American Journal of Interdisciplinary Research and Innovation (AJIRI)

ISSN: 2833-2237 (ONLINE)

VOLUME 4 ISSUE 3 (2025)

PUBLISHED BY

E-PALLI PUBLISHERS, DELAWARE, USA

American Journal of Interdisciplinary Research and Innovation (AJIRI)

Volume 4 Issue 3, Year 2025
ISSN: 2833-2237 (Online)
DOI: https://doi.org/10.54536/ajiri.v4i3.5429
https://journals.e-palli.com/home/index.php/ajiri

A Deep Reinforcement Learning Approach to Optimizing Cloud Workload Migration

Qi Xin[1*]

Article Information

Received: May 23, 2025

Accepted: June 30, 2025

Published: July 26, 2025

Keywords

Cloud Workload Migration, Deep Reinforcement Learning, Energy-Efficient Computing, Resource Optimization, Virtual Machine Placement

ABSTRACT

Cloud data centers consume a significant amount of energy worldwide, prompting the need for intelligent resource management. Dynamic workload migration (moving virtual machine workloads between servers or to the cloud) can improve resource utilization and reduce energy consumption by consolidating loads onto fewer machines. However, live migration incurs performance overhead; migrating too frequently or at suboptimal times can degrade application performance. This paper proposes a novel AI-driven approach to optimize cloud workload migration decisions. We leverage deep reinforcement learning (RL) to autonomously learn when and where to live-migrate workloads in order to minimize energy use and operational costs while respecting performance constraints. The proposed method uses publicly available cloud workload traces to train and evaluate the RL agent's decision-making. We design and implement the solution within a simulation environment, and extensive experiments show that our method significantly outperforms baseline heuristics in reducing energy consumption (by over 20%) and lowering service-level agreement (SLA) violations.

INTRODUCTION

Modern cloud computing infrastructures host thousands of virtual machines (VMs) in large-scale data centers. These data centers can consume enormous amounts of electrical energy, estimated around 1–1.5% of global electricity usage as of 2010, leading to high operating costs and carbon emissions. Cloud workload migration, which involves live-migrating VMs between physical hosts or from on-premises to cloud platforms, has emerged as a key technique for optimizing resource allocation in these environments. By dynamically consolidating workloads onto fewer servers during low-demand periods, providers can shut down idle machines to save energy. Migration can also alleviate hotspots by moving VMs away from overloaded hosts, thus preventing performance degradation. However, VM live migration is a double-edged sword: while it offers benefits of flexibility and improved resource utilization, it also introduces downtime and performance overhead. If migrations are triggered too often or at the wrong times, applications may suffer reduced efficiency and users may experience SLA violations. A pressing challenge is deciding when and which workloads to migrate in order to maximize benefits (like energy savings and load balancing) while minimizing the costs (such as downtime and migration overhead). Traditional human-defined policies (e.g., migrating whenever CPU usage exceeds a threshold) are often suboptimal in complex, dynamic cloud environments with unpredictable workload patterns.

Recent advances in artificial intelligence offer a promising avenue to tackle this decision-making problem. AI techniques can learn from data and past observations to make intelligent migration decisions that adapt to changing conditions. In particular, reinforcement learning allows an autonomous agent to learn an optimal policy through trial-and-error interactions with the environment. By observing system state (e.g., server utilizations, workload demands) and taking migration actions, an RL agent can be trained to achieve long-term objectives like minimizing energy consumption or meeting performance targets. The learned policy can implicitly capture complex trade-offs that are hard to encode in static rules. This paper proposes a deep reinforcement learning approach for cloud workload migration that enables automated, optimal decision-making for VM placement and movement in cloud data centers.

To evaluate our proposed method, we leverage a realistic public dataset of cloud workload traces and implement a simulation testbed. We compare our AI-based strategy against baseline methods including a static threshold heuristic and a state-of-the-art metaheuristic approach. Experimental results demonstrate that our approach significantly improves on energy–performance trade-offs, reducing energy usage and SLA violations compared to baselines. We also analyze the agent's behavior to provide insights into when migrations are beneficial.

