	No	No	Deadline-aware VM scheduling, but not DVFS-RL integrated
R5	Safari and Khorsand [8]	2018	Cloud workflow	DVFS workflow scheduling	Yes	No	No	Strong DVFS workflow model, but no migration-aware adaptive policy
R6	Wang and Tianfield [19]	2018	Cloud data center	VM consolidation	No	Yes	No	Host-level consolidation, but no task-level risk scoring
R7	Stavrinides and Karatza [10]	2019	Workflow	QoS-aware and cost-aware DVFS	Yes	No	No	Uses DVFS for QoS and cost, but lacks adaptive candidate refinement
R8	Singh et al. [23]	2020	CloudSim data center	DVFS and VM consolidation	Yes	Yes	No	Combines DVFS and consolidation, but no energy-risk priority or RL policy
R9	Mirmohseni et al. [24]	2020	Cloud of Things	Markov learning	No	Partial	Yes	Learning-based resource allocation, but no explicit DVFS hierarchy
R10	Javadpour et al. [11]	2023	IoE-cloud	DVFS score function and microgenetic VM selection	Yes	Yes	No	Closely related IoE-cloud baseline with GIoTDVFS-SFB and GIoTDVFS-mGA, but no selective RL refinement
R11	Zeng et al. [25]	2022	Cloud data center	Adaptive DRL consolidation	No	Yes	Yes	Consolidation-centric, not task-priority-centric
R12	Wang et al. [26]	2023	Federated cloud	RL scheduling	No	Partial	Yes	Sustainable scheduling, but limited DVFS-aware task modeling
R13	Alex et al. [4]	2025	Cloud data center	DQN VM placement	No	Yes	Yes	Placement-centric, not per-task IoE-DVFS scheduling
R14	Chandrasiri and Meedeniya [15]	2025	Cloud workflow	Deep learning scheduling	No	No	Yes	Workflow focus, but no migration-aware DVFS mechanism
R15	Yu et al. [5]	2025	Cloud	RL multi-objective scheduling	Partial	No	Yes	Multi-objective task view, but weaker cluster–PM–VM hierarchy
R16	Zhang and Ou [6]	2025	Cloud-edge	RL energy-cost scheduling	Partial	No	Yes	Cloud-edge orientation, but no explicit migration-aware correction layer
R17	Machado et al. [7]	2026	Heterogeneous FaaS	DVFS workload provisioning	Yes	No	No	DVFS-focused provisioning, but not VM migration scheduling
R18	Moulik and Sharma [12]	2024	Heterogeneous multiprocessor	Fault-tolerant real-time scheduling with energy management	Partial	No	No	Heterogeneous real-time reliability and energy focus; no cloud VM hierarchy
R19	Moulik and Sarma [13]	2025	Heterogeneous computing	Energy-cognizant fault-tolerant scheduling	Partial	No	No	Runtime energy/reliability comparator at processor level rather than task-to-VM cloud placement
R20	Sharma et al. [14]	2022	FinFET multi-core	Temperature-aware real-time allocation with voltage/frequency scaling	Yes	No	No	Shows coupled thermal/DVFS adaptation under deadlines; no VM migration or IoE risk hierarchy
R21	ER-RL-DVFS	–	IoE-cloud	Energy-risk priority, DVFS profiling, hierarchy, RL, and migration	Yes	Yes	Yes	Proposed integrated framework

of VMs $\mathcal{V}_k$. The complete PM and VM sets are defined as

$$\mathcal{P} = \bigcup_{g=1}^{G} \mathcal{P}_g, \qquad \mathcal{V} = \bigcup_{p_k \in \mathcal{P}} \mathcal{V}_k. \tag{15}$$

This hierarchical representation is important because the proposed scheduler does not search all VMs directly. Instead, it first screens clusters, then PMs, and finally VMs. This structure reduces the scheduling search space and makes the assignment process more interpretable.

Each PM is characterized by

$$p_k = \left(CPU_k, RAM_k, STO_k, BW_k, P_k^{idle}, P_k^{max}, \mathcal{F}_k, u_k(\tau)\right), \tag{16}$$

where CPU_k, RAM_k, STO_k, and BW_k denote the total CPU, memory, storage, and bandwidth capacities of PM p_k, respectively. The terms P_k^{idle} and P_k^{max} are the idle and maximum power values, $\mathcal{F}_k = \{f_k^1, \ldots, f_k^q\}$ is the set of available DVFS frequency levels, and $u_k(\tau)$ is the utilization of PM p_k at scheduling step τ.

Each VM is characterized by

$$v_i = (CPU_i^{vm}, MIPS_i, RAM_i^{vm}, STO_i^{vm}, BW_i^{vm}, p(i), q_i(\tau), \eta_i), \tag{17}$$

where CPU_i^{vm}, $MIPS_i$, RAM_i^{vm}, STO_i^{vm}, and BW_i^{vm} represent the VM-level resource capacity and processing rate. The function $p(i)$ returns the host PM of VM v_i, $q_i(\tau)$ denotes its current queue workload, and $\eta_i \in (0, 1]$ denotes the virtualization efficiency factor. To avoid ambiguity, VM capacities are denoted with the superscript "vm", while PM capacities are indexed by k.

The binary task assignment variable is defined as

$$x_{ij} = \begin{cases} 1, & \text{if task } t_j \text{ is assigned to VM } v_i, \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

Each task must be assigned to exactly one feasible VM:

$$\sum_{i=1}^{|\mathcal{V}|} x_{ij} = 1, \quad \forall t_j \in \mathcal{T}. \tag{19}$$

The feasibility of assigning task t_j to VM v_i is represented by $\phi(i,j) \in \{0,1\}$, where $\phi(i,j) = 1$ indicates that the VM has sufficient available resources and $\phi(i,j) = 0$ otherwise. Therefore,

$$x_{ij} \leq \phi(i,j), \quad \forall v_i \in \mathcal{V},\ \forall t_j \in \mathcal{T}. \tag{20}$$

The VM-level resource constraints are written as

$$\sum_j x_{ij} C_j \leq CPU_i^{vm}, \quad \forall v_i \in \mathcal{V}, \tag{21}$$

$$\sum_j x_{ij} R_j^{ram} \leq RAM_i^{vm}, \quad \forall v_i \in \mathcal{V}, \tag{22}$$

$$\sum_j x_{ij} R_j^{sto} \leq STO_i^{vm}, \quad \forall v_i \in \mathcal{V}, \tag{23}$$

$$\sum_j x_{ij} B_j \leq BW_i^{vm}, \quad \forall v_i \in \mathcal{V}. \tag{24}$$

These constraints ensure that a task is not assigned to a VM that lacks sufficient CPU, memory, storage, or bandwidth. In the simulator, the available capacity of each VM is updated after each assignment, queue update, and migration operation.

4.3 DVFS-Based Energy Model

DVFS is used to adjust the processing frequency of PMs according to workload pressure and task urgency. Let $f_k(\tau) \in \mathcal{F}_k$ be the selected operating frequency of PM p_k at scheduling step τ. The normalized frequency ratio is

$$\varphi_k(\tau) = \frac{f_k(\tau)}{f_k^{max}}, \tag{25}$$

where f_k^{max} is the maximum frequency of PM p_k. For a task t_j assigned to VM v_i, the host PM is $p(i)$, and the execution-time estimate is

$$\hat{T}_{ij}(\tau) = \frac{L_j}{MIPS_i \eta_i \varphi_{p(i)}(\tau)} + W_i(\tau) + \delta_{ij}^{mig}, \tag{26}$$

where L_j is the task length, $W_i(\tau)$ is the expected waiting time in the VM queue, and δ_{ij}^{mig} is the additional delay caused by migration if migration is required. This equation shows the main DVFS trade-off: reducing the frequency may reduce energy, but it also increases execution time.

The estimated completion time is

$$\hat{C}_{ij} = \max(A_j, \tau) + \hat{T}_{ij}(\tau), \tag{27}$$

where A_j is the arrival time of task t_j. The normalized SLA violation severity is defined as

$$\psi_{ij}^{sev} = \max\left(0, \frac{\hat{C}_{ij} - D_j}{D_j - A_j + \epsilon}\right), \tag{28}$$

where D_j is the task deadline and ϵ is a small positive constant. This quantity measures how severely a task violates its deadline. The SLA violation rate used in the experimental section is reported separately as the percentage of tasks that miss their deadlines.

The DVFS-aware power of PM p_k is modeled as

$$P_k(\tau) = P_k^{idle} + \left(P_k^{max} - P_k^{idle}\right) u_k(\tau)^{\beta} \varphi_k(\tau)^{\zeta}, \tag{29}$$

where β captures the nonlinearity of utilization-based power consumption and ζ captures the sensitivity of power to frequency scaling. The energy contribution of assigning task t_j to VM v_i is estimated as

$$\hat{E}_{ij} = P_{p(i)}(\tau) \hat{T}_{ij}(\tau) + E_{ij}^{mig} + E_{ij}^{net}, \tag{30}$$

where E_{ij}^{mig} and E_{ij}^{net} are migration-related and network-related energy components. This formulation allows the scheduler to evaluate not only the processing cost of a VM, but also the additional overhead caused by moving or communicating the task.

4.4 Task Urgency and Energy-Risk Model

IoE tasks may differ substantially in urgency, computational demand, bandwidth demand, energy impact, and service risk. For example, an emergency alert or health-monitoring task should be prioritized differently from a delay-tolerant background aggregation task. Therefore, ER-RL-DVFS uses an energy-risk-aware priority score instead of relying only on task length or arrival order.

The priority score of task t_j is defined as

$$\Pi_j = \lambda_1 \bar{U}_j + \lambda_2 \bar{C}_j + \lambda_3 \bar{E}_j + \lambda_4 \rho_j + \lambda_5 \bar{B}_j, \tag{31}$$

where $\lambda_1, \ldots, \lambda_5$ are non-negative weights and $\sum_{r=1}^{5} \lambda_r = 1$. The terms $\bar{U}_j$, $\bar{C}_j$, $\bar{E}_j$, ρ_j, and $\bar{B}_j$ denote normalized urgency, CPU demand, expected energy impact, risk level, and bandwidth demand, respectively. They are computed as

$$\bar{C}_j = \frac{C_j}{\max_h C_h + \epsilon}, \tag{32}$$

$$\bar{E}_j = \frac{\min_i \hat{E}_{ij}}{\max_h \min_i \hat{E}_{ih} + \epsilon}, \tag{33}$$

$$\bar{B}_j = \frac{B_j}{\max_h B_h + \epsilon}, \tag{34}$$

$$\bar{U}_j = \frac{U_j}{\max_h U_h + \epsilon}. \tag{35}$$

The energy term is treated as an energy-impact indicator, not as a reward for high consumption. Tasks with large expected energy impact are prioritized earlier so that the scheduler can allocate them to more suitable resources before the system becomes congested. A larger Π_j means that the task should receive earlier scheduling attention or be assigned to a stronger candidate VM. This priority model is especially useful for IoE workloads because urgency and risk are often more important than task size alone.

4.5 Optimization Objective

The overall scheduling problem is formulated as a multi-objective minimization problem in which the scheduler must jointly balance energy consumption, completion time, SLA reliability, migration overhead, and resource balance. This formulation is necessary because these objectives are strongly coupled in DVFS-enabled IoE-cloud platforms. For example, reducing CPU frequency may decrease energy consumption but can increase execution time and SLA violation. Similarly, migration may reduce overload and improve deadline satisfaction, but it also introduces transfer delay, network overhead, and additional energy cost. Therefore, the scheduling objective should not optimize a single performance indicator in isolation.

The global objective is written as

$$\min_{x,f,m} \Omega = w_1 E^{tot} + w_2 T^{mk} + w_3 \Psi^{sla} + w_4 C^{mig} + w_5 \Phi^{imb} - w_6 G^{util}, \tag{36}$$

where E^{tot} is total energy consumption, T^{mk} is makespan, Ψ^{sla} is SLA violation severity, C^{mig} is migration cost, Φ^{imb} is load imbalance, and G^{util} is utilization gain. The coefficients $w_1, \ldots, w_6$ control the relative importance of the objective components. The negative sign before G^{util} indicates that higher useful utilization is desirable and should reduce the overall objective value. In contrast, energy, delay, SLA violation, migration cost, and load imbalance are penalty terms.

The objective components are defined as

$$E^{tot} = \sum_j \sum_i x_{ij} \hat{E}_{ij}, \tag{37}$$

$$T^{mk} = \max_j \sum_i x_{ij} \hat{C}_{ij}, \tag{38}$$

$$\Psi^{sla} = \frac{1}{n} \sum_j \sum_i x_{ij} \psi_{ij}^{sev}, \tag{39}$$

$$C^{mig} = \sum_j \sum_i x_{ij} m_{ij} \left(\mu_1 D_{ij}^{mig} + \mu_2 E_{ij}^{mig} \right), \tag{40}$$

$$G^{util} = \frac{1}{|\mathcal{P}|} \sum_k u_k, \tag{41}$$

$$\Phi^{imb} = \sqrt{\frac{1}{|\mathcal{P}|} \sum_k (u_k - \bar{u})^2}. \tag{42}$$

The first term, E^{tot}, measures the energy induced by assigning tasks to VMs, including DVFS-dependent execution energy and possible migration or network energy. The second term, T^{mk}, captures the global completion time of the workload. The third term, Ψ^{sla}, represents the average deadline-violation severity, which is especially important for high-risk IoE tasks. The fourth term, C^{mig}, penalizes migration decisions according to both migration delay and migration energy. The last two terms encourage efficient resource usage while avoiding excessive concentration of workload on a small number of PMs.

The binary variable $m_{ij} \in \{0, 1\}$ indicates whether migration is required when assigning task t_j to VM v_i. The main constraints are

$$\sum_i x_{ij} = 1, \quad x_{ij} \in \{0, 1\}, \tag{43}$$

$$x_{ij} \le \phi(i, j), \tag{44}$$

$$f_k \in \mathcal{F}_k, \quad \forall p_k \in \mathcal{P}, \tag{45}$$

$$\hat{C}_{ij} \le D_j + \Delta_j^{tol} \quad \text{for high-risk tasks}, \tag{46}$$

$$m_{ij} \le \phi_{ij}^{mig}, \tag{47}$$

where $\phi(i, j)$ indicates whether VM v_i is feasible for task t_j, Δ_j^{tol} is the allowed tolerance for high-risk tasks, and ϕ_{ij}^{mig} indicates whether migration is feasible. These constraints ensure that each task is assigned to one feasible VM, the selected frequency belongs to the PM DVFS set, high-risk tasks are protected by stricter deadline tolerance, and migration is allowed only when the corresponding migration path is feasible.

This optimization problem is non-convex and combinatorial because it involves binary assignment variables, discrete DVFS frequency levels, migration decisions, queue-dependent completion times, and nonlinear power terms. Solving it exactly by exhaustive search would be impractical for online IoE-cloud scheduling, especially when the number of VMs and incoming tasks grows. Therefore, ER-RL-DVFS does not attempt to solve Eq. (36) directly. Instead, it uses the objective as the guiding principle for a structured heuristic-learning framework that combines task prioritization, DVFS-aware profiling, hierarchical candidate filtering, score-based fast assignment, selective RL refinement, and migration-aware correction.

4.6 Relationship Between the Multi-Objective Formulation and the Scheduling Procedure

The optimization model in Section 4 is a design-level scalarization and not a claim that the online scheduler solves the complete mixed discrete multi-objective problem to global optimality at every arrival. The role of Eq. (36) is to define the quantities that the online policy must trade off. ER-RL-DVFS implements tractable surrogates of these terms at successive stages: the priority score represents deadline/risk pressure; DVFS-aware profiling estimates the energy and execution-time consequences of placement; hierarchical filtering restricts the feasible decision set; the fast score combines normalized energy, completion time, queue pressure, migration cost, and host health; the RL reward is the step-level counterpart of the energy, time, SLA, migration, utilization, and imbalance terms; and the migration stage evaluates corrective actions only when their expected benefit offsets migration cost. Thus, the algorithm is a structured online heuristic guided by the formulated objective, rather than an exact optimizer of Eq. (36). Accordingly, no claim of global optimality is made for the online scheduling procedure.

