ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/379958261

Optimizing Resource Allocation in Cloud Infrastructure through AI Automation: A Comparative Study

Article *in* Journal of Knowledge Learning and Science Technology ISSN 2959-6386 · May 2023

DOI: 10.60087/jklst.vol2.n2.p302

CITATIONS
15

READS
453

3 authors:

Musarath Jahan Karamthulla
Independent Researcher
8 PUBLICATIONS **58** CITATIONS
SEE PROFILE

Ravish Tillu
Royal Bank of Canada
11 PUBLICATIONS **50** CITATIONS
SEE PROFILE

Jesu Narkarunai Arasu Malaiyappan
18 PUBLICATIONS **103** CITATIONS
SEE PROFILE

All content following this page was uploaded by Jesu Narkarunai Arasu Malaiyappan on 20 April 2024.

The user has requested enhancement of the downloaded file.

ISSN: 2959-6386 (Online), Vol. 2, Issue 2

Journal of Knowledge Learning and Science Technology

journal homepage: https://jklst.org/index.php/home

Optimizing Resource Allocation in Cloud Infrastructure through AI Automation: A Comparative Study

Musarath Jahan Karamthulla[1], Jesu Narkarunai Arasu Malaiyappan[2], Ravish Tillu[3]

[1]TransUnion, USA
[2]Meta Platforms Inc, USA
[3]RBC Capital Markets, USA

Abstract

Optimizing resource allocation in cloud infrastructure is paramount for ensuring efficient utilization of computing resources and minimizing operational costs. With the proliferation of diverse workloads and dynamic user demands, manual resource management becomes increasingly challenging. In this context, artificial intelligence (AI) automation emerges as a promising approach to enhance resource allocation efficiency. This paper presents a comparative study of various AI techniques applied to optimize resource allocation in cloud environments. We explore the efficacy of machine learning, evolutionary algorithms, and deep reinforcement learning methods in dynamically allocating resources to meet performance objectives while minimizing costs. Through a comprehensive evaluation of these approaches using real-world datasets and simulation experiments, we highlight their strengths, limitations, and comparative performance. Our findings provide valuable insights into the effectiveness of AI-driven resource allocation strategies, enabling cloud providers and practitioners to make informed decisions for enhancing cloud infrastructure management.

Keywords: Cloud Computing, Resource Allocation, Artificial Intelligence, Automation, Machine Learning, Evolutionary Algorithms, Deep Reinforcement Learning, Comparative Study.

Article Information:
Article history: *Received:* 01/05/2023 *Accepted:* 10/05/2023 *Online:* 16/05/2023 *Published*: 16/05/2023
DOI: https://doi.org/10.60087/jklst.vol2.n2.p302

Correspondences'author: Musarath Jahan Karamthulla

Introduction

The advent of cloud computing has revolutionized the way computing resources are provisioned and managed,

offering unparalleled scalability, flexibility, and cost-effectiveness. Cloud infrastructure enables users to access a wide array of resources on-demand, ranging from computing power and storage to networking capabilities, without the need for heavy upfront investments in hardware. However, as the demand for cloud services continues to surge and workloads become increasingly diverse and dynamic, efficient resource allocation becomes a critical challenge for cloud providers.

Optimizing resource allocation in cloud infrastructure involves dynamically assigning computing resources to various tasks and applications to ensure optimal performance and cost efficiency. Traditional approaches to resource allocation often rely on static heuristics or manual intervention, which are ill-equipped to handle the complexity and variability of modern cloud environments. Moreover, the lack of scalability and adaptability of these approaches hinders their effectiveness in meeting evolving user demands.

In response to these challenges, artificial intelligence (AI) techniques have emerged as promising solutions for automating and optimizing resource allocation in cloud environments. By leveraging AI algorithms and methodologies, such as machine learning, evolutionary algorithms, and deep reinforcement learning, cloud providers can develop intelligent systems capable of autonomously managing resource allocation tasks with minimal human intervention.