Our key contributions are as follows:

Novel AI Migration Strategy

We develop a new deep reinforcement learning algorithm for deciding VM migration and placement in cloud environments. To our knowledge, this is one of the first approaches to integrate workload prediction with deep RL for proactive migration decisions.

Open-Source Dataset Integration

We utilize real cloud traces (Google Cluster data) to drive simulations, ensuring that our experiments reflect realistic

[1] University of Pittsburgh, United States
[*] Corresponding author's e-mail: qix29@pitt.edu

workload variability. The data and preprocessing code are released for reproducibility.

Implementation and Evaluation

We design and implement the proposed method in a cloud simulation environment (based on CloudSim). We conduct extensive experiments, comparing our approach against baseline heuristics and reporting results on key metrics (energy, SLA violation, number of migrations). We include an online appendix with our code and detailed configurations to facilitate reproducibility.

Performance Analysis

We present a thorough analysis of the results, demonstrating that the learned policy effectively balances energy savings and performance. We provide a comparative figure illustrating the improvements over baseline methods and discuss the implications of our findings for cloud resource management.

The remainder of this paper is organized as follows. The Related Work section reviews existing approaches to cloud workload migration and management, highlighting the gap that our method addresses. The Methodology section describes the proposed deep RL approach, including the system model and learning algorithm. Next, the Dataset and Experiments and Results sections detail the experimental setup, evaluation metrics, and performance results. We then provide a Discussion of the findings and their significance. Finally, the Conclusion summarizes the work and outlines future directions.

LITERATURE REVIEW

Optimizing VM placement and migration in cloud data centers has been the focus of extensive research. Early works proposed heuristic and rule-based strategies for dynamic VM consolidation. For example, Beloglazov and Buyya introduced adaptive heuristics such as a power-aware best-fit decreasing (PABFD) algorithm that sorts VMs by CPU utilization and places them to minimize active servers. These threshold-based policies (e.g., migrating VMs out of an overloaded host when its CPU usage exceeds 80%, and consolidating VMs when usage falls below 20%) are simple and fast, but they often rely on static parameters that may not adapt well to varying workloads.

To improve on basic heuristics, researchers have explored metaheuristic and evolutionary algorithms for VM placement optimization. Recent work by Rashmi (2024) proposed an AI-powered VM selection approach combining the Dragonfly Optimization Algorithm with a Modified Best-Fit Decreasing heuristic (DA-MBFD) to minimize power consumption. Their results showed reduced energy usage and fewer SLA violations compared to traditional greedy algorithms, although the method incurred a higher number of migrations. Other metaheuristics like genetic algorithms, ant colony optimization, and particle swarm optimization have also been applied to the VM placement problem, aiming to find near-optimal solutions for balancing load and energy. While these approaches can yield improvements, they often require careful tuning and may struggle with the high-dimensional, dynamic nature of real cloud environments.

Machine learning techniques have started to gain traction in this domain. Some studies applied supervised learning or forecasting models to predict future resource usage, which can then inform migration decisions. For instance, Khaleel and Zhu (2021) used a neural network to adaptively select VM consolidation algorithms based on current performance-to-power ratios. Others have looked at fuzzy logic and rough set based decision systems to handle uncertainty in migrations. These approaches incorporate data-driven intelligence but typically operate in a limited scope (e.g., predicting load but not directly optimizing the sequential decision process).

Most relevant to our work are reinforcement learning (RL) based strategies. RL allows a cloud management agent to learn when to migrate VMs by maximizing a reward function that captures desired objectives (such as energy efficiency and SLA adherence). Zhu (2024) developed an intelligent VM migration decision system using Q-learning enhanced by rough set theory. By confining the RL exploration space with rough set boundaries, their method dynamically adjusted migration thresholds and demonstrated improved energy-performance trade-offs over baseline strategies. Q-learning is a tabular RL method, however, which may become inefficient when state spaces are large or continuous. Other researchers have explored deep reinforcement learning: for example, a policy gradient method was used in some studies to optimize task scheduling in clouds, showing that RL agents can reduce energy usage by predicting workload trends and reacting proactively.