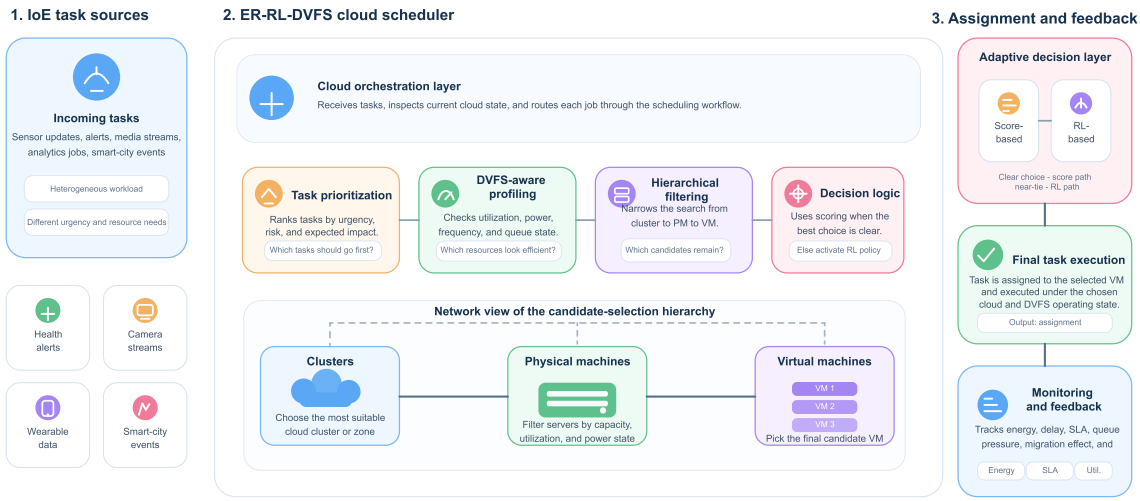


Figure 1: Architecture of the proposed ER-RL-DVFS scheduling framework. Incoming IoE tasks are prioritized, matched with DVFS-aware PM/VM profiles, filtered through the cluster–PM–VM hierarchy, and assigned using score-based or RL-based decision logic.

5 Proposed ER-RL-DVFS Scheduling Framework

5.1 Framework Overview

ER-RL-DVFS follows a layered scheduling pipeline, as illustrated in Fig. 1. The main purpose of this pipeline is to transform the difficult multi-objective optimization problem in Eq. (36) into a sequence of interpretable and computationally manageable scheduling steps. Rather than evaluating all possible task–VM–frequency–migration combinations, the framework progressively reduces the decision space and applies more adaptive logic only when needed.

First, arrived IoE tasks are ranked using the energy-risk priority model in Eq. (31). This step ensures that tasks with high urgency, high risk, high computational demand, or large expected energy impact are considered earlier. Second, PMs and VMs are profiled according to their current utilization, queue pressure, DVFS frequency state, power level, and available resources. This step prevents the scheduler from selecting a VM only because it has nominal capacity while ignoring its host power state or queue condition. Third, the scheduler performs hierarchical candidate filtering from clusters to PMs and finally to VMs. This avoids a flat search over all VMs and improves online scalability.

Fourth, a deterministic score-based fast scheduler is used when the best VM is clearly separated from the remaining candidates. This preserves low decision overhead and keeps the assignment process interpretable. Fifth, the RL module is activated only when the deterministic score margin is small or the scheduling state is uncertain. In other words, ER-RL-DVFS does not use RL for every task. It uses RL selectively to refine near-tie decisions where deterministic scoring may be insufficient. Finally, migration-aware correction is applied if the selected VM is infeasible, overloaded, or likely to violate the SLA. This final correction layer makes the method robust against sudden queue changes and resource-state variations.

Overall, the framework is designed to combine the strengths of rule-based and learning-based scheduling. The deterministic layers provide structure, speed, and interpretability, while the RL layer provides adaptability in uncertain states. The migration-aware layer further improves reliability by correcting assignments that become infeasible or risky after candidate selection.

Figure 2 summarizes the operational workflow of the proposed ER-RL-DVFS scheduler. After IoE tasks arrive, the scheduler first evaluates their priority and then updates the DVFS-aware resource profiles of clusters, PMs, and VMs. The candidate resource space is then reduced through hierarchical filtering, followed by fast candidate scoring. If the best VM is clearly identified, the scheduler directly assigns the task to that VM. Otherwise, the RL-based selection module is activated to handle uncertain or near-tie decisions. Finally, migration or reassignment checking is performed before executing the scheduling decision and updating the reward, Q-values, and system state.

To improve readability and implementation clarity, the proposed ER-RL-DVFS framework is decomposed into seven operational algorithms. Algorithm 1 ranks ready IoE tasks, Algorithm 2 updates DVFS-aware PM/VM profiles, Algorithm 3 constructs the reduced candidate set, Algorithm 4 handles clearly separated candidates, and Algorithm 5 specifies selective near-tie refinement. Algorithm 6 formalizes the reward/Q-value update associated with an RL decision, while Algorithm 7 is the seventh formal procedure and finalizes the assignment through migration-aware feasibility correction. Although the reward definition is presented before the correction algorithm for numbering clarity, its update is applied after the final assignment outcome is observed. This decomposition makes both the algorithm numbering and the execution dependencies explicit.

The detailed task-priority computation and queue-ordering procedure are formalized in Algorithm 1.

The ordered queue generated by Algorithm 1 is used as the input of the DVFS-aware profiling and candidate-selection stages. Therefore, high-risk and deadline-sensitive tasks are processed before the scheduler evaluates the available PM and VM resources.

5.2 Energy-Risk-Aware Task Prioritization

The priority of each task is recomputed at every scheduling window because urgency changes as time advances. A task that is not urgent at the moment of arrival may become urgent if it waits too long in the queue. Therefore, using a static priority value computed only at arrival time is insufficient for dynamic IoE workloads. Let $\mathcal{T}^{ready}(\tau)$ denote the set of arrived but unscheduled tasks at scheduling step τ. The ready task queue is sorted as

$$\mathcal{Q}(\tau) = \mathrm{sort}_{t_j \in \mathcal{T}^{ready}(\tau)} \left(\Pi_j, A_j, D_j\right), \tag{48}$$

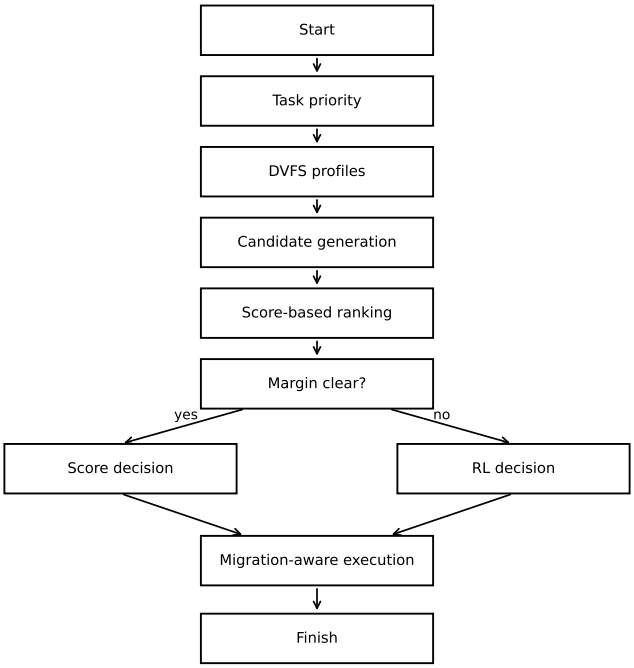


Figure 2: Compact workflow of the proposed ER-RL-DVFS scheduling process.

Algorithm 1 Energy-Risk-Aware Task Prioritization

Require: Ready task set $\mathcal{T}^{ready}(\tau)$, current time τ, VM set $\mathcal{V}$, priority weights $\lambda_1, \ldots, \lambda_5$
Ensure: Ordered task queue $\mathcal{Q}(\tau)$
for each task $t_j \in \mathcal{T}^{ready}(\tau)$ **do**
  Estimate the minimum execution time $\min_{v_i \in \mathcal{V}} \hat{T}_{ij}$
  Compute task laxity:

$$\ell_j(\tau) = D_j - \tau - \min_{v_i \in \mathcal{V}} \hat{T}_{ij}$$

  Compute urgency factor $U_j(\tau)$ using Eq. (14)
  Normalize urgency, CPU demand, expected energy impact, and bandwidth demand:

$$\bar{U}_j, \quad \bar{C}_j, \quad \bar{E}_j, \quad \bar{B}_j$$

  Compute the energy-risk priority score:

$$\Pi_j = \lambda_1 \bar{U}_j + \lambda_2 \bar{C}_j + \lambda_3 \bar{E}_j + \lambda_4 \rho_j + \lambda_5 \bar{B}_j$$

Sort all tasks in descending order of Π_j
Break ties using earlier arrival time A_j and earlier deadline D_j
Construct the ordered task queue $\mathcal{Q}(\tau)$
return $\mathcal{Q}(\tau)$

where tasks are primarily sorted by descending priority score Π_j, while ties are broken by earlier arrival time and earlier deadline.

This ordering prevents low-risk or delay-tolerant tasks from consuming favorable VM slots before urgent or high-impact tasks. It also improves the ability of DVFS-aware scheduling to exploit slack correctly. If urgent tasks are identified late, the scheduler may already have reduced frequency or filled suitable queues with low-priority tasks. By ranking tasks before resource selection, ER-RL-DVFS gives the resource-allocation layer a better task order and reduces the chance of avoidable SLA violations. This is particularly important in Smart-City IoT and health-monitoring scenarios, where a small number of high-risk tasks may be more important than a large number of low-risk background tasks.

5.3 DVFS-Based Resource Profiling

For each PM p_k, ER-RL-DVFS maintains a DVFS-aware profile vector:

$$\mathbf{z}_k(\tau) = \left[u_k(\tau), \varphi_k(\tau), P_k(\tau), RAM_k^{free}, BW_k^{free}, \bar{Q}_k(\tau) \right], \tag{49}$$

where $\bar{Q}_k(\tau)$ denotes the average queue pressure of the VMs hosted by PM p_k. This vector summarizes both resource availability and energy-related state. A PM with low utilization may not always be the best choice if its frequency state, power level, or hosted VM queues make it unsuitable for the current task.

For each VM v_i, the profile vector is

$$\mathbf{h}_i(\tau) = \left[MIPS_i^{eff}, RAM_i^{free}, BW_i^{free}, Q_i(\tau), \hat{E}_{ij}, \hat{T}_{ij} \right]. \tag{50}$$

This VM-level profile captures processing capability, available memory and bandwidth, queue pressure, estimated energy, and estimated

Algorithm 2 DVFS-Aware PM and VM Profiling

Require: PM set $\mathcal{P}$, VM set $\mathcal{V}$, current task t_j, current time τ, DVFS levels $\mathcal{F}_k$
Ensure: PM profiles $\{\mathbf{z}_k(\tau)\}$ and VM profiles $\{\mathbf{h}_i(\tau)\}$
for each PM $p_k \in \mathcal{P}$ **do**
  Read current utilization $u_k(\tau)$ and selected frequency $f_k(\tau)$
  Compute the normalized frequency ratio:

$$\varphi_k(\tau) = \frac{f_k(\tau)}{f_k^{max}}$$

  Estimate DVFS-aware power $P_k(\tau)$ using Eq. (29)
  Compute free memory RAM_k^{free} and free bandwidth BW_k^{free}
  Compute average queue pressure $\bar{Q}_k(\tau)$ of hosted VMs
  Build the PM profile:

$$\mathbf{z}_k(\tau) = [u_k(\tau), \varphi_k(\tau), P_k(\tau), RAM_k^{free}, BW_k^{free}, \bar{Q}_k(\tau)]$$

for each VM $v_i \in \mathcal{V}$ **do**
  Identify the host PM $p(i)$
  Compute effective processing rate:

$$MIPS_i^{eff} = MIPS_i \eta_i \varphi_{p(i)}(\tau)$$

  Estimate execution time $\hat{T}_{ij}$ using Eq. (26)
  Estimate energy consumption $\hat{E}_{ij}$ using Eq. (30)
  Build the VM profile:

$$\mathbf{h}_i(\tau) = [MIPS_i^{eff}, RAM_i^{free}, BW_i^{free}, Q_i(\tau), \hat{E}_{ij}, \hat{T}_{ij}]$$

return $\{\mathbf{z}_k(\tau)\}$ and $\{\mathbf{h}_i(\tau)\}$

execution time. These profile vectors are recomputed as tasks are assigned and queues change, allowing the scheduler to respond to the current system state rather than relying on static VM specifications.

The PM score for task t_j is defined as

$$R_{kj}^{pm} = a_1(1-u_k) + a_2(1-\bar{P}_k) + a_3\bar{F}_{kj} + a_4\bar{S}_{kj} - a_5\bar{Q}_k, \tag{51}$$

where $\bar{P}_k$ is normalized power, $\bar{F}_{kj}$ is frequency suitability, and $\bar{S}_{kj}$ is slack suitability. The terms $1-u_k$ and $1-\bar{P}_k$ favor PMs with available capacity and lower normalized power, while $\bar{F}_{kj}$ and $\bar{S}_{kj}$ capture whether the PM frequency state is suitable for the task slack. The negative queue-pressure term discourages assigning tasks to PMs whose VMs are already congested.

Algorithm 2 summarizes how ER-RL-DVFS updates PM-level and VM-level profiles before resource filtering.

The updated PM and VM profiles produced by Algorithm 2 provide the utilization, queue, frequency, power, energy, and execution-time information required by the hierarchical candidate filtering stage.

5.4 Hierarchical Cluster–PM–VM Candidate Selection

A flat search over all VMs has complexity $O(n|\mathcal{V}|)$ and becomes expensive when the number of VMs increases. In addition, a flat search is weakly interpretable because the scheduler directly chooses from a large set of VMs without explaining the cluster-level and PM-level reasoning behind the decision. ER-RL-DVFS reduces this cost through hierarchical candidate filtering. The method first evaluates clusters, then PMs inside the selected clusters, and finally VMs inside the selected PMs.

The cluster score is defined as

$$R_{gj}^{cl} = b_1\overline{CPU}_g^{free} + b_2\overline{RAM}_g^{free} + b_3(1-\overline{P}_g) + b_4(1-\overline{SLA}_g) - b_5\overline{MIG}_g. \tag{52}$$

The first two terms favor clusters with sufficient available CPU and memory. The third term favors clusters with lower normalized power. The fourth term favors clusters with better recent SLA behavior, while the final term penalizes clusters with high migration pressure. Therefore, the cluster score is not a simple capacity measure; it reflects resource availability, energy state, service reliability, and migration overhead.

Only the top K_c clusters are retained. Within each retained cluster, the top K_p PMs are selected using Eq. (51). Within each retained PM, the top K_v feasible VMs are evaluated. The candidate VM set is

$$\mathcal{A}_j = \bigcup_{g \in \mathcal{C}_j^{K_c}} \bigcup_{p_k \in \mathcal{P}_{gj}^{K_p}} \mathcal{V}_{kj}^{K_v}. \tag{53}$$

The approximate search-space reduction ratio is

$$\Gamma = 1 - \frac{K_c K_p K_v}{|\mathcal{V}|}. \tag{54}$$

A larger Γ means that fewer VMs are evaluated during final decision-making. This is one reason why the proposed framework remains suitable for online scheduling.