This paper aims to investigate the role of AI automation in optimizing resource allocation in cloud infrastructure through a comparative study of different approaches. We examine the efficacy of machine learning algorithms, evolutionary algorithms, and deep reinforcement learning techniques in dynamically allocating resources to meet performance objectives while minimizing operational costs. By conducting a comprehensive evaluation of these approaches using real-world datasets and simulation experiments, we seek to provide insights into their strengths, limitations, and comparative performance.

Through this research, we aim to contribute to the advancement of AI-driven resource allocation strategies in cloud computing, enabling cloud providers and practitioners to make informed decisions for enhancing the efficiency and effectiveness of cloud infrastructure management.

Objectives:

Objective 1: Evaluate the effectiveness of machine learning algorithms, evolutionary algorithms, and deep reinforcement learning techniques in optimizing resource allocation within cloud infrastructure.

Objective 2: Assess the performance of AI-driven resource allocation strategies in meeting varying performance objectives such as response time, throughput, and resource utilization while minimizing operational costs.

Objective 3: Conduct a comparative analysis of different AI automation approaches to identify their strengths, limitations, and suitability for specific cloud computing environments and workload characteristics.

Literature Review

Cloud computing is a rapidly evolving field that requires efficient resource allocation to optimize performance and reduce costs. Several studies have proposed different methods for optimizing resource allocation in cloud infrastructure. Arvindhan and Kumar proposed a deep reinforcement learning-based actor-critic method that outperforms existing allocation methods in terms of resource utilization, energy efficiency, and overall cost [1]. George introduced an optimized task scheduled resource allocation model using a combination of Sen's multi-objective functions and auto-encoder deep neural network-based method, which significantly improves task scheduling efficiency, reduces energy usage, and makespan [2]. Funika, Koperek, and Kitowski used reinforcement learning and deep learning techniques to automate the scaling of heterogeneous resources in a compute cloud environment, resulting in cost optimization and improved performance [3]. Shi, Lingareddy, Suo, and Nguyen proposed the MinPlus algorithm for task scheduling, which demonstrated decreased turnaround time, enhanced resource load balancing, and increased throughput [4]. These studies provide valuable insights into the use of AI automation for optimizing resource allocation in cloud infrastructure.

Methodology

1. Data Collection: Gather real-world datasets representing various cloud workloads, including information on

resource usage patterns, application characteristics, and performance metrics. Additionally, collect data on cloud infrastructure parameters such as available computing resources, network bandwidth, and storage capacity.

2. Algorithm Selection: Choose a set of AI-driven resource allocation algorithms for evaluation, including machine learning algorithms (e.g., regression, classification, clustering), evolutionary algorithms (e.g., genetic algorithms, particle swarm optimization), and deep reinforcement learning techniques (e.g., deep Q-learning, policy gradients).

3. Experimental Setup: Design experiments to evaluate the performance of selected algorithms in optimizing resource allocation within cloud infrastructure. Define performance metrics such as response time, throughput, resource utilization, and cost efficiency to assess the effectiveness of each algorithm.

4. Training and Evaluation: Train AI models using the collected datasets and evaluate their performance through cross-validation or holdout validation techniques. Fine-tune hyper parameters and optimize model architectures to achieve optimal performance.

5. Simulation Experiments: Conduct simulation experiments to assess the scalability, robustness, and adaptability of AI-driven resource allocation strategies under various workload scenarios and cloud environment configurations. Use simulation tools such as Clouds or discrete-event simulators to mimic real-world cloud environments.

6. Comparative Analysis: Compare the performance of different AI algorithms in terms of their ability to meet performance objectives while minimizing operational costs. Analyse the strengths, limitations, and trade-offs of each approach based on experimental results and observations.

7. Sensitivity Analysis: Perform sensitivity analysis to investigate the impact of changing parameters such as workload intensity, resource constraints, and algorithmic parameters on the performance of AI-driven resource allocation strategies.

8. Statistical Analysis: Apply statistical techniques to analyse experimental results and determine the significance of observed differences between the performances of different algorithms. Conduct hypothesis testing to validate the effectiveness of AI automation in optimizing resource allocation in cloud infrastructure.