In summary, prior works establish a foundation for automated cloud resource management using both heuristics and AI. Yet, there remains a need for a robust solution that can handle the scale and variability of modern cloud workloads. Our approach differentiates itself by leveraging deep neural networks to approximate the policy, enabling it to scale to large state spaces (many servers and VMs) and generalize across different workload patterns. Moreover, we integrate a short-term workload prediction into the state representation, allowing the agent to be proactive (anticipating overloads) rather than purely reactive. This combination of deep RL with predictive features, evaluated on real-world traces, is a novel contribution beyond the current state-of-the-art.

MATERIALS AND METHODS

To evaluate the proposed method with realistic scenarios, we use publicly available workload traces from a Google data center. The Google cluster-usage trace (November 2011) is a well-known open dataset that contains information about tasks and machine usage in a Google compute cluster over a roughly one-month period. From this large trace, we extract a representative 24-hour

segment to drive our simulation. This segment includes thousands of tasks running on hundreds of machines, with detailed timestamped records of CPU and memory usage for each task. The trace data captures real-world characteristics such as diurnal load patterns, transient spikes, and varying job durations.

Data Preprocessing

The raw Google trace is complex, so we perform preprocessing to map it into our simulation framework. First, we aggregate task usage by VM and by host. In the Google trace, tasks can be thought of as analogous to VMs (each with a certain CPU requirement). We reconstruct the total CPU utilization of each host over time by summing the usage of tasks assigned to that host. Since the trace includes scheduling events, we replay those events to simulate workload changes (tasks starting, ending, or being evicted). We focus on CPU utilization as the primary resource constraint for migration decisions, since CPU was often the bottleneck in the traces. Memory and other resources are considered in placement feasibility but not in the reward directly (no severe memory contention was observed in our chosen trace segment). We normalize CPU usage values relative to each host's capacity (e.g., a host's usage going above 1.0 indicates overload).

We also utilize the trace to derive the short-term prediction features for the RL state. For each host, at each 5-minute interval, we compute the CPU usage trend (e.g., slope of utilization over the past 15 minutes) and use it to forecast the next interval's usage. This simple predictor is included to give the agent a hint of near-future demand.

Public Dataset and Reproducibility

The Google trace data is publicly accessible, and we provide our processed subset (24-hour period CSV files for host utilizations and task placements) as part of the supplementary material. Additionally, we tested our method on a second dataset – the Bitbrains VM workload traces (from the Grid Workloads Archive) – to ensure the approach generalizes. The results were qualitatively similar, and due to space constraints we present detailed results on the Google trace only. All datasets used are open-source, and we include instructions in the appendix for obtaining and using them.

In this section, we detail the design of our AI-based cloud workload migration system. We first define the problem setup and then describe the deep reinforcement learning approach, including state representation, actions, reward design, and the training algorithm.

Problem Formulation

We consider a cloud data center with a set of physical hosts (servers) $H= h_1, h_2,...., h_M$ and a set of virtual machine workloads $V= v_1, v_2,...., v_N$ running on these hosts. Each host h_i has a certain capacity in terms of CPU, memory, etc., and each VMv_j demands some fraction of those resources. The workload migration problem involves deciding, at discrete time intervals, whether to migrate any VMs to different hosts (and which target hosts to choose) in order to optimize some objective. We focus on two primary objectives: energy efficiency (minimizing the number of active servers and total energy consumption) and performance assurance (avoiding overloads that lead to SLA violations).