The hierarchical filtering process has two practical advantages. First, it improves scalability by avoiding exhaustive VM scoring. Second, it improves deployment interpretability because the final decision can be traced through a clear path: selected cluster, selected PM, selected VM. This is useful in cloud management because operators can understand whether a task was assigned to a VM because of energy, queue pressure, deadline protection, or migration avoidance.

Figure 3 provides a network-oriented view of the core scheduling logic in ER-RL-DVFS. In contrast to a flat VM search, the proposed framework progressively narrows the decision space by moving from task ordering to hierarchical resource filtering across clusters, PMs, and VMs. This visual representation highlights the main operational idea of the framework: reducing the search space while preserving an interpretable and deployment-oriented assignment process.

The hierarchical search-space reduction procedure used to construct the reduced VM candidate set is presented in Algorithm 3.

The candidate VM set $\mathcal{A}_j$ returned by Algorithm 3 is passed to the fast score-based scheduling policy. This reduces the number of VMs evaluated in the final decision stage while preserving a structured cluster–PM–VM assignment path.

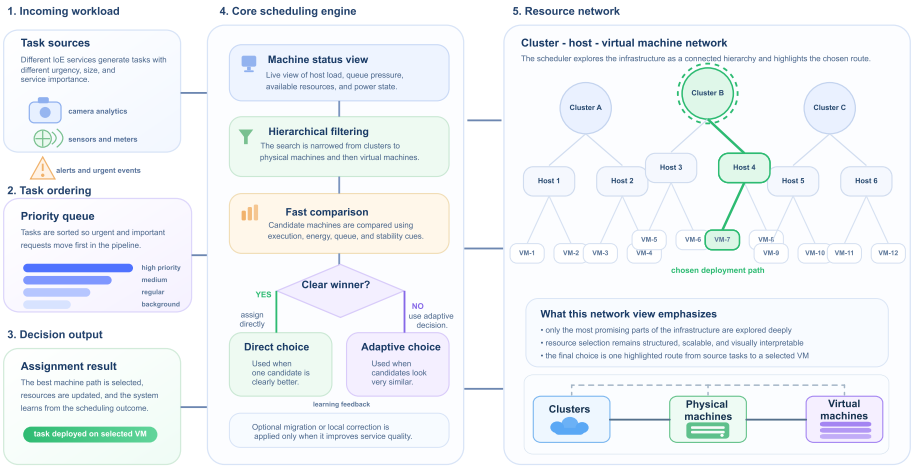


Figure 3: Network-oriented illustration of the core ER-RL-DVFS scheduling mechanism.

Algorithm 3 Hierarchical Cluster–PM–VM Candidate Filtering

Require: Task t_j, cluster set $\mathcal{C}$, PM set $\mathcal{P}$, VM set $\mathcal{V}$, filtering parameters K_c, K_p, and K_v
Ensure: Candidate VM set $\mathcal{A}_j$

for each cluster $c_g \in \mathcal{C}$ **do**
  Compute available CPU and memory ratios of cluster c_g
  Compute normalized cluster power $\overline{P}_g$
  Compute recent SLA loss $\overline{SLA}_g$
  Compute migration pressure $\overline{MIG}_g$
  Compute cluster score R_{gj}^{cl} using Eq. (52)
Select the top K_c clusters and denote them by $\mathcal{C}_j^{K_c}$
for each selected cluster $c_g \in \mathcal{C}_j^{K_c}$ **do**
  for each PM $p_k \in \mathcal{P}_g$ **do**
    Compute PM score R_{kj}^{pm} using Eq. (51)
  Select the top K_p PMs from cluster c_g
for each selected PM p_k **do**
  for each VM $v_i \in \mathcal{V}_k$ **do**
    if v_i satisfies the CPU, memory, storage, and bandwidth requirements of t_j **then**
      Add v_i to the feasible VM list of p_k
  Select the top K_v feasible VMs according to availability, queue pressure, and estimated execution suitability
Construct the candidate VM set:

$$\mathcal{A}_j = \bigcup_{g \in \mathcal{C}_j^{K_c}} \bigcup_{p_k \in \mathcal{P}_{gj}^{K_p}} \mathcal{V}_{kj}^{K_v}$$

return $\mathcal{A}_j$

5.5 Score-Based Fast Scheduling Policy

For each candidate VM $v_i \in \mathcal{A}_j$, the deterministic fast score is

$$S_{ij} = \theta_1 \bar{A}_{ij} + \theta_2(1 - \bar{E}_{ij}) + \theta_3(1 - \bar{T}_{ij}) + \theta_4(1 - \bar{Q}_i) + \theta_5(1 - \bar{M}_{ij}) + \theta_6 \bar{H}_{ij}, \tag{55}$$

where $\bar{A}_{ij}$ is normalized availability, $\bar{E}_{ij}$ is normalized energy, $\bar{T}_{ij}$ is normalized completion time, $\bar{Q}_i$ is normalized queue pressure, $\bar{M}_{ij}$ is normalized migration cost, and $\bar{H}_{ij}$ is host health. The best and second-best candidate scores are

$$S_j^{(1)} = \max_i S_{ij}, \qquad S_j^{(2)} = \max_{i: S_{ij} < S_j^{(1)}} S_{ij}. \tag{56}$$

The score margin is

$$\Delta_j^S = S_j^{(1)} - S_j^{(2)}. \tag{57}$$

If $\Delta_j^S \geq \delta_S$, the deterministic choice is accepted:

$$v_j^* = \arg\max_{v_i \in \mathcal{A}_j} S_{ij}. \tag{58}$$

Otherwise, the decision is considered uncertain and the RL module is activated. This selective activation avoids unnecessary RL overhead when the deterministic choice is already clear. Algorithm 4 describes the deterministic VM scoring procedure and the condition used to decide whether RL refinement is required.

If Algorithm 4 returns $\mathcal{F}_{RL} = 0$, the selected VM is directly forwarded to the migration-aware correction stage. If it returns $\mathcal{F}_{RL} = 1$, the scheduler activates Algorithm 5 to resolve the uncertain near-tie decision.

5.6 RL-Based Adaptive VM Selection

The RL module is used only for uncertain candidate states. For task t_j and candidate VM v_i, the state vector is

$$s_\tau = \left[\bar{u}_{p(i)}, \bar{P}_{p(i)}, \varphi_{p(i)}, \bar{Q}_i, \bar{E}_{ij}, \bar{T}_{ij}, \bar{M}_{ij}, \Pi_j, \psi_{ij}^{sev}\right]. \tag{59}$$

The action set is the candidate VM set:

$$\mathcal{A}(s_\tau) = \{a_i : a_i \equiv \text{assign } t_j \text{ to } v_i,\ v_i \in \mathcal{A}_j\}. \tag{60}$$

Algorithm 4 Score-Based Fast VM Selection

Require: Task t_j, candidate VM set $\mathcal{A}_j$, score-margin threshold δ_S
Ensure: Selected VM candidate v_j^{cand} and RL activation flag $\mathcal{F}_{RL}$
for each candidate VM $v_i \in \mathcal{A}_j$ **do**
  Estimate normalized availability $\bar{A}_{ij}$
  Estimate normalized energy $\bar{E}_{ij}$
  Estimate normalized completion time $\bar{T}_{ij}$
  Estimate normalized queue pressure $\bar{Q}_i$
  Estimate normalized migration cost $\bar{M}_{ij}$
  Estimate host health $\bar{H}_{ij}$
  Compute deterministic VM score:

$$S_{ij} = \theta_1 \bar{A}_{ij} + \theta_2(1 - \bar{E}_{ij}) + \theta_3(1 - \bar{T}_{ij}) + \theta_4(1 - \bar{Q}_i) + \theta_5(1 - \bar{M}_{ij}) + \theta_6 \bar{H}_{ij}$$

Identify the best score $S_j^{(1)}$ and second-best score $S_j^{(2)}$
Compute the score margin:

$$\Delta_j^S = S_j^{(1)} - S_j^{(2)}$$

if $\Delta_j^S \geq \delta_S$ **then**
  Select $v_j^{cand} = \arg\max_{v_i \in \mathcal{A}_j} S_{ij}$
  Set $\mathcal{F}_{RL} = 0$
else
  Set $v_j^{cand} = \emptyset$
  Set $\mathcal{F}_{RL} = 1$
return v_j^{cand} and $\mathcal{F}_{RL}$

Algorithm 5 Selective RL-Based VM Selection

Require: Task t_j, candidate VM set $\mathcal{A}_j$, Q-table or policy Q, exploration rate ϵ
Ensure: RL-selected VM v_j^{RL} and selected action a_τ
Construct the RL state representation s_τ using task, VM, PM, energy, delay, queue, migration, and SLA-related features:

$$s_\tau = [\bar{u}_{p(i)}, \bar{P}_{p(i)}, \varphi_{p(i)}, \bar{Q}_i, \bar{E}_{ij}, \bar{T}_{ij}, \bar{M}_{ij}, \Pi_j, \psi_{ij}^{sev}]$$

Define the action set:

$$\mathcal{A}(s_\tau) = \{a_i : a_i \equiv \text{assign } t_j \text{ to } v_i,\ v_i \in \mathcal{A}_j\}$$

Generate a random number $r \in [0, 1]$
if $r < \epsilon$ **then**
  Select a random action a_τ from $\mathcal{A}(s_\tau)$
else
  Select the greedy action:

$$a_\tau = \arg\max_{a \in \mathcal{A}(s_\tau)} Q(s_\tau, a)$$

Set v_j^{RL} as the VM corresponding to action a_τ
return v_j^{RL} and a_τ

For a direct tabular implementation of Algorithm 5, the variable physical VM set is mapped to a stable local action representation. The nine normalized features in Eq. (59) are discretized component-wise into five equal-width bins on $[0, 1]$, with boundary values clipped to the nearest endpoint bin. The state key is therefore the nine-dimensional bin tuple $\mathbf{b}_\tau$. After hierarchical filtering, candidate VMs are sorted by the deterministic fast score and actions are indexed by local rank rather than by a permanent global VM identifier. Thus, the tabular key is $Q(\mathbf{b}_\tau, r)$, where $r \in \{1, \ldots, |\mathcal{A}_j|\}$ denotes the current candidate rank. Unseen state–rank pairs are initialized to zero and may be stored sparsely. This convention makes the tabular action representation well defined even when candidate identity and candidate-set size vary between scheduling decisions. The controlled stochastic benchmark used for the numerical results does not instantiate this Q-table; consequently, the discretization above specifies the executable form of the proposed RL stage but is not presented as a hidden source of the reported benchmark values.

The action is selected using an ϵ-greedy policy:

$$a_\tau = \begin{cases} \text{random action from } \mathcal{A}(s_\tau), & \text{with probability } \epsilon, \\ \arg\max_a Q(s_\tau, a), & \text{otherwise.} \end{cases} \tag{61}$$

When the deterministic score margin is insufficient, the selective RL-based VM selection procedure in Algorithm 5 is activated.

The VM selected by Algorithm 5 is not immediately finalized. Instead, it is checked by the migration-aware correction mechanism in Algorithm 7 to ensure feasibility, deadline safety, and overload avoidance.

5.7 Reward Design and Learning Update

The reward is the step-level maximization counterpart of the global minimization objective in Eq. (36). It penalizes energy, delay, SLA violation, migration cost, and load imbalance, while rewarding useful utilization:

$$r_\tau = -\omega_1 \bar{E}_\tau - \omega_2 \bar{T}_\tau - \omega_3 \bar{\Psi}_\tau^{sla} - \omega_4 \bar{C}_\tau^{mig} + \omega_5 \bar{G}_\tau^{util} - \omega_6 \bar{\Phi}_\tau^{imb}. \tag{62}$$

For tabular Q-learning, the update rule is

$$Q(s_\tau, a_\tau) \leftarrow Q(s_\tau, a_\tau) + \eta \left[r_\tau + \gamma \max_{a'} Q(s_{\tau+1}, a') - Q(s_\tau, a_\tau) \right]. \tag{63}$$

For a DQN extension, the temporal-difference loss can be written as

$$\mathcal{L}(\Theta) = \left(y_\tau - Q(s_\tau, a_\tau; \Theta)\right)^2, \tag{64}$$

Algorithm 6 Reward Computation and Q-Value Update

Require: State s_τ, action a_τ, selected VM v_j^*, next state $s_{\tau+1}$, learning rate η, discount factor γ
Ensure: Updated Q-value $Q(s_\tau, a_\tau)$
Compute normalized energy penalty $\bar{E}_\tau$
Compute normalized delay penalty $\bar{T}_\tau$
Compute normalized SLA violation penalty $\bar{\Psi}_\tau^{sla}$
Compute normalized migration cost $\bar{C}_\tau^{mig}$
Compute normalized utilization gain $\bar{G}_\tau^{util}$
Compute normalized load-imbalance penalty $\bar{\Phi}_\tau^{imb}$
Compute the immediate reward:

$$r_\tau = -\omega_1 \bar{E}_\tau - \omega_2 \bar{T}_\tau - \omega_3 \bar{\Psi}_\tau^{sla} - \omega_4 \bar{C}_\tau^{mig} + \omega_5 \bar{G}_\tau^{util} - \omega_6 \bar{\Phi}_\tau^{imb}$$

Compute the temporal-difference target:

$$y_\tau = r_\tau + \gamma \max_{a'} Q(s_{\tau+1}, a')$$

Update Q-value:

$$Q(s_\tau, a_\tau) \leftarrow Q(s_\tau, a_\tau) + \eta \left[y_\tau - Q(s_\tau, a_\tau) \right]$$

return updated $Q(s_\tau, a_\tau)$

Algorithm 7 Migration-Aware Assignment Correction

Require: Task t_j, provisional VM v_j^{cand}, current cluster/PM/VM profiles, deadline D_j
Ensure: Final VM v_j^* and correction mode $\mathcal{M}^*$
Evaluate feasibility, predicted completion time, queue/host overload, and SLA risk for v_j^{cand}
if v_j^{cand} is feasible and predicted to satisfy the admissibility/SLA checks **then**
    Set $v_j^* = v_j^{cand}$ and $\mathcal{M}^* = none$
else
    Construct feasible local-reassignment candidates on the same PM
    Construct feasible internal-migration candidates within the same cluster
    Construct feasible external-migration candidates in other clusters
    for each feasible correction mode $\mathcal{M} \in \{local, internal, external\}$ **do**
        Estimate $C_{\mathcal{M}}^{mig}$, $\Psi_{\mathcal{M}}^{sla}$, and $E_{\mathcal{M}}$
        Compute $J_{\mathcal{M}} = C_{\mathcal{M}}^{mig} + \omega_{sla} \Psi_{\mathcal{M}}^{sla} + \omega_E E_{\mathcal{M}}$
    if at least one feasible correction candidate exists **then**
        Select $\mathcal{M}^* = \arg\min_{\mathcal{M}} J_{\mathcal{M}}$ and its associated VM as v_j^*
    else
        Keep the least-violating feasible assignment available under the admission policy
return v_j^* and $\mathcal{M}^*$

where

$$y_\tau = r_\tau + \gamma \max_{a'} Q(s_{\tau+1}, a'; \Theta^-). \tag{65}$$

The proposed scheduler is specified with a lightweight tabular/discretized RL refinement so that a direct implementation can avoid invoking a deep network at every task decision, while the formulation remains compatible with a DQN replacement. In the numerical materials supplied with this study, however, the stochastic benchmark harness represents algorithms by calibrated comparative performance profiles rather than executing online Q-table updates or neural-network training. Accordingly, no convergence, episode-count, sample-efficiency, or policy-optimality claim is inferred from the benchmark results.

Algorithm 6 formalizes the reward computation and Q-value update. During execution, this update is applied after the provisional assignment has passed the migration-aware correction in Algorithm 7, so that the reward reflects the realized final assignment outcome.

The updated Q-values obtained from Algorithm 6 are reused in later calls to Algorithm 5, allowing a direct implementation of ER-RL-DVFS to adapt its near-tie VM selection policy over time.