9. Discussion and Interpretation: Interpret experimental findings and discuss implications for cloud infrastructure management. Identify potential challenges, limitations, and future research directions for further improving AI-driven resource allocation techniques in cloud computing environments.

Background:

II. Literature Review

Conventional actor-critic methods often suffer from excessive variation and erroneous policy gradients due to the linear functional form used to estimate the action-value function. To address this issue, the Advantage Actor-Critic was developed, as noted by W. Zhang et al. (2021) and Zhu et al. (2022). In this approach, the actor makes decisions based on random chance, while the critic provides feedback in the form of scores, allowing the actor to adjust the likelihood of choosing a particular action accordingly. Advantage functions play a significant role in reducing policy gradient variance. However, when a neural network approximates the value or policy function, function approximation errors may arise, as highlighted by Jin et al. (2023) and D. Zhang et al. (2023).

In the context of radio access network offloading modeled as a Markov Decision Process, a reinforcement learning method based on a double-deep Q-network was designed, aiming to combat state-space explosion and dynamically discover optimal loading methods, as proposed by the authors. To enhance computational dumping efficiency, a deep learning technique based on the State-Action-Reward-State-Action (SARSA) framework was suggested by Serrano-Guerrero et al. (2021). Comparative simulation experiments conducted by the authors evaluated various

scheduling techniques, including the first-come, first-served (FCFS) algorithm and a genetic algorithm (GA). Results indicate that the proposed method surpasses competing algorithms in terms of makespan (total execution time) and resource utilization.

Function approximation errors pose challenges to actor-critic methods, particularly in high-dimensional or continuous action spaces, as discussed by Niu et al. (2021), Sankalp et al. (2022), and Tang et al. (2023). To mitigate this issue, an algorithm named Twin Delayed Deep Deterministic (TD3) policy gradient is proposed, incorporating three essential modifications to enhance function approximation accuracy.

The literature also discusses actor-critic deep reinforcement learning algorithms, comprising an actor suggesting resource allocation decisions and a critic evaluating their quality. These components, implemented as neural networks, are trained using a combination of supervised and reinforcement learning techniques (Ferratti et al., 2021).

Evaluation of the algorithm in a simulated cloud data center environment reveals its superiority over traditional resource allocation algorithms in terms of resource utilization and workload completion time. This suggests that the proposed approach has the potential to enhance the efficiency and adaptability of resource allocation in cloud data centers (Fernandez-Gauna et al., 2022; Sun et al., 2019).

In the realm of cloud computing, load balancing has emerged as a crucial aspect, driving the exploration and implementation of reinforcement learning (RL) algorithms. These RL algorithms offer versatile solutions to the dynamic challenges inherent in load balancing within cloud environments. Here, we present notable examples of RL algorithms that have been successfully applied to address this problem.

Firstly, Q-Learning stands out as a model-free RL algorithm adept at learning an optimal action-value function through iterative updates of Q-values, guided by rewards received from different states and actions. Its application in dynamic load balancing scenarios within cloud computing has been documented (Li et al., 2023).

Deep Reinforcement Learning represents a sophisticated integration of RL with deep neural networks, facilitating the acquisition of complex policies. Within cloud computing, this amalgamation has been leveraged to optimize the allocation of virtual machines to physical servers, thereby enhancing overall resource utilization.

Actor-Critic Methods, another category of RL algorithms, offer a hybrid approach by combining value-based methods, such as Q-learning, with policy-based methods like policy gradient methods. This amalgamation enables the learning of policies while simultaneously estimating their values, proving beneficial in optimizing virtual machine allocation and mitigating energy consumption in cloud environments.

Furthermore, priority-based scheduling strategies have been employed, wherein tasks are assigned priorities based on their attributes such as resource requirements and expected duration. While this approach demonstrates simplicity and efficiency, its optimality is context-dependent.