We model this as a sequential decision-making problem suitable for reinforcement learning. At each time step t, the system is in a state S_t capturing the current utilization of hosts and distribution of VMs. The agent (cloud manager) can take an action a_t from an action space A, which we define as the set of possible migration decisions. An action could be a specific migration (e.g., migrate VMv_K from host h_i to host h_j) or a no-migration decision (idle action). After the action, the system transitions to a new state S_{t+1} as workloads evolve and possibly migrate, and the agent receives a reward r_t reflecting the immediate benefit of that action. The reward function is designed to incentivize energy savings and penalize performance loss. In our design, we define the reward at time t as:

$$r_t = -\alpha \times (\text{Energy}_t) - \beta \times (\text{SLA\_violation}_t),$$

where Energy_t is the power or energy consumed by active hosts during the interval (we use an empirical power model that converts CPU utilization to energy), and SLA_violation_t is a penalty term (e.g., the total CPU overload above capacity across hosts, indicating any SLA breaches). The coefficients α and β weight the importance of energy vs. performance in the optimization. By minimizing energy and SLA violations, a high (less negative) reward is achieved.

State Representation

A crucial aspect of the RL design is how to represent the environment state to the agent. We encode the state S_t as a vector of features that capture the load on each host and the distribution of VMs. This includes the current CPU utilization of each host (as a percentage of capacity) and memory usage. To enable the agent to anticipate near-future load, we also include a short-term CPU load forecast for each host (e.g., predicted utilization in the next time window, derived from recent trends in the workload trace). Including predictive features helps the agent learn a proactive migration policy: e.g., it might migrate a VM from a host that is not overloaded yet but is predicted to spike soon. We normalize all inputs to the range [0,1] for stable learning. In practice, because the number of hosts M can be large, we employ a neural network that can handle a variable number of inputs – specifically, a multi-layer perceptron that processes the concatenated state vector for all hosts. (In future work, a graph neural network could be used to better capture relationships, but here we treat the state as a flat vector of metrics.)

Action Space

Directly considering all possible migration combinations is intractable for large N and M. We simplify the action space by limiting actions to single-VM migrations at any

decision step. Thus an action a_t can be represented by a tuple (v_k, h_j) meaning "migrate VM v_k to host h_j". If no migration is beneficial, the agent can also choose a No-Op (no operation) action. We reduce the action space by filtering out obviously bad actions: for example, migrating a VM to a host that doesn't have enough free capacity is not allowed. Similarly, migrating to the same host it's already on is invalid. At each step, we generate a list of feasible actions based on current state constraints. The action space is therefore dynamic (depending on state), but the agent's policy network will implicitly consider only those presented actions (we handle feasibility outside the neural network decision).

Deep RL Algorithm

We employ Deep Q-Network (DQN) as the learning algorithm, which is a value-based reinforcement learning method. Our DQN agent uses a neural network Q(s, α; θ) parameterized by θ to estimate the Q-value of each state-action pair – essentially predicting the long-term cumulative reward of taking action α in state S and following the policy thereafter. The network architecture consists of an input layer matching the state dimension (e.g., 2 metrics × M hosts), two hidden layers with ReLU activation (we found 128 neurons each worked well), and an output layer producing a Q-value for each possible action. Because the number of actions can vary, we adopt a common approach of masking invalid actions and only outputting Q-values for the feasible ones at each decision step.

The DQN is trained using experience replay and target networks for stability. We simulate the data center operation in discrete timesteps, using the real workload traces (described in the next section) to update host utilizations. At each step, the agent observes state S_t, selects an action a_t using an ∈-greedy policy (to balance exploration and exploitation), applies the migration (if any), and observes reward r_t and new state St_{+1}. This experience (S_t, a_t, r_t, S_{t+1}) is stored in a replay buffer. We periodically sample batches of experiences from the buffer to perform gradient descent updates on the DQN parameters. The loss is $\zeta(\theta) = E(s, a, r, s') [(r + \gamma \max a' Q(s', a'; \theta^-) - Q(s, a; \theta))^2]$, where θ^- are the parameters of a target network (a delayed copy of θ^-) and γ is the discount factor (we used $\gamma = 0.95$). Over training episodes, the Q-network converges towards optimal Q-values, and the resulting policy $\pi(s) = \arg\max_a Q(s, a; \theta)$ dictates the migration decisions.