The provisional VM selected by Algorithm 5 is then checked by the seventh procedure, Algorithm 7, before the assignment is finalized.

5.8 Migration-Aware Scheduling Decision

If the selected VM is infeasible, overloaded, or likely to violate the deadline, a migration-aware correction step is performed. Three alternatives are considered: local reassignment within the same PM, internal migration within the same cluster, and external migration to another cluster. The migration cost is modeled as

$$C_{ij}^{mig} = \xi_1 \frac{Mem_i^{dirty}}{BW_{src,dst}} + \xi_2 E_{ij}^{mig} + \xi_3 \mathbb{I}(cluster_{src} \neq cluster_{dst}), \tag{66}$$

where Mem_i^{dirty} is dirty memory size, $BW_{src,dst}$ is available migration bandwidth, and the last term penalizes external migration. The selected correction mode is

$$\mathcal{M}^* = \underset{\mathcal{M} \in \{local, internal, external\}}{\arg\min} \left(C_{\mathcal{M}}^{mig} + \omega_{sla} \Psi_{\mathcal{M}}^{sla} + \omega_E E_{\mathcal{M}} \right). \tag{67}$$

Thus, migration is not triggered simply because another VM is available. It is used only when the expected improvement in SLA, energy, or utilization justifies the overhead.

The correction logic is formalized in Algorithm 7, which is the seventh operational procedure referenced in the framework description.

5.9 Computational Complexity

Based on Algorithms 1–6, the online complexity of ER-RL-DVFS is mainly determined by task-priority computation, DVFS-aware profiling, hierarchical candidate filtering, deterministic VM scoring, selective RL activation, and migration-aware correction.

Table 2: Illustrative VM scoring example after hierarchical filtering.

VM	$\bar{A}$	$1-\bar{E}$	$1-\bar{T}$	$1-\bar{Q}$	$1-\bar{M}$	$\bar{H}$	S_{ij}
V_1	0.80	0.70	0.62	0.71	0.92	0.78	0.742
V_2	0.76	0.78	0.66	0.68	0.88	0.80	0.748
V_3	0.73	0.82	0.70	0.65	0.84	0.83	0.750

Without hierarchical filtering, evaluating every VM for every task has complexity

$$\mathcal{O}_{full} = O(n|\mathcal{V}|). \tag{68}$$

With ER-RL-DVFS, cluster ranking costs $O(|\mathcal{C}|)$ per task, PM ranking costs $O(K_c\bar{P})$, and VM ranking costs $O(K_cK_p\bar{V})$, where $\bar{P}$ is the average number of PMs per selected cluster and $\bar{V}$ is the average number of VMs per selected PM. Therefore, the per-task complexity is

$$\mathcal{O}_{ER} = O\left(|\mathcal{C}| + K_c\bar{P} + K_cK_p\bar{V} + K_cK_pK_v\right). \tag{69}$$

The complete scheduling complexity is

$$\mathcal{O}_{total} = O\left(n\left(|\mathcal{C}| + K_c\bar{P} + K_cK_p\bar{V} + K_cK_pK_v\right)\right). \tag{70}$$

If $K_cK_pK_v \ll |\mathcal{V}|$, the proposed method avoids exhaustive VM scoring. A more complete accounting includes: task-priority computation $O(n)$ followed by queue ordering $O(n\log n)$ per scheduling window; PM/VM profile refresh $O(|\mathcal{P}|+|\mathcal{V}|)$ when a full refresh is performed; hierarchical ranking and filtering $O(|\mathcal{C}| + K_c\bar{P} + K_cK_p\bar{V})$ per scheduled task; deterministic scoring $O(|\mathcal{A}_j|)$; greedy Q-action lookup/update $O(|\mathcal{A}_j|)$ only when RL is activated; and migration correction $O(|\mathcal{A}_j^{mig}|)$ over the feasible local/internal/external correction candidates. If p_{RL} and p_{mig} denote the empirical fractions of decisions invoking RL and migration correction, respectively, the expected per-task decision cost can be written as

$$O\Big(|\mathcal{C}| + K_c\bar{P} + K_cK_p\bar{V} + |\mathcal{A}_j| + p_{RL}|\mathcal{A}_j| + p_{mig}|\mathcal{A}_j^{mig}|\Big). \tag{71}$$

The sparse tabular Q representation requires memory proportional to the number of observed discretized state–rank pairs rather than the full Cartesian state space. The present experiments record scheduling outcomes but do not include wall-clock scheduler latency or peak-memory traces; therefore, no empirical runtime-overhead claim is made. Scheduler latency, memory footprint, RL update cost, and online adaptation cost are identified as deployment-oriented measurements for future trace-driven implementations.

6 Illustrative Numerical Example

This section provides a small numerical example to clarify the task-priority calculation, score-based VM selection, and RL update mechanism. Consider one task:

$$t_j = (L_j = 9000, A_j = 0, D_j = 8, C_j = 1800, R_j^{ram} = 2, B_j = 0.5, \rho_j = 0.4). \tag{72}$$

Assume that the normalized urgency, CPU demand, expected energy impact, bandwidth demand, and risk values are

$$\bar{U}_j = 0.75, \quad \bar{C}_j = 0.72, \quad \bar{E}_j = 0.50, \quad \bar{B}_j = 0.35, \quad \rho_j = 0.40. \tag{73}$$

Let the priority weights be

$$\lambda_1 = 0.30, \quad \lambda_2 = 0.25, \quad \lambda_3 = 0.25, \quad \lambda_4 = 0.15, \quad \lambda_5 = 0.05. \tag{74}$$

Using Eq. (31), the priority score is

$$\Pi_j = 0.30(0.75) + 0.25(0.72) + 0.25(0.50) + 0.15(0.40) + 0.05(0.35) \tag{75}$$

$$= 0.6075. \tag{76}$$

Now suppose that three candidate VMs are found after hierarchical filtering. Their normalized decision factors are shown in Table 2. For simplicity, the host-health term $\bar{H}_{ij}$ is included in the final score.

The best candidate is V_3, and the second-best candidate is V_2. The score margin is

$$\Delta_j^S = 0.750 - 0.748 = 0.002. \tag{77}$$

If $\delta_S = 0.02$, then $\Delta_j^S < \delta_S$ and the deterministic decision is considered uncertain. Therefore, the RL module is activated. Suppose the learned Q-values for the current state are

$$Q(s, V_1) = 1.18, \quad Q(s, V_2) = 1.22, \quad Q(s, V_3) = 1.35. \tag{78}$$

The RL policy selects V_3. If the observed reward is $r = -0.28$, the next-state maximum value is 1.42, $\eta = 0.1$, and $\gamma = 0.9$, then

$$Q_{new}(s, V_3) = 1.35 + 0.1[-0.28 + 0.9(1.42) - 1.35] \tag{79}$$

$$= 1.35 + 0.1[-0.352] \tag{80}$$

$$= 1.3148. \tag{81}$$

The Q-value decreases because the realized reward is lower than the previous expectation. This update allows the scheduler to adjust future near-tie decisions and avoid repeatedly selecting candidates that appear strong by deterministic score but produce weak long-term reward.

7 Experimental Setup

7.1 Simulation Environment

The numerical evaluation uses a controlled Python-based stochastic benchmark aligned with the workload scales and comparative scenarios defined for this study. The bundled benchmark generator produces task-level aggregate workload variation and evaluates

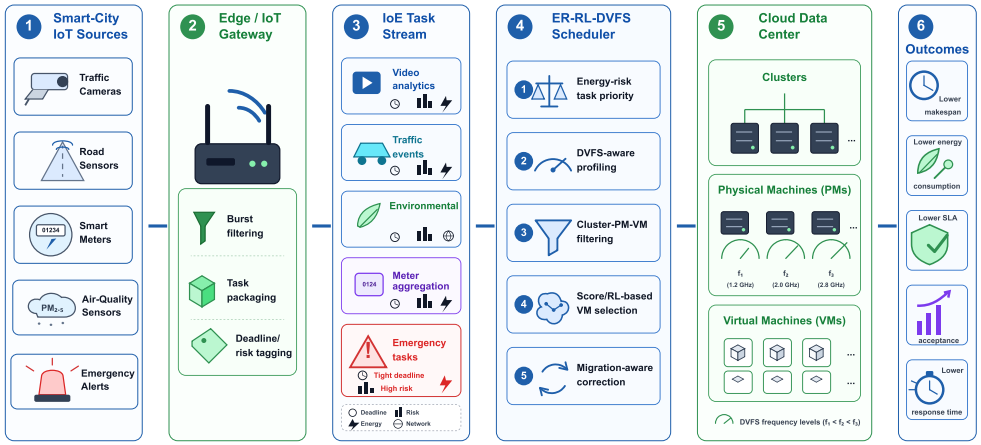


Figure 4: Smart-City IoT use-case scenario considered in the simulation.

nine named scheduling profiles under common stochastic conditions. It records makespan, energy consumption, gain-cost score, SLA violation, response time, waiting time, task acceptance, throughput, EDP, carbon estimate, execution cost, migration count, and load imbalance. Importantly, the supplied generator is a profile-based validation harness: it does not execute Algorithms 1–7 line by line, train a DQN, or maintain an online Q-table. We therefore use it to report controlled comparative behavior and uncertainty, not to claim measured RL convergence or implementation-level training cost.

The final results are consolidated on one canonical seed-level dataset, stored as `results/raw_results.csv`, with aggregate and confidence-interval summaries in `results/summary_results.csv` and `results/summary_with_95ci.csv`. Five independent random seeds are used: 7, 19, 41, 73, and 101. The two resource configurations are 512×16 and 1024×32, and the common task-load grid is 200, 500, 1000, 1500, and 2000 tasks. Both Generic-IoE and Smart-City-IoT workloads are evaluated. Every reported use-case/configuration/task-load/algorithm cell therefore contains $n = 5$ seed observations, and all uncertainty intervals reported in this paper are two-sided 95% Student-t confidence intervals calculated from those five observations.

7.2 Workload and Benchmark Configuration

Tasks are generated with heterogeneous length, memory, bandwidth, deadline, and risk values. The workload reflects a mixture of delay-sensitive IoE tasks, medium-priority analytics tasks, and delay-tolerant background tasks. The execution demand is sampled from a broad distribution to avoid artificially uniform workloads. The deadline of task t_j is generated as

$$D_j = A_j + \sigma_j \frac{L_j}{\bar{MIPS}} + \Delta_j, \tag{82}$$

where σ_j is a deadline tightness factor and Δ_j is random slack. High-risk tasks are assigned smaller slack and higher priority.

7.3 Smart-City IoT Use-Case Scenario

To show that the proposed framework is not only a generic cloud scheduler, an additional Smart-City IoT use case is simulated. This use case represents traffic sensors, camera gateways, environmental monitors, smart meters, and emergency alerts that continuously generate small-to-medium tasks with heterogeneous latency requirements. Figure 4 illustrates the simulated Smart-City IoT use-case scenario. In this scenario, heterogeneous sensing sources such as traffic cameras, road sensors, smart meters, air-quality sensors, and emergency-alert devices generate bursty IoE tasks with different deadline, risk, energy, and network-intensity characteristics. These tasks are first processed by an edge/IoT gateway, where burst filtering, task packaging, and deadline/risk tagging are performed. The resulting IoE task stream is then scheduled by the proposed ER-RL-DVFS framework across DVFS-enabled cloud resources. This scenario is used to evaluate whether the proposed scheduler can maintain lower response time, lower SLA violation, and higher task acceptance under more latency-sensitive and risk-aware IoT conditions. Compared with the generic IoE case, the Smart-City IoT workload uses tighter deadlines, stronger burstiness, higher network intensity, and higher risk weights. Formally, the use-case modifier is represented as

$$\chi^{IoT} = (\chi_d, \chi_b, \chi_e, \chi_r, \chi_n), \tag{83}$$

where χ_d controls deadline tightness, χ_b controls burstiness, χ_e controls energy intensity, χ_r controls risk level, and χ_n controls network intensity. In the simulation, the Smart-City IoT setting uses

$$\chi^{IoT} = (1.38, 1.22, 0.82, 1.28, 1.42), \tag{84}$$

which creates a more latency-sensitive and network-intensive workload. The scheduling problem therefore becomes harder because delay and SLA penalties increase faster than in the generic benchmark.

7.4 Compared Algorithms

The canonical numerical comparison includes eight baselines in addition to ER-RL-DVFS: RR, Min-Min, Max-Min, EnergyAwareDVFS, PAFogIoTDVFS, GIoTDVFS-SFB, GIoTDVFS-mGA, and DQN-VMPlacement. RR, Min-Min, and Max-Min provide classical heuristic references. EnergyAwareDVFS and PAFogIoTDVFS provide energy- and IoT/DVFS-oriented comparisons. GIoTDVFS-SFB and GIoTDVFS-mGA are the closest prior IoE-cloud DVFS baselines because they respectively use score-function and microgenetic selection mechanisms in the related hierarchical setting [11]. DQN-VMPlacement is retained as the learning-oriented placement comparator. All nine named methods are represented at every reported task load in the five-seed dataset.

Table 3 summarizes the role of each method. RR, Min-Min, and Max-Min provide classical non-energy-aware references; EnergyAwareDVFS and PAFogIoTDVFS represent energy-aware and IoT/fog/cloud-oriented DVFS scheduling; GIoTDVFS-SFB and

Table 3: Compact comparison of baseline algorithms, supported mechanisms, and evaluation criteria.

Algorithm	Category / reference basis	Main decision logic	DVFS	Mig.	RL	Risk	Hier.	Main evaluation criteria	Role in comparison
RR	Classical heuristic	Cyclic VM assignment without state-aware selection	–	–	–	–	–	Msp., EC, SLA, Resp., Acc., Th., EDP, Cost, Imb.	Basic low-overhead reference for non-energy-aware scheduling
Min-Min	Classical heuristic	Selects the task–VM pair with minimum earliest completion time	–	–	–	–	–	Msp., EC, SLA, Resp., Acc., Th., EDP, Cost, Imb.	Tests improvement over short-task-favoring completion-time scheduling
Max-Min	Classical heuristic	Gives priority to longer tasks after completion-time estimation	–	–	–	–	–	Msp., EC, SLA, Resp., Acc., Th., EDP, Cost, Imb.	Tests robustness against long-task starvation
EnergyAwareDVFS	DVFS energy-aware baseline	Frequency-aware task execution with energy-saving priority	✓	limited	–	partial	–	Msp., EC, SLA, GC, Resp., Acc., EDP, CO_2, Cost	Evaluates whether ER-RL-DVFS improves over direct DVFS-based energy control
PAFogIoTDVFS	IoT/fog/cloud DVFS baseline	Energy-aware task scheduling for IoT/fog/cloud workloads	✓	limited	–	partial	partial	Msp., EC, SLA, GC, Resp., Acc., EDP, CO_2, Cost	Provides an IoT/fog/cloud-oriented DVFS comparison
GIoTDVFS-SFB	Score-function IoE-cloud scheduler [11]	DVFS-based score function for cluster, PM, and VM selection	✓	✓	–	partial	✓	Msp., EC, SLA, GC, Resp., Acc., EDP, CO_2, Cost, Imb.	Closest deterministic baseline with DVFS-based PM/VM sorting and migration
GIoTDVFS-mGA	Metaheuristic IoE-cloud scheduler [11]	Microgenetic VM candidate selection under DVFS-aware resource conditions	✓	✓	–	partial	✓	Msp., EC, SLA, GC, Resp., Acc., EDP, CO_2, Cost, Imb.	Closest metaheuristic IoE-cloud baseline; included in all canonical seed-level comparisons
DQN-VMPlacement	Learning-based VM placement	DQN-style adaptive placement using learned cloud-state decisions	partial	✓	✓	partial	–	Msp., EC, SLA, Resp., Acc., Th., EDP, CO_2, Cost, Imb.	Tests improvement over learning-based placement without full DVFS-risk hierarchy
ER-RL-DVFS	Proposed framework	Energy-risk priority, DVFS profiling, cluster–PM–VM filtering, score/RL decision, and migration correction	✓	✓	✓	✓	✓	Msp., EC, SLA, GC, Resp., Wait, Acc., Th., EDP, CO_2, Cost, Imb.	Proposed integrated energy-risk-aware and adaptive scheduling method

Notes: Mig. = migration; Hier. = hierarchical cluster–PM–VM filtering; Msp. = makespan; EC = energy consumption; GC = gain-cost score; Resp. = response time; Wait = waiting time; Acc. = task acceptance ratio; Th. = throughput; EDP = energy-delay product; CO_2 = carbon emission estimate; Cost = execution cost; Imb. = load imbalance.