Genetic algorithms offer a distinct paradigm by employing evolutionary principles to evolve schedules that optimize specific objective functions, such as maximizing resource utilization and minimizing job wait times (Liao et al., 2023). Despite their effectiveness in tackling complex scheduling problems, genetic algorithms may necessitate a substantial number of iterations to converge to an optimal solution, as evidenced by prior studies (Tang et al., 2023; Al-Habob et al., 2020; Wang, Zhang, Liu, Zhao, et al., 2022a).

Algorithm	Type	Characteristics
Multilayer Perceptron (MLP)	Feedforward Neural Network	It uses backpropagation to train multiple layers of neurons for classification or regression tasks.
Convolutional Neural Network (CNN)	Feedforward Neural Network	It uses convolutional layers to extract spatial features from images, often used for image classification tasks.

Recurrent Neural Network (RNN)	Recurrent Neural Network	It uses feedback connections to maintain a "memory" of previous inputs and make predictions based on sequence data, often used for natural language processing tasks.
LongShort-Term Memory (LSTM)	Recurrent Neural Network	A type of RNN that includes a gating mechanism to selectively remember or forget previous inputs, often used for time series prediction and natural language processing tasks.
Autoencoder	Unsupervised Learning	It uses a neural network to compress and reconstruct input data, often used for dimensionality reduction and anomaly detection.
Generative Adversarial Network (GAN)	Unsupervised Learning	It uses two neural networks to generate synthetic data indistinguishable from accurate data, often used for image and text generation.
Deep Belief Network (DBN)	Unsupervised Learning	They comprise multiple layers of restricted Boltzmann machines (RBMs), used for unsupervised feature learning and generative tasks.
Reinforcement Learning	Reinforcement Learning	It uses trial-and-error learning to optimise a policy for an agent in an environment, often used for game-playing and robotics tasks.

Table I describes Gradient Boosting Machine (GBM) Ensemble Learning that Combines multiple decision trees to improve the performance of a model, often used for regression and classification tasks. Random Forest Ensemble Learning Combines various decision trees and selects features randomly to improve the performance of a model, often used for regression and classification tasks. K-Means Clustering Unsupervised Learning Partitions data into K clusters based on similarity, often used for data segmentation and anomaly detection tasks. Support Vector Machine (SVM) Supervised Learning Maximizes the margin between different classes to classify data, often used for classification and regression tasks. Naive Bayes Classifier Supervised Learning Uses Bayes' theorem to calculate the probability of each category based on the input features, often used for classification tasks. K-Nearest Neighbors (KNN) Supervised Learning Classifies data based on the K nearest data points in the feature space, often used for classification and regression tasks.

This paper aims to explore the utilization and efficacy of these RL algorithms and priority-based scheduling strategies in the context of load balancing within cloud computing. Through a comprehensive review and analysis, we seek to elucidate their strengths, limitations, and practical implications for optimizing cloud infrastructure.

1.1 Reinforcement Learning with Policy Gradients:
In scheduling jobs, the reinforcement learning approach employs a policy gradient algorithm, as outlined by Liu et al. (2019). This algorithm directly optimizes the procedure, mapping states to actions, through gradient ascent on the expected reward (see Fig. 1). While effective for learning intricate scheduling policies, this approach may necessitate a substantial amount of training data to converge to an optimal policy.

Ant Colony Optimization:
This strategy utilizes an algorithm to identify an optimal schedule by simulating the behavior of ants. The algorithm generates a population of "ants" that explore the search space by depositing pheromones, attracting other ants to promising solutions (Zhou et al., 2019; Wang, Zhang, Liu, Li et al., 2022). With the evaporation of pheromones over time, the algorithm converges to a solution with a high concentration of pheromones. While proficient in finding optimal solutions for complex scheduling problems, this approach may require numerous ants to adequately explore the search space (Wang, Zhang, Liu, Zhao et al., 2022b; Zhou et al., 2020).