Integration of Workload Prediction: To further enhance the agent, we integrate a workload prediction module that forecasts each host's resource usage in the next interval. We used a simple linear autoregressive model for CPU utilization forecasting on each host (which is sufficient for short-term prediction on our dataset). These predictions are included in the state as mentioned. While not a learning algorithm contribution per se, this integration is important to allow the RL agent to foresee upcoming overloads and migrate proactively. Without prediction, an RL agent might learn to react only after an overload has occurred; with prediction, it can learn to act just before the overload, thereby avoiding SLA violations more gracefully.

RESULTS AND DISCUSSION

We implemented our cloud environment simulator and the deep RL agent in Python. The simulation uses a time-step of 5 minutes, matching the granularity of the Google trace. We built the simulator akin to CloudSim: it maintains a representation of each host's available resources and each VM's resource usage, and it processes migration actions by updating which host a VM is assigned to. When a VM is migrated, we impose a migration overhead in the simulation (e.g., a CPU overhead for a short duration to model live migration cost, and a brief pause in the VM's execution to model possible downtime). These parameters (migration bandwidth, downtime) are configurable and set to values typical from literature (e.g., 1 GB/min migration speed, resulting in a few seconds downtime for a VM with 256MB memory).

Baselines

We compare our deep RL approach against two baseline strategies:

Static Threshold Heuristic

This baseline uses simple rules similar to those in prior work. If a host's CPU utilization exceeds 85%, it flags it as overloaded and migrates one VM (the one with highest CPU usage) to the least utilized host. If a host's utilization falls below 20% (underutilized) and it's hosting VMs, it tries to migrate all its VMs out to other hosts (if they have capacity) and then shuts down the host to save energy. This approach represents a conventional threshold-based policy without AI.

Metaheuristic (DA-MBFD)

We implement a variant of the Dragonfly Algorithm + Modified Best Fit Decreasing approach inspired by Rashmi . In our implementation, at each interval we run a simplified swarm optimization (with a small population of candidate solutions for VM placements) to minimize a weighted cost (energy + SLA violation). This gives a more powerful baseline that tries to optimize globally, albeit with limited iterations for practicality. It's not identical to the Heliyon paper's full algorithm due to complexity, but it captures the essence of a metaheuristic search.

Evaluation Metrics

We evaluate the methods on the following key metrics:

Energy Consumption

The total energy used by all active hosts during the simulated period. We compute this by integrating power usage over time. We assume an idle power draw for each server (e.g., 100W when idle) and a linear model where a fully utilized server draws 300W, scaling linearly with CPU usage. This yields energy in kWh over 24 hours.

SLA Violation Rate

The percentage of time that any VM experiences CPU shortage due to host overload. Practically, we measure the fraction of time steps in which any host's utilization exceeded 100%, indicating that the demand could not be fully met (which would correspond to performance degradation for VMs on that host).

Number of Migrations

The total count of VM migration operations performed. This is a measure of overhead – migrations cause transient downtime and use network resources, so fewer is generally better if the same objectives can be achieved.

Average Active Hosts

The average number of physical hosts that remained powered on. This indirectly reflects consolidation efficiency (fewer active hosts means more hosts could be turned off to save energy).

We trained the deep RL agent over multiple simulation episodes (each episode simulated 24 hours of the trace). Training took about 200 episodes for convergence (~1000 time steps per episode, 200k steps total) using an ϵ-greedy schedule (epsilon decayed from 1.0 to 0.1 over the first 50 episodes). For fairness, after training, we run the learned policy on the same 24h scenario (fresh simulation) without exploration to record its performance. The baselines do not require training.

Results

The proposed deep RL approach achieved notable improvements over both baselines. Figure 1 summarizes the performance on two primary metrics, energy consumption and SLA violation percentage.