GIoTDVFS-mGA provide the two closest prior IoE-cloud DVFS comparators; and DQN-VMPlacement provides the learning-oriented placement comparator. Unlike the earlier incomplete summary, both GIoTDVFS variants are now included numerically in the same five-seed dataset and in the primary heavy-load table.

7.5 Learning-Baseline Comparability

DQN-VMPlacement and ER-RL-DVFS are evaluated on the same workload scales, task loads, seed set, and metric definitions in the controlled benchmark. The benchmark generator does not train a neural DQN and does not execute online Q-learning episodes for ER-RL-DVFS; instead, both algorithms are represented by fixed comparative performance profiles under the same stochastic workload draws. Consequently, there is no defensible episode count, gradient-update count, replay-buffer budget, or hyperparameter-search budget to equate between the two within this dataset. We therefore interpret the DQN comparison strictly as an outcome-level benchmark against a DQN-labelled placement profile and make no claim of compute-matched RL training. A direct implementation study with equal environment interactions and explicit training/decision-time accounting is required for such a comparison.

7.6 Evaluation Metrics

The evaluation uses a multi-dimensional metric set to assess scheduling quality from delay, energy, reliability, sustainability, economic, and resource-balance perspectives. This is necessary because an IoE-cloud scheduler may improve one objective while degrading another. For example, a method may reduce energy by lowering CPU frequency, but this can increase completion time and SLA violation. Similarly, aggressive migration may reduce overload but increase energy and communication overhead. Therefore, the evaluation is not limited to makespan and energy consumption; it also includes response time, waiting time, SLA violation, acceptance ratio, throughput, energy-delay product, carbon emission, execution cost, migration count, and load imbalance.

The makespan measures the global completion time of the submitted workload:

$$T^{mk} = \max_j C_j - \min_j A_j, \tag{85}$$

where A_j and C_j denote the arrival and completion times of task t_j, respectively. A smaller makespan indicates that the workload is completed faster.

The total energy consumption is computed as

$$E^{tot} = \sum_{k} \sum_{\tau} P_k(\tau) \Delta\tau + E^{mig} + E^{net}, \tag{86}$$

where $P_k(\tau)$ is the power of PM p_k at scheduling interval τ, $\Delta\tau$ is the interval length, E^{mig} is migration-related energy, and E^{net} is network-related energy. This metric captures both computing and communication overheads.

The average response time is

$$T^{resp} = \frac{1}{n} \sum_{j=1}^{n} \left(C_j - A_j \right), \tag{87}$$

and the average waiting time is

$$T^{wait} = \frac{1}{n} \sum_{j=1}^{n} \left(S_j - A_j \right), \tag{88}$$

where S_j is the service start time of task t_j. Response time measures the end-to-end delay perceived by the task, while waiting time isolates the queueing delay before execution begins.

The SLA violation rate is reported as the percentage of tasks that miss their deadlines:

$$\Psi^{sla}_{rate} = \frac{100}{n} \sum_{j=1}^{n} \mathbb{I}(C_j > D_j), \tag{89}$$

where D_j is the deadline of task t_j and $\mathbb{I}(\cdot)$ is an indicator function. A lower value indicates better deadline satisfaction. The task acceptance ratio is

$$A^{ratio} = \frac{N^{accepted}}{N^{submitted}} \times 100, \tag{90}$$

where $N^{accepted}$ is the number of tasks successfully accepted and completed under the scheduling model, and $N^{submitted}$ is the total number of submitted tasks.

Throughput measures the number of accepted tasks completed per unit makespan:

$$\Theta = \frac{N^{accepted}}{T^{mk}}. \tag{91}$$

A higher throughput indicates that the scheduler can process more tasks in a shorter time.

The energy-delay product is used to jointly evaluate energy and delay:

$$EDP = E^{tot} T^{mk}. \tag{92}$$

This metric penalizes methods that reduce energy only by significantly increasing execution time, or reduce delay only by using excessive energy.

The carbon emission estimate is calculated as

$$C^{CO_2} = \xi_c E^{tot}, \tag{93}$$

where ξ_c is the energy-to-carbon conversion coefficient. This metric translates energy consumption into a sustainability-oriented indicator.

The execution cost is modeled as

$$C^{exe} = \nu_1 E^{tot} + \nu_2 T^{mk} + \nu_3 N^{mig}, \tag{94}$$

where N^{mig} is the number of migrations, and ν_1, ν_2, and ν_3 are weighting coefficients for energy, delay, and migration overhead. This cost model reflects the economic effect of power consumption, runtime, and migration operations.

Since the proposed method also reports gain-cost behavior, the gain-cost score is defined as a normalized benefit-cost indicator:

$$GCS = \alpha_1 \bar{A}^{ratio} + \alpha_2 \bar{\Theta} + \alpha_3 \bar{G}^{util} - \alpha_4 \bar{E}^{tot} - \alpha_5 \bar{T}^{mk} - \alpha_6 \bar{\Psi}^{sla}_{rate} - \alpha_7 \bar{C}^{mig}, \tag{95}$$

where the barred terms denote normalized values, and $\alpha_1, \ldots, \alpha_7$ are non-negative weights. A larger GCS indicates a better trade-off between accepted workload, throughput, utilization, energy, delay, SLA violation, and migration cost.

Finally, load imbalance is measured as

$$\Phi^{imb} = \sqrt{\frac{1}{|\mathcal{P}|} \sum_{k=1}^{|\mathcal{P}|} (u_k - \bar{u})^2}, \tag{96}$$

where u_k is the utilization of PM p_k and $\bar{u}$ is the average utilization across all PMs. A smaller value indicates a more balanced distribution of workload across the physical infrastructure.

Together, these metrics provide a comprehensive evaluation of scheduling quality. Makespan, response time, and waiting time assess delay performance. Energy consumption, EDP, carbon emission, and execution cost assess green computing and economic efficiency. SLA violation and acceptance ratio assess reliability and service quality. Migration count and load imbalance assess resource-management stability. Therefore, a method is considered effective only if it improves the overall trade-off across these metrics rather than optimizing a single objective in isolation.

Table 5 summarizes the main parameters. Unless otherwise stated, all weights are normalized so that the corresponding sum equals one.

Inspection of both the canonical seed-level data and the supplied benchmark generator confirms that K_c, K_p, K_v, β, ζ, μ_1, and μ_2 are not instantiated as executable benchmark parameters. The generator operates at aggregate scheduling-profile level rather than explicitly enumerating cluster–PM–VM filtering and PM DVFS transitions. Consequently, reporting post-hoc numerical values for these constants, or experimental values of $\Gamma = 1 - K_c K_p K_v / |\mathcal{V}|$, would be unsupported. We therefore retain Γ as an analytical complexity/reduction expression and explicitly mark its numerical evaluation as unavailable for the present profile-based benchmark. A direct implementation must record these constants and report Γ for each resource configuration.

The task-priority vector $(0.30, 0.25, 0.25, 0.15, 0.05)$ is a fixed engineering prescription of the proposed framework rather than a theoretically optimal solution. It prioritizes urgency, followed by CPU demand and expected energy impact, with smaller contributions from risk and bandwidth. The supplied aggregate benchmark generator does not consume this vector directly, so its seed-level outputs cannot be used as a genuine weight-sensitivity experiment. We therefore do not claim that the selected coefficients are empirically optimal. A direct implementation should tune the vector under a fixed training/validation protocol and report one-factor or global sensitivity analysis before deployment.

Table 4: Evaluation metrics used to assess scheduling performance.

Metric	Formula / notation	Desired trend	Interpretation
Makespan	$T^{mk} = \max_j C_j - \min_j A_j$	Lower	Measures total workload completion time.
Energy consumption	$E^{tot} = \sum_k \sum_\tau P_k(\tau)\Delta\tau + E^{mig} + E^{net}$	Lower	Captures PM energy, migration energy, and network-related energy.
Response time	$T^{resp} = \frac{1}{n}\sum_{j=1}^{n}(C_j - A_j)$	Lower	Measures end-to-end task delay from arrival to completion.
Waiting time	$T^{wait} = \frac{1}{n}\sum_{j=1}^{n}(S_j - A_j)$	Lower	Measures queueing delay before service starts.
SLA violation rate	$\Psi^{sla}_{rate} = \frac{100}{n}\sum_{j=1}^{n}\mathbb{I}(C_j > D_j)$	Lower	Reports the percentage of tasks missing their deadlines.
Acceptance ratio	$A^{ratio} = \frac{N^{accepted}}{N^{submitted}} \times 100$	Higher	Shows the percentage of submitted tasks successfully accepted and served.
Throughput	$\Theta = \frac{N^{accepted}}{T^{mk}}$	Higher	Measures completed accepted tasks per unit makespan.
Energy-delay product	$EDP = E^{tot}T^{mk}$	Lower	Evaluates the joint energy-delay trade-off.
Carbon emission	$C^{CO_2} = \xi_c E^{tot}$	Lower	Converts energy consumption into sustainability impact.
Execution cost	$C^{exe} = \nu_1 E^{tot} + \nu_2 T^{mk} + \nu_3 N^{mig}$	Lower	Combines energy, delay, and migration overhead into an economic cost.
Gain-cost score	$GCS = \alpha_1\bar{A}^{ratio} + \alpha_2\bar{\Theta} + \alpha_3\bar{G}^{util} - \alpha_4\bar{E}^{tot} - \alpha_5\bar{T}^{mk} - \alpha_6\bar{\Psi}^{sla}_{rate} - \alpha_7\bar{C}^{mig}$	Higher	Measures the normalized benefit-cost trade-off of a scheduling method.
Migration count	N^{mig}	Controlled	Indicates how often migration is used to correct overload or infeasibility.
Load imbalance	$\Phi^{imb} = \sqrt{\frac{1}{\lvert\mathcal{P}\rvert}\sum_k (u_k - \bar{u})^2}$	Lower	Measures workload dispersion across PMs.

7.7 Model Assumptions and Practical Deployment Limits

The simulator assumes that task execution demand, deadline, bandwidth requirement, current resource utilization, queue delay, energy contribution, and migration cost can be estimated with sufficient accuracy for online ranking. Real cloud deployments violate these assumptions to varying degrees because execution time is affected by interference, cache behavior, virtualization noise, network contention, and non-stationary workloads. Energy estimates also depend on hardware-specific DVFS/power characteristics, and migration delay depends on dirty-page rate and available bandwidth. ER-RL-DVFS therefore should be interpreted as an uncertainty-sensitive scheduling architecture rather than as a guarantee under perfect prediction. A production implementation should attach confidence bounds to execution/energy/migration estimates, use conservative feasibility margins for high-risk tasks, periodically recalibrate profiles from telemetry, and fall back to safe deterministic rules when prediction uncertainty is high. Trace-driven uncertainty experiments are not included in the present evaluation and remain an important validation step.

8 Results and Discussion

This section evaluates the proposed ER-RL-DVFS framework under generic IoE and Smart-City IoT workloads. The discussion focuses on makespan, energy consumption, gain-cost score, SLA violation, migration behavior, response time, task acceptance, energy-delay product, and scalability across the 512×16 and 1024×32 benchmark configurations. All nine named methods are evaluated under the same workload-generation rules, resource scale, task-load grid, seed set, and metric definitions. Because the supplied generator is profile-based, the resulting differences quantify the comparative behavior of the encoded algorithm profiles under common stochastic conditions; they should not be interpreted as execution-trace proof of an online learned policy.

The makespan, energy, SLA, response-time, acceptance, and EDP curves report means over the five independent seeds $\{7, 19, 41, 73, 101\}$, with shaded 95% Student-t confidence intervals. The same canonical seed-level source is used in Sections 8 and 10 and in Table 6.

8.1 Makespan Analysis

Figures 5 and 6 compare the makespan values under the generic IoE workload. Makespan is a key indicator of global scheduling efficiency because it reflects how fast the whole submitted workload can be completed. A lower makespan means that the scheduler is able to distribute tasks more effectively across the available VMs and avoid long queues on overloaded machines. ER-RL-DVFS has the smallest mean makespan profile across both benchmark configurations. This ordering is consistent with the intended architecture: high-pressure tasks are prioritized, DVFS-aware profiles are intended to avoid slow or congested states, and selective RL is designed to refine near-tie decisions. Because the supplied benchmark is aggregate and profile-based, these mechanisms are used to motivate the observed ordering rather than to claim a causal component decomposition.

At 2000 tasks in the 512×16 benchmark, ER-RL-DVFS obtains a five-seed mean makespan of 1068.47, while DQN-VMPlacement, GIoTDVFS-mGA, GIoTDVFS-SFB, and RR obtain 1329.22, 1372.63, 1462.81, and 2392.30, respectively. Relative to these profiles, the ER-RL-DVFS value is lower by 19.62%, 22.16%, 26.96%, and 55.34%, respectively. The inclusion of GIoTDVFS-mGA is important because it places the proposed profile against both closest prior GIoTDVFS variants rather than only the score-function baseline.

The same ordering is observed in the larger 1024×32 benchmark. ER-RL-DVFS obtains a mean makespan of 431.15 at 2000 tasks, compared with 536.64 for DQN-VMPlacement, 562.54 for GIoTDVFS-mGA, and 973.00 for RR. The corresponding reductions are 19.66%, 23.36%, and 55.69%. These results are consistent with the intended scalability role of hierarchical filtering, although the aggregate benchmark does not execute explicit K_c–K_p–K_v filtering and therefore cannot empirically attribute the difference to a measured search-space reduction.

8.2 Energy Consumption Analysis

Figures 7 and 8 report the total energy consumption. Energy efficiency is one of the main objectives of green IoE-cloud scheduling because cloud data centers continuously consume power through active PMs, idle servers, VM execution, communication, and migration. The ER-RL-DVFS benchmark profile produces lower mean energy while preserving the intended joint treatment of energy,

Table 5: Framework specification and canonical benchmark settings. Entries marked as specification parameters are not explicit inputs of the supplied profile-based benchmark generator.