1.2 Key Contributions of This Paper:
a) Employing Deep Q-learning, target networks, and involvement rerun, we integrate a task scheduling method aimed

at reducing computing resource consumption during offloading.
b) We compare and examine various machine learning-based solutions for diverse load-balancing tasks in data centers.
c) Implementation of an Actor-Critic-based Compute-Intensive Workload Allocation Scheme in a cloud environment, considering various parameters.
d) The paper outlines and underscores the challenges and future research directions pertinent to the present study.

1.3 Organization of the Paper:
Section 2 provides an overview of Cloud Computing technology coupled with reinforcement learning. Section 3 delves into the proposed work, offering a detailed description of the performance of the Actor-Critic-based Compute-Intensive Workload Allocation in reinforcement learning. Section 4 discusses numerous open challenges and avenues for future research. Finally, conclusions and avenues for future work are drawn.

Proposed Methodology

The proposed approach employs deep reinforcement learning to derive an optimal resource allocation policy based on real-time workload dynamics and resource availability within a cloud environment. A fundamental component of this approach is the utilization of an actor-critic deep reinforcement learning algorithm, comprising an actor responsible for suggesting resource allocation decisions and a critic tasked with evaluating the quality of these decisions (see Fig. 2). Both the actor and critic components are implemented as neural networks and are trained using a hybrid approach combining supervised and reinforcement learning techniques.

To evaluate the effectiveness of the proposed algorithm, simulations are conducted within a cloud data center environment. Results from these simulations demonstrate the superiority of the proposed approach over traditional resource allocation algorithms, particularly in terms of resource utilization and workload completion time.

The optimal Q-value for a given state-action pair (f_t, l_t) is determined by the Bellman equation for the optimal Q-value in reinforcement learning. This equation calculates the expected sum of the immediate reward (R_x_t+1) and the discounted value of the maximum Q-value of the subsequent state (f_t+1) and all possible actions (l_t+1) that can be taken from it. The discount factor (α) in the equation determines the weight assigned to future rewards, allowing for recursive computation of the optimal Q-value for each state-action pair.

The Bellman equation serves as a foundational principle in reinforcement learning, offering a systematic framework for determining optimal Q-values and guiding decision-making processes within the proposed resource allocation algorithm.
This represents the reward received at the current time step (t+1). The discount factor, α (alpha), plays a crucial role in determining the significance of future rewards relative to immediate rewards. With a value ranging between 0 and 1, α ensures that the agent considers both immediate rewards and the potential future rewards, accounting for uncertainty and potential delays in their receipt.

max_l_t+1 Q*(f_t+1, l_t+1) denotes the maximum Q-value for the subsequent state f_t+1 and all feasible actions l_t+1 that can be taken from it, as depicted in Fig. 3. This value signifies the anticipated total reward achievable by adhering to the optimal policy from the subsequent state and action onwards. The max operator selects the highest Q-value, corresponding to the optimal action according to the existing policy.

The Σ[...] operator, representing the expectation, computes the anticipated value of the expression within the brackets, given the current state-action pair (f_t, l_t).

In summary, the Bellman equation for the optimal Q-value articulates the anticipated total reward for a state-action pair, incorporating the immediate reward and the maximum expected future rewards from the subsequent state and all possible actions. This equation serves as a cornerstone in reinforcement learning, commonly utilized in various Q-learning algorithms to update Q-values and enhance the agent's policy iteratively.

Furthermore, the introduction of self-adaptive systems is discussed, which possess the capability to monitor their behavior and dynamically adjust to changing conditions to achieve specified performance objectives.

ALGORITHM 1: CALCULATION OF GRADIENT ACTOR-CRITIC RL ALGORITHM

Initialize actor network Va θ (s) and critic network Dπθ (o, p) with weights and
Initialize actors and critics learning rate γn and γm, and TD error discount factor β
for each training epoch n = 1, 2, ..., N do
    Receive initial state s1, where s1 = environmental. observe()
    for each episode l = 1, 2, ..., L do
Select action according to Su, where
        Action (at) = actor. choose action(Su)
Execute action at, receive reward Rt and next.