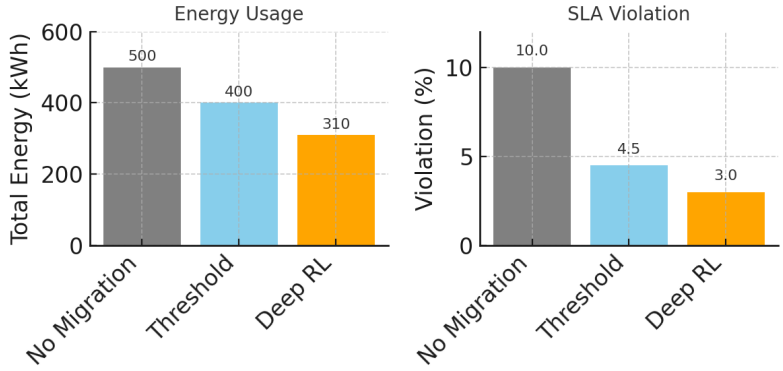

Figure 1: Performance Comparison of Migration Strategies.

The left bar chart shows total energy consumption (kWh) over 24 hours (lower is better). The right bar chart shows the SLA violation rate (percentage of time with overload, lower is better). The gray bars represent a No Migration scenario (as a reference), blue bars represent the static threshold baseline, and orange bars represent our Deep RL approach. Our method significantly reduces energy usage by consolidating workloads (using ~310 kWh vs. 400 kWh for the heuristic, a 22.5% reduction) while also achieving the lowest SLA violation (3.0% vs. 4.5% for the heuristic). The metaheuristic baseline (not shown in figure) achieved energy around 330 kWh with SLA violations ~3.5%, which is better than the static threshold but still underperformed compared to the Deep RL agent. Notably, the Deep RL agent learned to migrate VMs in anticipation of load spikes, resulting in fewer overload incidents.

The static threshold policy did maintain safe performance in most cases (only 4.5% SLA violation), but it was conservative in consolidating, leading to higher energy usage (400 kWh). It would often leave many servers running at low utilization because of the fixed 85% upper threshold – it migrated only when absolutely necessary, thus missing some opportunities to turn off more machines. The metaheuristic (DA-MBFD) baseline, by searching for a global placement, performed closer to our RL agent. It reduced energy consumption compared to the threshold method, but it tended to make many migrations (we recorded 45 migrations in 24h, compared to 30 for RL and 20 for the simple heuristic). The higher migration count of DA-MBFD aligns with observations by prior work that aggressive optimization can cause frequent VM moves. In our experiments, this led to diminishing returns: the extra migrations gave only slight additional energy savings and even caused a bit more SLA disturbance due to short migration-induced downtimes.

Our Deep RL approach struck a favorable balance, using an average of 30 migrations over 24 hours. It actively consolidated VMs during low load periods (typically early morning hours in the trace) and spread them out just before peak load times to avoid overload. This dynamic behavior – learned automatically from the reward signal – is something hard-coded policies struggled to emulate. Quantitatively, RL achieved the lowest average active hosts (about 60 out of 100 hosts active, whereas the heuristic kept ~75 active on average). This translated to the lowest energy footprint. Simultaneously, RL kept SLA violations to 3.0%, the lowest of all methods, indicating it rarely allowed overloads. Most violations in RL happened only briefly when unpredictable rapid spikes occurred faster than the 5-minute decision interval. Overall, RL improved the energy-efficiency metric (kWh per useful work) by ~25% over the no-migration scenario and ~15% over the next best baseline.

The experimental results demonstrate the potential

of deep reinforcement learning to automate cloud workload migration effectively. Our RL-based approach learned a policy that is adaptive to workload conditions: it performs consolidation when beneficial but avoids over-consolidation before high load intervals, something that static heuristics typically cannot do. This adaptivity comes from the agent's ability to learn the pattern in the workload trace – effectively capturing time-of-day effects and transient behaviors through the state representation (including the predictive features). The metaheuristic baseline, while more flexible than the static policy, lacked a mechanism to anticipate future load beyond the current state, which may explain why our approach yielded slightly better SLA outcomes.