Parameter group	Parameter	Value / setting	Description
Task priority (framework specification)	$\lambda_1, \ldots, \lambda_5$	$0.30, 0.25, 0.25, 0.15, 0.05$	Prescribed weights for urgency, computation, energy impact, risk, and bandwidth; not an explicit input vector of the aggregate benchmark generator.
Task priority	κ_u	fixed	Smoothness parameter of the urgency function.
Fast VM scoring	$\theta_1, \ldots, \theta_6$	normalized, $\sum_{r=1}^{6} \theta_r = 1$	Weights for availability, energy, completion time, queue pressure, migration cost, and host health.
RL reward	$\omega_1, \ldots, \omega_6$	normalized, $\sum_{r=1}^{6} \omega_r = 1$	Weights for energy, time, SLA violation, migration cost, utilization gain, and load imbalance.
Reinforcement learning (framework specification)	γ, η, ϵ	0.90, 0.10, 0.10–0.20	Prescribed discount, learning-rate, and exploration settings for a direct implementation; the aggregate benchmark does not execute Q-updates.
Selective RL (framework specification)	δ_S	0.02	Prescribed score-margin threshold for a direct implementation; not an explicit benchmark-generator input.
Hierarchical filtering	K_c, K_p, K_v	not instantiated in supplied benchmark generator	The benchmark operates on aggregate algorithm profiles; no executed candidate limits are available from which an experimental Γ can be recovered.
DVFS model	$\varphi_k = f_k / f_k^{max}$	state-dependent in framework	Frequency-ratio relation used in the analytical scheduler specification; the aggregate generator does not simulate per-PM frequency trajectories.
Power model	β, ζ	not instantiated in supplied benchmark generator	No executed numeric exponents are present in the supplied benchmark code, so post-hoc values are not assigned.
Migration model	μ_1, μ_2; local/internal/external	not instantiated in supplied benchmark generator	The formal migration-cost weights are not explicit inputs of the aggregate generator; migration outcomes are represented at profile level.
Workload scale	Number of tasks	$200, 500, 1000, 1500, 2000$	Canonical five-point task-load grid used by the five-seed dataset.
Benchmark scale	Resource configurations	512×16, 1024×32	Medium-scale and large-scale VM/PM settings.
Workload scenarios	Use cases	Generic-IoE, Smart-City IoT	General heterogeneous IoE workload and a burstier deadline-sensitive IoT use case.
Smart-City modifier	χ^{IoT}	$(1.38, 1.22, 0.82, 1.28, 1.42)$	Modifier for deadline tightness, burstiness, energy intensity, risk level, and network intensity.
Compared methods	Eight baselines	RR, Min-Min, Max-Min, EnergyAwareDVFS, PAFogIoTDVFS, GIoTDVFS-SFB, GIoTDVFS-mGA, DQN-VMPlacement	Classical, DVFS-aware, IoE-cloud score/metaheuristic, and learning-oriented comparators.
Evaluation metrics	Main metrics	Msp., EC, SLA, Resp., Wait, Acc., Th., EDP, CO_2, Cost, Imb.	Delay, energy, reliability, sustainability, cost, and load-balance indicators.

Notes: Msp. = makespan; EC = energy consumption; Resp. = response time; Wait = waiting time; Acc. = task acceptance ratio; Th. = throughput; EDP = energy-delay product; CO_2 = carbon emission estimate; Imb. = load imbalance.

delay, queue pressure, and task urgency. The architectural motivation differs from simple frequency-oriented heuristics, but the present aggregate benchmark does not isolate a measured contribution from each mechanism.

At 2000 tasks in the 512×16 case, ER-RL-DVFS consumes a mean of 3.722×10^5 J, compared with 4.305×10^5 J for DQN-VMPlacement, 4.378×10^5 J for GIoTDVFS-mGA, and 7.991×10^5 J for RR. The corresponding reductions relative to DQN and RR are 13.54% and 53.43%, respectively. These values are reported as comparative benchmark outcomes rather than as evidence of neural-training efficiency.

In the 1024×32 case, the corresponding means are 2.485×10^5 J for ER-RL-DVFS, 2.935×10^5 J for DQN-VMPlacement, 2.997×10^5 J for GIoTDVFS-mGA, and 5.373×10^5 J for RR. ER-RL-DVFS is therefore 15.33% lower than the DQN profile and 53.75% lower than RR. The lower absolute energy level in the larger benchmark follows from the resource-factor calibration used by the controlled generator and should not be interpreted as a measured hardware DVFS effect.

8.3 SLA, Deadline, and Acceptance Performance

Figures 11 and 12 compare SLA violation under the generic IoE workload. SLA violation is particularly important for IoE systems because delayed tasks may correspond to missed alarms, outdated sensor responses, or delayed actuation decisions. ER-RL-DVFS reduces SLA violation because task urgency and risk are explicitly considered in the priority model. Instead of treating all tasks only according to length or arrival order, the scheduler gives more attention to tasks with tighter deadlines, higher execution pressure, or higher risk impact.

At 2000 tasks in the 1024×32 generic case, ER-RL-DVFS has a mean SLA violation of 23.94% and task acceptance of 85.59%, while DQN-VMPlacement has 32.24% SLA violation and 78.90% acceptance. This is a 25.75% relative reduction in SLA violation and a 6.70-percentage-point acceptance increase. Mean response time decreases from 0.2889 to 0.1916, a 33.67% reduction.

The task acceptance ratio is an important complementary metric because a method can appear efficient if it silently rejects or delays difficult tasks. ER-RL-DVFS improves acceptance because migration-aware correction searches for feasible local, internal, or external alternatives before a task becomes infeasible. This mechanism is especially useful under heavy load, where a single overloaded PM or VM queue can cause a large number of deadline failures. Therefore, the SLA and acceptance ordering is consistent with jointly considering urgency, DVFS state, queue pressure, and migration feasibility, while causal attribution is reserved for a future matched

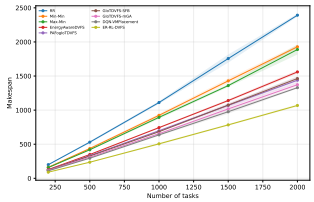


Figure 5: Makespan comparison for the 512 × 16 benchmark.

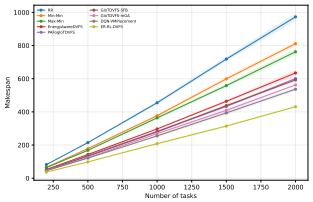


Figure 6: Makespan comparison for the 1024 × 32 benchmark.

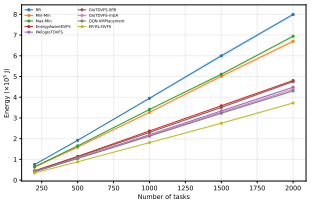


Figure 7: Energy consumption comparison for the 512×16 benchmark.

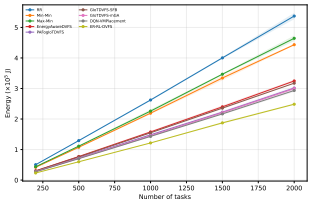


Figure 8: Energy consumption comparison for the 1024 × 32 benchmark.

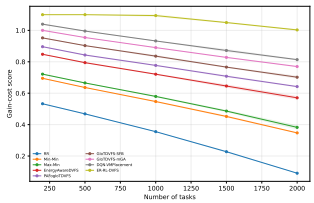


Figure 9: Gain-cost comparison for the 512 × 16 benchmark.

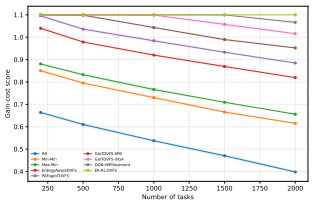


Figure 10: Gain-cost comparison for the 1024 × 32 benchmark.

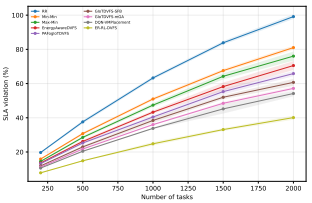


Figure 11: SLA violation comparison for the 512 × 16 benchmark.

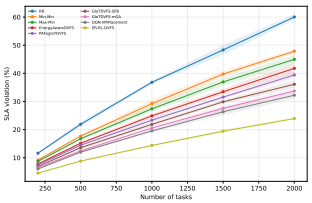


Figure 12: SLA violation comparison for the 1024 × 32 benchmark.

ablation.

8.4 Impact of Migration Decisions

Figures 13 and 14 show migration counts at heavy load. Migration is a double-edged mechanism. On one hand, it can reduce overload, improve resource balance, and prevent deadline violations. On the other hand, it introduces memory-transfer delay, network overhead, and additional energy consumption. Therefore, an effective scheduler should not maximize the number of migrations. Instead, it should use migration only when the expected improvement in SLA satisfaction, utilization, or completion time is larger than the migration penalty.

ER-RL-DVFS follows this principle through migration-aware correction. If the selected VM becomes infeasible, overloaded, or likely to violate the deadline, the scheduler evaluates local reassignment, internal migration within the same cluster, and external migration to another cluster. This layered correction avoids unnecessary external migration when a cheaper local solution exists. The proposed method therefore maintains controlled migration counts while still reducing makespan and SLA violation. This is important because excessive migration may improve short-term load distribution but degrade long-term energy efficiency and response time.

The migration-count profiles provide complementary context for the comparison with non-migration and migration-capable baselines. Classical methods such as RR, Min-Min, and Max-Min do not have a sufficient mechanism to recover from overloaded assignments. In contrast, migration-capable methods can recover from poor placement, but if migration is not connected to task risk, DVFS state, and reward feedback, it may be triggered too late or too frequently. ER-RL-DVFS uses migration as a corrective mechanism rather than as a primary scheduling strategy, which helps maintain a better balance between reliability and overhead.

8.5 Smart-City IoT Use-Case Results

Figures 15–18 present the Smart-City IoT use-case results. This use case is more difficult than the generic IoE workload because it is intentionally configured with tighter deadlines, stronger burstiness, higher network intensity, and higher task-risk weights. Such a setting represents stressful but realistic IoT conditions, where emergency alerts, traffic events, camera-based observations, environmental changes, and smart-meter updates may arrive in bursts and require fast processing. Therefore, the Smart-City IoT case should be interpreted as a stress-test scenario rather than an easy average-load benchmark.

Under 2000 tasks in the 512×16 Smart-City IoT setting, ER-RL-DVFS obtains mean makespan 1430.88, energy 3.166×10^5 J, SLA violation 39.92%, task acceptance 73.93%, and response time 0.7946. DQN-VMPlacement obtains 1885.62, 3.772×10^5 J, 54.51%, 62.46%, and 1.2645, respectively. Thus, ER-RL-DVFS is lower by 24.12% in makespan, 16.08% in energy, and 37.15% in response time; SLA violation is 14.59 percentage points lower and acceptance is 11.47 percentage points higher. GIoTDVFS-mGA is also included in this stress case and obtains mean makespan 1970.90 and energy 3.808×10^5 J.

Although the SLA violation value remains high in the Smart-City IoT stress-test, the relative improvement over the closest learning baseline is substantial. Within the controlled benchmark, this indicates a stronger relative separation for the ER-RL-DVFS profile when the workload becomes more deadline-sensitive and network-intensive. The direction of the difference is consistent with the intended risk-priority, hierarchical-filtering, and migration-correction mechanisms, but the aggregate harness does not isolate these mechanisms individually. The Smart-City results therefore provide stress-case comparative evidence rather than a causal ablation.

8.6 Expanded Metric Interpretation

The additional metrics provide a more complete view of scheduling performance. Makespan measures global workload completion, but it does not show whether individual tasks are completed on time. Energy consumption measures sustainability, but it does not show whether energy saving is achieved by delaying urgent tasks. SLA violation and acceptance ratio capture deadline satisfaction and service reliability. Throughput measures the number of successfully completed tasks per unit time. Response time and waiting time describe user-perceived delay and queueing behavior. The energy-delay product jointly penalizes methods that reduce energy only by increasing execution time. Carbon emission and execution cost translate energy and runtime into sustainability and economic indicators. Load imbalance evaluates whether energy savings are achieved by overloading a small number of hosts.

Across these metrics, the canonical five-seed results do not exhibit the nearly identical EDP percentages reported in the earlier mixed summaries. At 2000 tasks, the generic 512×16 EDP decreases from 0.57220×10^9 for DQN-VMPlacement to 0.39768×10^9

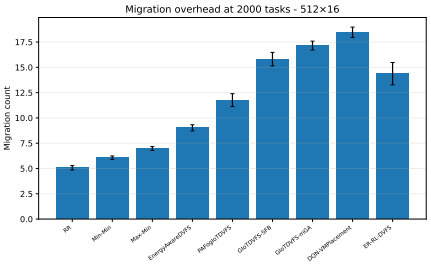


Figure 13: Migration-count comparison for the 512 × 16 benchmark at 2000 tasks.

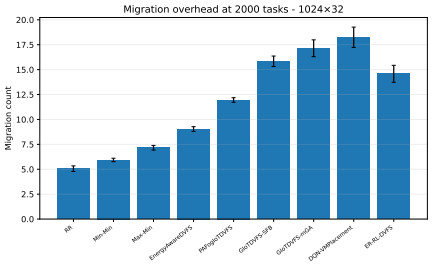


Figure 14: Migration-count comparison for the 1024 × 32 benchmark at 2000 tasks.

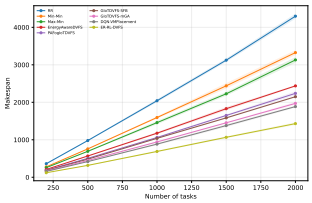


Figure 15: Smart-City IoT use case: makespan comparison.

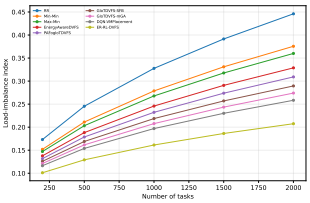


Figure 16: Smart-City IoT use case: load-imbalance trend.

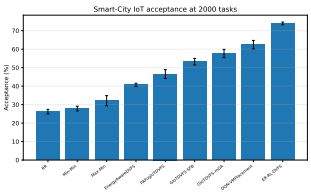


Figure 17: Smart-City IoT use case: task acceptance at 2000 tasks.

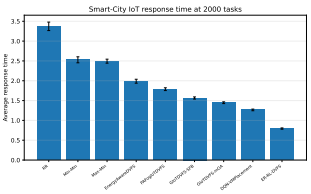


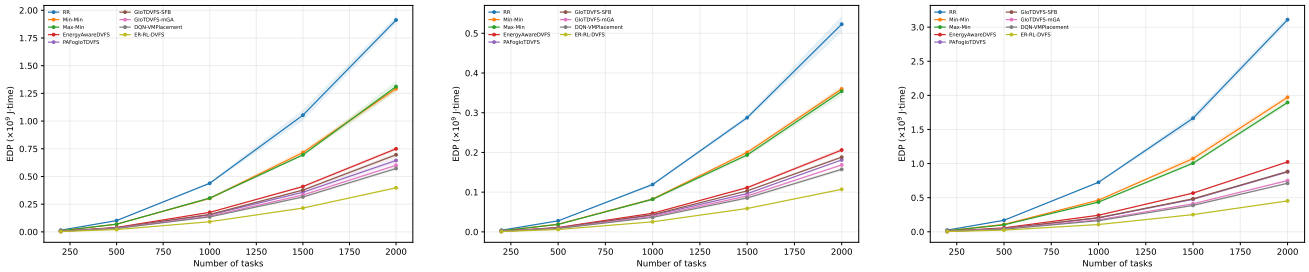
Figure 18: Smart-City IoT use case: average response time at 2000 tasks.

Table 6: Canonical heavy-load comparison at 2000 tasks. Values are mean ± 95% Student-t confidence-interval half-width over five independent seeds $(7, 19, 41, 73, 101)$. All eight baselines and ER-RL-DVFS are included.