The algorithm proceeds through the following steps:

1. Initialization: Start by initializing the actor network, represented as πθ(s), and the critic network, denoted as Qπθ(s, a), with random weights and biases. Also, initialize the learning rates for the actor and critic (γa and γc respectively), along with the TD error discount factor, β.

2. Training Epoch: For each training epoch, repeat the subsequent steps:

a. Initial State: Begin by receiving the initial state, s1, by observing the environment.

b. Episode Loop: For each episode, repeat the following steps:

i. Action Selection: Select an action, at, based on the current state, st, using the actor policy.

ii. Execution: Execute the chosen action, at, in the environment, receiving a reward, rt, and transitioning to the next state, st+1.

iii. TD Error Calculation: Calculate the TD error in the critic, δπθ, using the current action, reward, and next state.

iv. Policy Gradient Calculation: Compute the policy gradient in the actor, ∇θJ(θ), using the advantage function. This function is derived from the TD error, reflecting the log probability rise of the current action in the current state.

v. State Update: Update the current state to the next one, st = st+1.

3. Termination: Conclude the training process.

In essence, this algorithm aims to train an actor-critic RL model capable of learning a policy that maximizes cumulative rewards across a sequence of states and actions in an environment. It leverages neural networks to approximate both the policy and value function. The TD error discount factor regulates the balance between immediate and future rewards, while the policy gradient is updated using the advantage function to integrate information on the current policy's effectiveness. Actor-critic algorithms, such as the one depicted in Figure 4, concurrently optimize a policy (the actor) and estimate its value (the critic).

In the context of compute-intensive load balancing in cloud computing, the objective is to allocate computational resources (e.g., VM instances) to tasks effectively, minimizing processing time and maximizing resource utilization while adhering to constraints.

Equations utilized in actor-critic algorithms for this purpose include:

1. Policy Updates Equation:
\(\forall \theta = \omega \sigma \forall_{\theta} \log \pi(l_k | k, \theta) \cdot g\)

This equation updates the actor's policy weights (θ) based on the advantage estimate (σ) and the log probability of selecting the action l_t in state k_t according to the policy π.

2. Value Function Update Equation:
\(\forall Z(l_t) = \beta(\delta_t - V(l_t))\)

This equation updates the critic's value function (Z) for a state l_t based on the advantage estimate (δ_t) and a learning rate (α).

Performance Matrix Setup in a Cloud Environment

Simulation Environment

To evaluate the effectiveness of the proposed method, we establish a simulation environment using CloudSim, a Java-based cloud simulation tool. CloudSim allows us to create and configure a realistic cloud infrastructure where our suggested deep reinforcement learning technique can be implemented and evaluated effectively.

Our implementation of the deep reinforcement learning technique is developed in Python and integrated into the CloudSim-generated cloud environment. This integration enables us to assess the performance of our method in a simulated but realistic cloud setting.

In setting up the simulation environment, we draw upon Google's task events dataset to define and generate user tasks. This dataset provides a comprehensive set of task events, allowing us to create diverse and representative user tasks for our simulations.

For the infrastructure setup, we deploy four Virtual Machines (VMs), each equipped with 16 GB of RAM and one terabyte of storage capacity. These VM configurations are chosen to mirror typical cloud environments and provide a suitable platform for evaluating our proposed method.

During the simulations, tasks are executed on the VMs, with each task requiring storage space ranging from 5 GB to 100 GB. This variability in storage requirements ensures that our simulations encompass a wide range of workload scenarios, allowing us to thoroughly evaluate the performance of our method across different task types and resource demands.