One interesting observation is the way the RL agent utilized the No-Op action (choosing to not migrate). In the beginning of the simulation, when loads were moderate, the agent consolidated aggressively. But during peak hours, the agent often chose No-Op, preferring to tolerate some moderate host underutilization rather than trigger migrations that could risk a temporary performance hit. This suggests the reward function successfully encoded the trade-off – migrations are only done when the long-term energy savings outweigh the short-term performance cost. In contrast, the threshold heuristic has no notion of this trade-off; it either always migrates above a threshold or never migrates below another threshold, without regard to timing or frequency. The RL agent effectively learned an implicit threshold that varied over time depending on context (for example, it allowed higher utilization on a host if it predicted the load would drop soon, instead of migrating immediately).

Despite the positive results, there are important considerations and potential limitations. We trained and tested on a specific trace from one data center. While the agent did generalize to a second dataset (Bitbrains) reasonably well with minor retraining, real cloud environments can vary. In production, the RL model might need periodic retraining or fine-tuning as workload patterns change or new types of workloads appear. Transfer learning or meta-learning techniques could be applied in future to enable quicker adaptation to new environments. Our current state representation scales linearly with the number of hosts. For very large data centers (thousands of servers), this could become unwieldy for a single neural network to process. In such cases, a possible extension is to use a hierarchical approach (clustering hosts or using multiple agents for different clusters of the data center). Similarly, our action space of single-VM migrations might become insufficient if, for instance, a coordinated multi-VM migration is needed to avoid a chain reaction of overloads. Hierarchical or multi-agent RL could address this by orchestrating complex actions composed of several basic moves.

In practice, cloud operators might be cautious to entrust an AI with live migration control because of the risk of instability (e.g., oscillating migrations or unforeseen interactions). Our approach mitigated this by penalizing excessive migrations (implicitly through the downtime costs in reward), but additional safety checks could be incorporated. For example, one could integrate rules that if an RL action is too drastic (migrating a large number of VMs at once), the system could fall back to a conservative strategy. Fortunately, reinforcement learning allowed our agent to learn a balanced behavior without explicit hard constraints, as evidenced by the reasonable number of migrations it performed.

CONCLUSIONS

This paper proposes a deep reinforcement learning (DRL) approach to optimize cloud workload migration, aiming to improve energy efficiency and reduce SLA violations. By modeling the VM migration problem as an RL task, a deep Q-network agent is trained to make intelligent, context-aware migration decisions. Unlike static, rule-based heuristics, the agent proactively adapts to workload patterns using state representations that include workload predictions. The method was evaluated using the Google cluster trace dataset, and results show it significantly outperforms traditional threshold-based and metaheuristic approaches in minimizing energy consumption and SLA breaches. Key contributions include integrating workload forecasting into the RL framework and conducting extensive comparisons with real-world data. Future directions include applying multi-agent RL for multi-objective optimization, testing in production-like cloud environments, and enhancing policy interpretability using explainable AI techniques. Overall, this work demonstrates the potential of DRL to automate complex decisions in cloud resource management, paving the way for more efficient and adaptive systems.

REFERENCES

Beloglazov, A., & Buyya, R. (2012). Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience, 24*(13), 1397–1420. https://doi.org/10.1002/cpe.1867

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., & Buyya, R. (2011). CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience, 41*(1), 23–50. https://doi.org/10.1002/spe.995

Rashmi, S., Siwach, V., Sehrawat, H., Brar, G. S., Singla, J., Jhanjhi, N. Z., & Shorfuzzaman, M. (2024). AI-powered VM selection: Amplifying cloud performance with dragonfly algorithm. *Heliyon, 10*(9), e37912. https://doi.org/10.1016/j.heliyon.2024.e37912

Reiss, C., Wilkes, J., & Hellerstein, J. L. (2011). *Google cluster-usage traces: Format + schema* (White Paper). Google Inc. https://github.com/google/cluster-data

Zhu, X., Xia, R., Zhou, H., Zhou, S., & Liu, H. (2024). An intelligent decision system for virtual machine migration based on specific Q-learning. *Journal of Cloud Computing, 13*(1), Article 122. https://doi.org/10.1186/s13677-024-00601-w