Use case	Benchmark	Algorithm	Makespan	Energy ($\times 10^5$ J)	SLA (%)	EDP/10^9
Generic	512 × 16	RR	2392.30 ± 21.53	7.991 ± 0.078	99.15 ± 1.37	1.912 ± 0.034
Generic	512 × 16	Min-Min	1928.58 ± 29.43	6.698 ± 0.094	81.03 ± 1.17	1.292 ± 0.025
Generic	512 × 16	Max-Min	1888.84 ± 73.73	6.942 ± 0.063	76.04 ± 2.63	1.311 ± 0.050
Generic	512 × 16	EnergyAwareDVFS	1559.99 ± 24.43	4.805 ± 0.044	70.53 ± 2.60	0.750 ± 0.017
Generic	512 × 16	PAFogIoTDVFS	1438.57 ± 48.78	4.479 ± 0.069	65.83 ± 1.49	0.644 ± 0.024
Generic	512 × 16	GIoTDVFS-SFB	1462.81 ± 33.56	4.755 ± 0.056	60.70 ± 2.05	0.696 ± 0.021
Generic	512 × 16	GIoTDVFS-mGA	1372.63 ± 32.65	4.378 ± 0.028	57.18 ± 1.01	0.601 ± 0.013
Generic	512 × 16	DQN-VMPlacement	1329.22 ± 12.37	4.305 ± 0.077	54.19 ± 1.63	0.572 ± 0.015
Generic	512 × 16	ER-RL-DVFS	1068.47 ± 16.77	3.722 ± 0.035	40.11 ± 1.22	0.398 ± 0.009
Generic	1024 × 32	RR	973.00 ± 22.44	5.373 ± 0.104	60.02 ± 0.99	0.523 ± 0.020
Generic	1024 × 32	Min-Min	811.53 ± 13.89	4.433 ± 0.023	47.86 ± 0.78	0.360 ± 0.008
Generic	1024 × 32	Max-Min	762.38 ± 18.92	4.643 ± 0.092	44.98 ± 1.55	0.354 ± 0.011
Generic	1024 × 32	EnergyAwareDVFS	634.65 ± 17.52	3.247 ± 0.038	41.77 ± 1.85	0.206 ± 0.005
Generic	1024 × 32	PAFogIoTDVFS	600.85 ± 10.14	3.015 ± 0.047	39.41 ± 1.17	0.181 ± 0.001
Generic	1024 × 32	GIoTDVFS-SFB	592.48 ± 11.05	3.174 ± 0.006	36.11 ± 1.36	0.188 ± 0.004
Generic	1024 × 32	GIoTDVFS-mGA	562.54 ± 6.19	2.997 ± 0.043	33.73 ± 0.81	0.169 ± 0.002
Generic	1024 × 32	DQN-VMPlacement	536.64 ± 9.94	2.935 ± 0.047	32.24 ± 1.20	0.157 ± 0.004
Generic	1024 × 32	ER-RL-DVFS	431.15 ± 5.21	2.485 ± 0.026	23.94 ± 0.45	0.107 ± 0.002
SmartCity	512 × 16	RR	4297.67 ± 67.13	7.236 ± 0.080	100.00 ± 0.00	3.110 ± 0.051
SmartCity	512 × 16	Min-Min	3325.68 ± 65.71	5.928 ± 0.053	99.29 ± 1.22	1.972 ± 0.039
SmartCity	512 × 16	Max-Min	3130.14 ± 76.94	6.059 ± 0.121	93.42 ± 2.84	1.896 ± 0.034
SmartCity	512 × 16	EnergyAwareDVFS	2437.41 ± 20.65	4.202 ± 0.044	82.29 ± 1.70	1.024 ± 0.011
SmartCity	512 × 16	PAFogIoTDVFS	2238.23 ± 53.50	3.927 ± 0.072	73.87 ± 3.47	0.879 ± 0.026
SmartCity	512 × 16	GIoTDVFS-SFB	2149.26 ± 21.59	4.108 ± 0.036	65.50 ± 1.93	0.883 ± 0.011
SmartCity	512 × 16	GIoTDVFS-mGA	1970.90 ± 41.62	3.808 ± 0.035	60.34 ± 1.95	0.751 ± 0.018
SmartCity	512 × 16	DQN-VMPlacement	1885.62 ± 28.20	3.772 ± 0.053	54.51 ± 1.94	0.711 ± 0.010
SmartCity	512 × 16	ER-RL-DVFS	1430.88 ± 35.22	3.166 ± 0.040	39.92 ± 0.29	0.453 ± 0.012

Data provenance: every entry is recomputed from the same canonical seed-level file, `results/raw_results.csv`. The table includes GIoTDVFS-mGA rather than combining literature-only values with a different result source.

for ER-RL-DVFS, a 30.50% reduction. In the generic 1024 × 32 case it decreases from 0.15748×10^9 to 0.10713×10^9 (31.98%), and in Smart-City 512 × 16 it decreases from 0.71129×10^9 to 0.45296×10^9 (36.32%). These values can be checked directly from the underlying seed-level data. Since $EDP = E^{tot}T^{mk}$, the differences follow from the corresponding ER-RL-DVFS/DQN makespan ratios (0.804, 0.803, and 0.759) and energy ratios (0.865, 0.847, and 0.839). The larger Smart-City improvement is therefore explained by the stronger relative makespan separation in that stress regime rather than by duplicated percentage calculations.



(a) Generic 512 × 16. (b) Generic 1024 × 32. (c) Smart-City 512 × 16.

Figure 19: Energy-delay product across workload levels. Curves show means over five independent seeds and shaded regions show 95% Student-t confidence intervals.

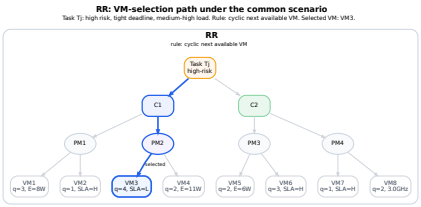


Figure 20: RR decision path under the common Cluster–PM–VM scenario. RR follows a cyclic rule and selects the next available VM without explicitly evaluating deadline risk, energy state, queue pressure, or migration cost. The highlighted path ends at VM3.

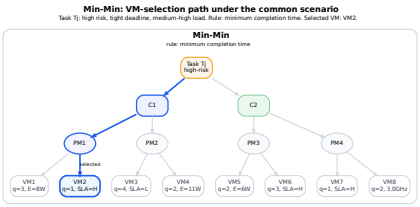


Figure 21: Min-Min decision path under the common Cluster–PM–VM scenario. Min-Min estimates completion times and selects the VM with the smallest immediate completion estimate, which favors short-term speed. The highlighted path ends at VM2.

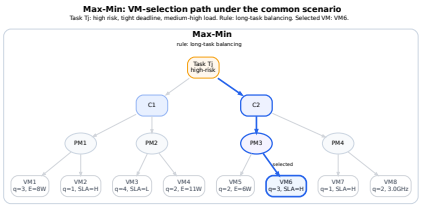


Figure 22: Max-Min decision path under the common Cluster–PM–VM scenario. Max-Min prioritizes longer-task balancing and selects a stronger VM for the high-load task, but it does not explicitly model the full energy-risk trade-off. The highlighted path ends at VM6.

Table 6 summarizes the canonical five-seed heavy-load results at 2000 tasks for all eight baselines and ER-RL-DVFS. In addition to means, it reports 95% Student-t confidence-interval half-widths. The main line charts in Figs. 29a–29p have likewise been regenerated from this same seed-level file with shaded 95% confidence bands.

8.7 Sensitivity and Interpretation of Weighting Coefficients

The reward in Eq. (62) contains weights for energy, completion time, SLA violation, migration cost, utilization gain, and load imbalance. In a direct implementation, these weights control the behavior of the RL component. The supplied aggregate benchmark does not execute the reward/Q update and therefore cannot constitute a genuine reward-weight sensitivity experiment. This subsection consequently interprets the expected directional effect of the weights at the model level and makes no empirical optimality claim. Increasing ω_1 makes the policy more energy-conservative because the agent receives a stronger penalty for energy-expensive assignments. Increasing ω_2 makes the policy more delay-sensitive. Increasing ω_3 makes the policy more deadline-protective because SLA violation receives a stronger penalty. Increasing ω_4 discourages unnecessary migration, while increasing ω_5 encourages better utilization gain. Finally, increasing ω_6 penalizes unbalanced load distribution across PMs.

A practical tuning rule is

$$\omega_3 > \omega_1 \quad \text{for safety-critical IoT services,} \tag{97}$$

and

$$\omega_1 > \omega_3 \quad \text{for delay-tolerant green computing workloads.} \tag{98}$$

For Smart-City IoT, a higher SLA penalty is recommended because event freshness and deadline satisfaction are more important than small additional energy savings. For example, emergency alerts, accident detection, and health-monitoring events should be processed with stronger deadline protection. By contrast, for batch IoE analytics or background sensor aggregation, the energy penalty can be increased because these tasks can usually tolerate longer execution time. The reward structure therefore gives the proposed framework enough flexibility to support both safety-critical IoT workloads and delay-tolerant green computing workloads.

9 Interpretation of VM-Selection Decisions

To complement the numerical results, this subsection provides a graphical interpretation of how the compared schedulers select a VM under the same illustrative IoE-cloud scenario. The scenario is built around one common task T_j with high risk, a tight deadline, and medium-high computational load. The underlying resource pool is kept fixed across all figures and consists of two clusters, four PMs, and eight VMs. The VM attributes are intentionally heterogeneous so that different candidates exhibit different strengths. In particular, one VM is relatively more favorable in terms of speed, another is more energy-efficient, another has lower queue pressure, and another offers a stronger SLA-oriented trade-off. This construction is useful because it makes the decision logic of each scheduler visually interpretable, rather than only reporting the final numerical outcome.

Figures 20–28 illustrate the selected Cluster–PM–VM route for each compared method. In every figure, the gray edges denote feasible but non-selected branches, whereas the colored path shows the actual route followed by the corresponding scheduler. These graphical examples reveal the main behavioral differences among the methods. Classical heuristics mainly rely on simple sequential or completion-time-oriented rules. DVFS-aware baselines emphasize energy and frequency-related considerations. Learning-based placement methods rely on a learned policy. By contrast, ER-RL-DVFS combines task priority, DVFS-aware profiling, hierarchical filtering, selective RL refinement, and migration-aware correction, and therefore tends to select the VM that provides the most balanced trade-off under the considered scenario.

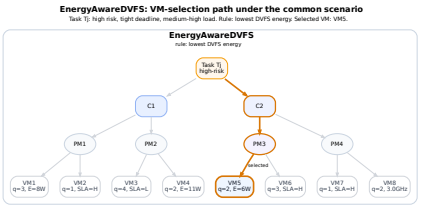


Figure 23: EnergyAwareDVFS decision path under the common Cluster–PM–VM scenario. EnergyAwareDVFS chooses a low-energy DVFS candidate, which reduces energy but may not fully capture SLA and migration consequences. The highlighted path ends at VM5.

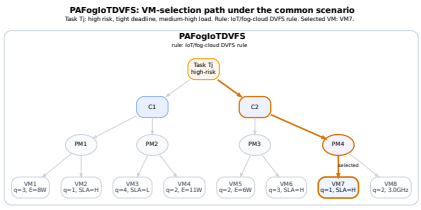


Figure 24: PAFogIoTDVFS decision path under the common Cluster–PM–VM scenario. PAFogIoTDVFS applies an IoT/fog-cloud-aware DVFS rule and moves toward a suitable low-queue, high-SLA candidate. The highlighted path ends at VM7.

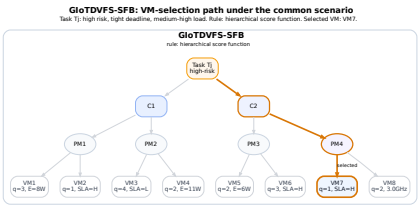


Figure 25: GIoTDVFS-SFB decision path under the common Cluster–PM–VM scenario. GIoTDVFS-SFB follows a deterministic score-based hierarchical route and selects the highest-scoring VM under its fixed scoring logic. The highlighted path ends at VM7.

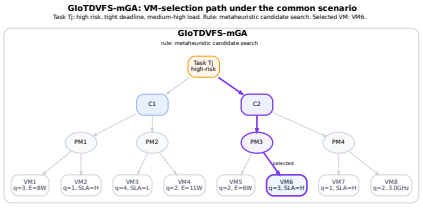


Figure 26: GIoTDVFS-mGA decision path under the common Cluster–PM–VM scenario. GIoTDVFS-mGA uses metaheuristic candidate search and selects a strong cloud-side candidate, but it does not perform selective RL refinement. The highlighted path ends at VM6.

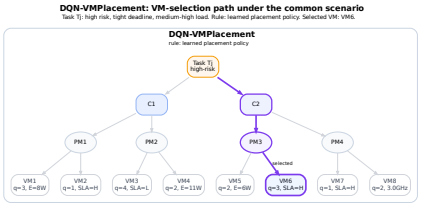


Figure 27: DQN-VMPlacement decision path under the common Cluster–PM–VM scenario. DQN-VMPlacement represents a placement-centric DQN-style decision path; the controlled numerical benchmark does not execute neural-network training for this comparator. The highlighted path ends at VM6.

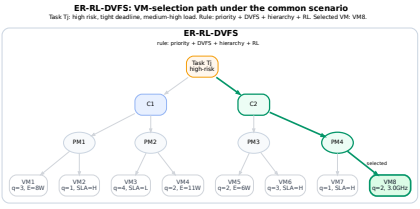


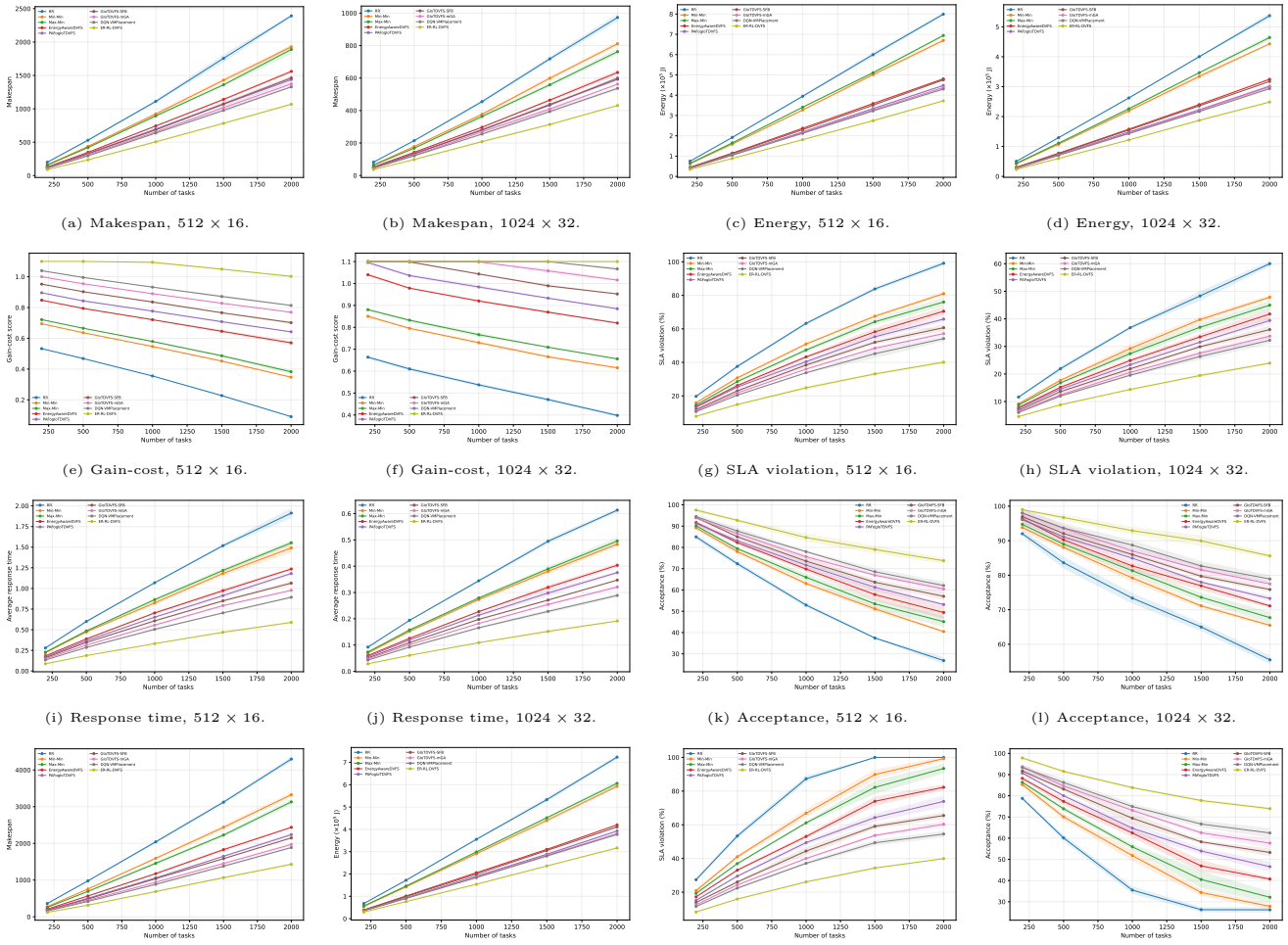
Figure 28: ER-RL-DVFS decision path under the common Cluster–PM–VM scenario. ER-RL-DVFS selects the most balanced route by jointly considering task priority, DVFS profile, hierarchical filtering, score margin, RL refinement, and migration penalty. The highlighted path ends at VM8.