Algorithm	Response Time	CPU Utilisation	Throughput	Task Completion Time (ms)
GA	350.0	0.061	0.078	0.075

DSOS	352.1	0.052	0.065	0.062
MSDE	421.0	0.051	0.059	0.051
PSO	450.23	0.480	0.490	0.056
WOW	520.3	0.490	0.491	0.561
DQL	572.592	0.451	0.4386	0.495
ACD-RL	510	0.445	0.4275	0.470

Simulation Results

The performance of various resource allocation algorithms in a cloud computing environment is compared in Table II. This comparison encompasses four key metrics: response time, CPU utilization, throughput, and task completion time (measured in milliseconds). The algorithms under consideration include Genetic Algorithm (GA), Dynamic Search Optimization Strategy (DSOS), Modified Symbiotic Organism Search (MSDE), Particle Swarm Optimization (PSO), Whale Optimization Algorithm (WOA), Deep Q-Learning (DQL), and Actor-Critic Deep Reinforcement Learning (ACD-RL).

Upon analysis of the table data, it is observed that GA and DSOS exhibit the lowest response times, with GA achieving the fastest response time of 350.0 ms. In terms of CPU utilization, WOA demonstrates the highest value, reaching 0.490, while PSO and DQL achieve the highest throughput values of 0.490 and 0.4386, respectively. The task completion time is relatively low across most algorithms, with MSDE achieving the lowest value of 0.051 ms.

Relative to the other algorithms, ACD-RL performs reasonably well, with a response time of 510 ms, CPU utilization of 0.445, throughput of 0.4275, and task completion time of 0.470 ms.

Overall, Table II provides insights into the comparative performance of different resource allocation algorithms in a cloud computing environment, facilitating informed decision-making regarding algorithm selection for specific cloud computing tasks.

Conclusion and Future Scope

The proposed scheme is designed to optimize task execution time by efficiently allocating resources and distributing workloads across different servers. Our next step involves enhancing the model by incorporating a priority order mechanism, which will allow us to assign relative importance to tasks and adjust the state space, action space, and reward function accordingly.

Algorithms will be trained to learn scheduling policies that consider the priority order of tasks, allowing for the observation of how prioritizing high-priority tasks affects the rewards they receive. The performance of our proposed Actor-Critic Deep Reinforcement Learning (ACD-RL) agent surpasses that of six other algorithms, demonstrating its effectiveness in optimizing resource allocation.

In terms of system cost, the ACD-RL agent exhibits similar behavior to the Dynamic Queueing (DQL) and Particle Swarm Optimization (PSO) algorithms, as task distribution is evenly spread across all data. However, algorithms like Dynamic Search Optimization Strategy (DSOS) and Modified Symbiotic Organism Search (MSDE) could achieve better performance if their search spaces were reduced, thereby minimizing CPU usage and improving overall system efficiency.

Quantization levels affect performance, with higher quantization levels marginally improving throughput and task completion time, although not as significantly as the ACD-RL approach. ACD-RL excels in determining optimal actions by achieving a 23% improvement over previous studies in terms of indexing LB values, while maintaining a concise Task Completion Time. Additionally, ACD-RL ensures efficient CPU utilization, reducing it to 12% compared to other algorithms' potential 38%.

For future endeavors, we plan to explore the establishment of an edge cloud computing network system to facilitate collaborative computing tasks. Additionally, we aim to simplify the training process by adopting federated learning-based RL, which requires only live data flow to the data center for sharing model parameters, eliminating the need for local training data and reducing computational and communication complexities.

References

[1]. Kumar, B. K., Majumdar, A., Ismail, S. A., Dixit, R. R., Wahab, H., & Ahsan, M. H. (2023, November). Predictive Classification of Covid-19: Assessing the Impact of Digital Technologies. In *2023 7th International Conference on Electronics, Communication and Aerospace Technology (ICECA)* (pp. 1083-1091). IEEE.
Doi: https://doi.org/10.1109/TNNLS.2011.2179810

[2]. Schumaker, R. P., Veronin, M. A., Rohm, T., Boyett, M., & Dixit, R. R. (2021). A data driven approach to profile potential SARS-CoV-2 drug interactions using TylerADE. *Journal of International Technology and Information Management*, *30*(3), 108-142.
DOI: https://doi.org/10.58729/1941-6679.1504

[3]. Schumaker, R., Veronin, M., Rohm, T., Dixit, R., Aljawarneh, S., & Lara, J. (2021). An Analysis of Covid-19 Vaccine Allergic Reactions. *Journal of International Technology and Information Management*, *30*(4), 24-40. DOI: https://doi.org/10.58729/1941-6679.1521

[4]. Dixit, R. R., Schumaker, R. P., & Veronin, M. A. (2018). A Decision Tree Analysis of Opioid and Prescription Drug Interactions Leading to Death Using the FAERS Database. In *IIMA/ICITED Joint Conference 2018* (pp. 67-67). INTERNATIONAL INFORMATION MANAGEMENT ASSOCIATION.
https://doi.org/10.17613/1q3s-cc46

[5]. Veronin, M. A., Schumaker, R. P., Dixit, R. R., & Elath, H. (2019). Opioids and frequency counts in the US Food and Drug Administration Adverse Event Reporting System (FAERS) database: A quantitative view of the epidemic. *Drug, Healthcare and Patient Safety*, 65-70.
https://www.tandfonline.com/doi/full/10.2147/DHPS.S214771

[6]. Veronin, M. A., Schumaker, R. P., & Dixit, R. (2020). The irony of MedWatch and the FAERS database: an assessment of data input errors and potential consequences. *Journal of Pharmacy Technology*, *36*(4), 164-167.
https://doi.org/10.1177/8755122520928

[7]. Veronin, M. A., Schumaker, R. P., Dixit, R. R., Dhake, P., & Ogwo, M. (2020). A systematic approach to'cleaning'of drug name records data in the FAERS database: a case report. *International Journal of Big Data Management*, *1*(2), 105-118.
https://doi.org/10.1504/IJBDM.2020.112404

[8]. Schumaker, R. P., Veronin, M. A., & Dixit, R. R. (2022). Determining Mortality Likelihood of Opioid Drug Combinations using Decision Tree Analysis.
https://doi.org/10.21203/rs.3.rs-2340823/v1

[9]. Schumaker, R. P., Veronin, M. A., Dixit, R. R., Dhake, P., & Manson, D. (2017). Calculating a Severity Score of an Adverse Drug Event Using Machine Learning on the FAERS

Database. In *IIMA/ICITED UWS Joint Conference* (pp. 20-30). INTERNATIONAL INFORMATION MANAGEMENT ASSOCIATION.

[10]. Dixit, R. R. (2018). Factors Influencing Healthtech Literacy: An Empirical Analysis of Socioeconomic, Demographic, Technological, and Health-Related Variables. *Applied Research in Artificial Intelligence and Cloud Computing*, *1*(1), 23-37.

[11]. Dixit, R. R. (2022). Predicting Fetal Health using Cardiotocograms: A Machine Learning Approach. *Journal of Advanced Analytics in Healthcare Management*, *6*(1), 43-57. Retrieved from https://research.tensorgate.org/index.php/JAAHM/article/view/38

[12]. Dixit, R. R. (2021). Risk Assessment for Hospital Readmissions: Insights from Machine Learning Algorithms. *Sage Science Review of Applied Machine Learning*, *4*(2), 1-15. Retrieved from https://journals.sagescience.org/index.php/ssraml/article/view/68

[13]. Ravi, K. C., Dixit, R. R., Singh, S., Gopatoti, A., & Yadav, A. S. (2023, November). AI-Powered Pancreas Navigator: Delving into the Depths of Early Pancreatic Cancer Diagnosis using Advanced Deep Learning Techniques. In *2023 9th International Conference on Smart Structures and Systems (ICSSS)* (pp. 1-6). IEEE. https://doi.org/10.1109/ICSSS58085.2023.10407836

[14]. Khan, M. S., Dixit, R. R., Majumdar, A., Koti, V. M., Bhushan, S., & Yadav, V. (2023, November). Improving Multi-Organ Cancer Diagnosis through a Machine Learning Ensemble Approach. In *2023 7th International Conference on Electronics, Communication and Aerospace Technology (ICECA)* (pp. 1075-1082). IEEE. https://doi.org/10.1109/ICECA58529.2023.10394923

View publication stats