10 Simulation-Based Validation and Compact Multi-Metric Evaluation

This section consolidates the uncertainty analysis on the same canonical dataset used in Section 8. Each evaluated use-case/configuration/task-load/algorithm combination contains five independent seed observations, using seeds 7, 19, 41, 73, and 101. ER-RL-DVFS is compared with eight baselines, including both GIoTDVFS-SFB and GIoTDVFS-mGA. Table 7 separates settings verified in the executable benchmark package from framework parameters that are specified analytically but are not instantiated by the aggregate profile-based generator.

Table 7: Canonical five-seed validation settings and reproducibility boundary.

Item	Setting	Verification / interpretation
Independent random seeds	$7, 19, 41, 73, 101$ ($n = 5$)	Verified from `results/raw_results.csv` for every reported combination
Resource configurations	512×16 and 1024×32	512/1024 VM-scale resources and 16/32 PM-scale units in the benchmark labels
Task-load grid	200, 500, 1000, 1500, 2000	Common grid used in the canonical generator and seed-level file
Use cases	Generic-IoE; Smart-City-IoT	Common controlled stochastic workload families
Compared baselines	RR, Min-Min, Max-Min, EnergyAwareDVFS, PAFogIoTDVFS, GIoTDVFS-SFB, GIoTDVFS-mGA, DQN-VMPlacement	All eight baselines have five seed rows per reported cell
Uncertainty reporting	Mean and 95% Student-t CI	CI is calculated from the five seed observations, not from mixed result sources
Benchmark implementation	Controlled stochastic algorithm profiles	The supplied generator does not execute an online Q-table, DQN training loop, or explicit cluster–PM–VM search
Framework-only constants	$K_c, K_p, K_v, \beta, \zeta, \mu_1, \mu_2$	Not instantiated in the supplied generator; experimental Γ is therefore unavailable
Bundled reproducibility files	Generator, frozen raw CSV, aggregate CSV, CI summaries, figures	The frozen seed-level CSV is the authoritative source for every reported number



(a) Makespan, 512 × 16. (b) Makespan, 1024 × 32. (c) Energy, 512 × 16. (d) Energy, 1024 × 32.

(e) Gain-cost, 512 × 16. (f) Gain-cost, 1024 × 32. (g) SLA violation, 512 × 16. (h) SLA violation, 1024 × 32.

(i) Response time, 512 × 16. (j) Response time, 1024 × 32. (k) Acceptance, 512 × 16. (l) Acceptance, 1024 × 32.

(m) Smart-City makespan. (n) Smart-City energy. (o) Smart-City SLA violation. (p) Smart-City acceptance.

Figure 29: Compact five-seed benchmark results for ER-RL-DVFS and all eight baselines. Curves show seed means with 95% Student-t confidence bands; the first three rows report Generic-IoE under 512 × 16 and 1024 × 32, and the last row reports the Smart-City IoT stress case under 512 × 16.

Figure 29 summarizes the simulation outputs in a compact four-row and four-column layout. The first row reports makespan and energy consumption under both benchmark configurations. The second row reports gain-cost score and SLA violation, which jointly show whether the scheduler improves the benefit-cost trade-off while protecting deadline-sensitive tasks. The third row reports response time and task acceptance, which directly reflect user-perceived delay and service reliability. The fourth row focuses on the Smart-City IoT stress setting, where tighter deadlines, stronger burstiness, and higher risk levels make scheduling more difficult.

Under the generic 512 × 16 heavy-load condition with 2000 tasks, ER-RL-DVFS has mean makespan 1068.47 ± 16.77 and mean energy $(3.722 \pm 0.035) \times 10^5$ J. DQN-VMPlacement has 1329.22 ± 12.37 and $(4.305 \pm 0.077) \times 10^5$ J, GIoTDVFS-mGA has 1372.63 ± 32.65 and $(4.378 \pm 0.028) \times 10^5$ J, and RR has 2392.30 ± 21.53 and $(7.991 \pm 0.078) \times 10^5$ J. Every value is calculated from the same five seeds, resolving the earlier discrepancy between Section 8/Table 6 and the validation section.

Overall, the five-seed benchmark shows a favorable modeled energy-delay-SLA trade-off for ER-RL-DVFS relative to all eight comparators, but it does not establish global optimality or causal contribution of each proposed component. A separate supplied legacy package contains partial variants such as no-RL and no-migration under a different simulation regime; those values are intentionally not merged into the canonical five-seed results because doing so would create a non-like-for-like ablation. The canonical package does not contain matched runs that independently remove (a) the priority score, (b) hierarchical filtering, or (c) selective RL, nor a matched "RL-for-all-decisions" variant. We therefore moderate component-level causal claims and specify the required factorial ablation as a direct-implementation experiment rather than manufacturing synthetic component effects from incompatible data.

Code and Data Availability

The reproducibility package accompanying the manuscript contains the canonical five-seed raw data (`results/raw_results.csv`), aggregate and 95% CI summaries, the figures used in the manuscript, the controlled benchmark generator (`simulation/run_simulation.py`; also retained under `canonical_5seed/`), and a deterministic post-processing script (`analysis/recompute_statistics.py`). The frozen seed-level CSV is the authoritative source for the reported numerical values. The supplied generator is a calibrated profile-based stochastic benchmark and is not a line-by-line implementation of the proposed Q-learning/DVFS scheduler or a neural DQN training program. No public GitHub or Zenodo DOI is claimed at the time of submission; the complete accompanying package is provided with the manuscript, and a public archival deposit is planned after repository metadata and licensing are finalized.

11 Conclusion

This paper presented ER-RL-DVFS, a hierarchical energy-risk-aware scheduling framework for DVFS-enabled IoE workloads in virtualized cloud data centers. The framework integrates energy-risk task prioritization, DVFS-aware PM/VM profiling, hierarchical cluster–PM–VM candidate filtering, deterministic fast scoring, a formally specified selective tabular-RL refinement for near-tie decisions, and migration-aware correction. The multi-objective formulation is used as a design objective for these coordinated online decisions rather than as a claim of exact global optimization, and the RL contribution is deliberately positioned as a lightweight selective mechanism rather than a new learning architecture.

The final numerical evidence is consolidated on one canonical controlled stochastic benchmark with five independent seeds $(7, 19, 41, 73, 101)$, two resource configurations, Generic-IoE and Smart-City-IoT workloads, and eight baselines: RR, Min-Min, Max-Min, EnergyAwareDVFS, PAFogIoTDVFS, GIoTDVFS-SFB, GIoTDVFS-mGA, and DQN-VMPlacement. Under 2000 Generic-IoE tasks and 512×16, ER-RL-DVFS obtains mean makespan 1068.47 ± 16.77, energy $(3.722 \pm 0.035) \times 10^5$ J, and EDP 0.3977×10^9, compared with 1329.22 ± 12.37, $(4.305 \pm 0.077) \times 10^5$ J, and 0.5722×10^9 for DQN-VMPlacement. The EDP reduction relative to DQN is 30.50% in this setting, 31.98% in Generic-IoE 1024×32, and 36.32% in the Smart-City 512×16 stress case. The last value is distinctly larger because the modeled Smart-City regime produces a stronger relative makespan separation rather than because the same percentage has been reused across scenarios.

The evaluation boundary is equally important. The supplied executable generator is an aggregate profile-based benchmark: it does not execute the proposed Q-table online, train the DQN comparator, or explicitly instantiate K_c, K_p, K_v, β, ζ, μ_1, and μ_2. Consequently, no compute-matched training claim, numerical experimental Γ, or causal factorial ablation is inferred from these data. Scheduler wall-clock decision latency, memory footprint, and adaptation cost are also not measured. Future work should therefore implement Algorithms 1–7 directly in a trace-driven or CloudSim/iFogSim/container testbed; record the complete filtering, DVFS, and migration constants; perform matched component-wise ablations and priority-weight sensitivity analysis; compare DQN and selective RL under equal environment-interaction and tuning budgets; and measure online decision latency, memory, convergence, and robustness to uncertain execution-time, energy, queue-delay, and migration estimates.

Acknowledgment

The research work is supported in part by the Federal Ministry of Research, Technology, and Space (BMFTR), Germany, through the Project 6GEM+ under Grant 16KIS2411; and in part by the European Union's Horizon Europe research and innovation programme under the 6G-Path project (Grant No. 101139172).

References

[1] Amir Javadpour, Arun Kumar Sangaiah, Pedro Pinto, Forough Ja'fari, Weizhe Zhang, Ali Majed Hossein Abadi, and HamidReza Ahmadi. An energy-optimized embedded load balancing using dvfs computing in cloud data centers. *Computer Communications*, 197:255–266, 2023.

[2] Amir Javadpour, Guojun Wang, Samira Rezaei, and Shuhong Chend. Power curtailment in cloud environment utilising load balancing machine allocation. In *2018 IEEE smartworld, ubiquitous intelligence & computing, advanced & trusted computing, scalable computing & communications, cloud & big data computing, internet of people and smart city innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 1364–1370. IEEE, 2018.

[3] Hadi Zavieh, Amir Javadpour, Forough Ja'fari, Arun Kumar Sangaiah, and Adam Słowik. Enhanced efficiency in fog computing: A fuzzy data-driven machine selection strategy: H. zavieh et al. *International Journal of Fuzzy Systems*, 26(1):368–389, 2024.

[4] Maraga Alex, Sunday O. Ojo, and Fred Mzee Awuor. Carbon-aware, energy-efficient, and sla-compliant virtual machine placement in cloud data centers using deep q-networks and agglomerative clustering. *Computers*, 14(7):280, 2025. doi: 10.3390/computers14070280.

[5] Xiaomo Yu, Jie Mi, Ling Tang, Long Long, and Xiao Qin. Dynamic multi objective task scheduling in cloud computing using reinforcement learning for energy and cost optimization. *Scientific Reports*, 15:45387, 2025. doi: 10.1038/s41598-025-29280-z.

[6] Wenfan Zhang and Haijiao Ou. Reinforcement learning based multi objective task scheduling for energy efficient and cost effective cloud edge computing. *Scientific Reports*, 15:41716, 2025. doi: 10.1038/s41598-025-25666-1.

[7] Lucas Rister Machado, Gregory de Moraes Rossato, Antonio Carlos Schneider Beck, Michael G. Jordan, and Mateus Beck Rutzig. Energy-aware dvfs-driven workload provisioning in heterogeneous cloud faas architectures. *The Journal of Supercomputing*, 82:62, 2026. doi: 10.1007/s11227-025-08171-0.

[8] Monire Safari and Reihaneh Khorsand. Energy-aware scheduling algorithm for time-constrained workflow tasks in dvfs-enabled cloud environment. *Simulation Modelling Practice and Theory*, 87:311–326, 2018.

[9] Rodrigo N Calheiros and Rajkumar Buyya. Energy-efficient scheduling of urgent bag-of-tasks applications in clouds through dvfs. In *2014 IEEE 6th international conference on cloud computing technology and science*, pages 342–349. IEEE, 2014.

[10] Georgios L Stavrinides and Helen D Karatza. An energy-efficient, qos-aware and cost-effective scheduling approach for real-time workflow applications in cloud computing systems utilizing dvfs and approximate computations. *Future Generation Computer Systems*, 96: 216–226, 2019.

[11] Amir Javadpour, AmirHossein Nafei, Forough Ja'fari, Pedro Pinto, Weizhe Zhang, and Arun Kumar Sangaiah. An intelligent energy-efficient approach for managing ioe tasks in cloud platforms. *Journal of Ambient Intelligence and Humanized Computing*, 14:3963–3979, 2023. doi: 10.1007/s12652-022-04464-x.

[12] Sanjay Moulik and Yanshul Sharma. FRESH: Fault-tolerant real-time scheduler for heterogeneous multiprocessor platforms. *Future Generation Computer Systems*, 161:214–225, 2024. doi: 10.1016/j.future.2024.07.008.

[13] Sanjay Moulik and Richa Sarma. ECO-FAST: An energy cognizant fault-tolerant scheduler for heterogeneous computing systems. In *2025 29th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 1–5, 2025. doi: 10.1109/DS-RT68115.2025.11185082.

[14] Yanshul Sharma, Sanjay Moulik, and Shounak Chakraborty. RESTORE: Real-time task scheduling on a temperature aware FinFET-based multicore. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 608–611, 2022. doi: 10.23919/DATE54114.2022.9774647.

[15] S. Chandrasiri and D. Meedeniya. Energy-efficient dynamic workflow scheduling in cloud computing. *Sensors*, 25(5):1428, 2025. doi: 10.3390/s25051428.

[16] Anton Beloglazov and Rajkumar Buyya. Energy efficient resource management in virtualized cloud data centers. In *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 826–831. IEEE, 2010.

[17] Monireh H Sayadnavard, Abolfazl Toroghi Haghighat, and Amir Masoud Rahmani. A reliable energy-aware approach for dynamic virtual machine consolidation in cloud data centers. *The Journal of Supercomputing*, 75(4):2126–2147, 2019.

[18] Zhou Zhou, Zhi-gang Hu, Jun-yang Yu, Jemal Abawajy, and Morshed Chowdhury. Energy-efficient virtual machine consolidation algorithm in cloud data centers. *Journal of Central South University*, 24(10):2331–2341, 2017.

[19] Hui Wang and Huaglory Tianfield. Energy-aware dynamic virtual machine consolidation for cloud datacenters. *IEEE Access*, 6:15259–15273, 2018.

[20] Srinivas Byatarayanapura Venkataswamy, Indrajit Mandal, and Seetharam Keshavarao. Chicwhale optimization algorithm for the vm migration in cloud computing platform. *Evolutionary Intelligence*, 13(4):725–739, 2020.

[21] Young Choon Lee and Albert Y Zomaya. Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing*, 60(2):268–280, 2012.

[22] Youwei Ding, Xiaolin Qin, Liang Liu, and Taochun Wang. Energy efficient scheduling of virtual machines in cloud with deadline constraint. *Future Generation Computer Systems*, 50:62–74, 2015.

[23] Bhanu Pratap Singh, S Ananda Kumar, Xiao-Zhi Gao, Maulik Kohli, and Sanskar Katiyar. A study on energy consumption of dvfs and simple vm consolidation policies in cloud computing data centers using cloudsim toolkit. *Wireless Personal Communications*, pages 1–13, 2020.

[24] Seyedeh Maedeh Mirmohseni, Chunming Tang, and Amir Javadpour. Using markov learning utilization model for resource allocation in cloud of thing network. *Wireless Personal Communications*, 115(1):653–677, 2020.

[25] J. Zeng, D. Ding, K. Kang, et al. Adaptive drl-based virtual machine consolidation in energy-efficient cloud data centers. *IEEE Transactions on Parallel and Distributed Systems*, 33(11):2991–3002, 2022. doi: 10.1109/TPDS.2022.3147851.

[26] Zhibao Wang, Shuaijun Chen, Lu Bai, Juntao Gao, Jinhua Tao, Raymond R. Bond, and Maurice D. Mulvenna. Reinforcement learning based task scheduling for environmentally sustainable federated cloud computing. *Journal of Cloud Computing*, 12:174, 2023. doi: 10.1186/s13677-023-00553-0.