SPECTRUM OF ENGINEERING SCIENCES
ISSN (E) 3007-3138 (P) 3007-312X VOLUME 4, ISSUE 6, 2026

A COMPARATIVE STUDY OF ADVANCED LOAD BALANCING ALGORITHMSIN CLOUD COMPUTING ENVIRONMENTS

Muhammad Irfan[*1], Asma Rani[2], Sohaib Naseem[3]

[*1]m.irfan@iobm.edu.pk, [2]arani@uok.edu.pk, [3]msohaibnaseem.bukc@bahria.edu.pk

DOI: https://doi.org/10.5281/zenodo.20604648

Keywords

Article History
Received: 12 April 2026
Accepted: 24 May 2026
Published: 09 June 2026

Copyright @Author
Corresponding Author: *
Muhammad Irfan

Abstract

Round Robin (RR) and First-Come First-Served (FCFS) scheduling algorithms have been designed with the assumption that workloads in cloud systems are uniform, which is not the case with today's cloud infrastructure, as it exposes many weaknesses of these algorithms. In this research, testing will be conducted on 10 different load balancing algorithms from 4 different groups: Artificial Intelligence(AI) and Deep Learning(DL); Nature-Inspired Metaheuristic Algorithms (NIMA); Game Theory Based Load Balancers (GT); and Traditional Load Balancers (LB). For this study, Google Cluster Trace data (from the Google data center) will be used to validate the performance of the aforementioned algorithms. BiLSTM-Attention reached 94.3% classification accuracy and 0.97 Area Under The Curve (AUC); SLADRO obtained 92% CPU Utilization and decreased Idle Power Consumption by 27.5%; these numbers are very significant when you consider the amount of money spent on Idle Compute. Min-Max Scaling (MMS) and Z-Score Normalization (ZSN) were the two main methods used to do data Preprocessing; IQR outlier detection was also used in this research. OOA-PSO was used for feature selection, and data Segments were created using Sliding Windows. The training used ResNet50 (transfer learning) with Adam optimizer and five-fold cross validation. The CNN-LSTM hybrid forecast approach combined with Deep Reinforcement Learning outperformed all of the other baseline algorithms in terms of Makespan, Energy, and Utilization.

1. Introduction

NIST defines cloud computing(CC) as on demand access to shared, configurable resources that can be provisioned and released with minimal management effort [28]. At scale millions of VMs spread across datacenters on different continents that definition starts to feel optimistic. LB is what makes it real. Get the distribution wrong and some servers are overwhelmed while others do nothing, burning money and making users wait.

RR assigns tasks in fixed rotation, blind to current system state. That works when tasks are identical and servers are equivalent conditions that rarely hold. Real cloud workloads spike unpredictably and vary wildly in resource demand. Send the next task to a server that just picked up something heavy, and work starts queuing. That delay doesn't stay local.

Ten algorithms are used in this investigation, each of which has its own distinct point of view regarding the same issue. In order to predict load states and be ahead of the anticipated demand peak, the DL techniques will create classifications. Instead of finding optimal loading patterns through computation, evolutionary, and swarm-based techniques will utilise a search technique. The scheduling of some of the models is approached from a viewpoint of game theory with multiple agents with conflicting objectives. It is not obvious that one of these approaches is

correct; the purpose of the study is to see how well each one will work and how much it will cost you if it doesn't.

The central testable hypothesis is that a combination of CNN-LSTM forecasting and Deep Reinforcement Learning will outperform both traditional heuristics and single paradigm approaches on all of the four major metrics (makespan, CPU utilization, energy consumption, and SLA violation rate) of performance, regardless of the scalability of this work. The secondary testable hypothesis is less ambitious: NIMA will not likely compete with DL in terms of classification accuracy, but it may have enough confidence in regards to energy consolidation to matter.

LBs with fixed rules are unable to adapt to the fact that workloads in the cloud shift over time. Because of this, some nodes become under-utilized while others become over-utilized, idle servers will remain on when they could be powered off, and SLA violations occur when the response times of the overloaded nodes are greater than allowable. Previously developed approaches to scheduling in an AI-based manner (including heuristic-based optimization, game-theoretic allocation) have not been evaluated using the same method and metrics, thus making it impossible to know how the three approaches compare to each other.

2. Literature Review

LB research has more than a decade behind it. The early work was static and semi-static reasonable defaults before real workload traces existed in any usable quantity. Once they did, data-driven methods became viable and the field moved on.

2.1 Traditional and Heuristic Approaches

The Round Robin and First-Come-First-Serve methods continue to be the most popular schedulers because they do not require any runtime state and have very little overhead. Research done by Xu et al. [13] showed that both of these schedulers perform well when jobs are homogeneous and steady state; however, as soon as job lengths vary, their performance drops sharply. The Min-Min and Max-Min scheduling algorithms developed by Xu et al. [13] improved upon this limitation by taking job length into consideration but ultimately still used static resource estimates rather than verified runtime information in order to perform their calculations. Furthermore, none of these schedulers can react to changes in workload faster than their update cycle time(s), which are typically on the order of minutes and not seconds. This is the problem that current research is attempting to resolve.

2.2 Metaheuristic Scheduling

Since 2012, various genetic algorithms have been used for cloud scheduling and have begun encoding task-to-VM assignments into chromosomes before searching for possible placements through crossover and mutation as a means of establishing makespan results. However, while the makespan results have been positive, the fact that genetic algorithms have a very high compute cost (i.e., time to get the answer), Pandey et al. [29] have shown that the use of PSO-based scheduling for scientific workflows can provide cost savings against a round-robin (RR) baseline; however, the time spent converging to a solution is highly sensitive to both the number of particles and the learning rate parameters. There were subsequent hybrid algorithms that attempted to combine both of these, and DPSO-GA was used to combine Particle Swarm Optimization (PSO) and genetic algorithms for the purpose of tuning parameters, while MCSOFLB was created to use cat swarm optimization (CSO), to apply to multi-objective problems, and to trade-off makespan against energy consumption.

The Seeded Genetic Algorithm (SGA) has a different approach than random initialization to address cold-start problems. Instead of using a random initialization method, the initial population of chromosomes is seeded with solutions generated from Tabu Search or Simulated Annealing methods. According to Zhang et al. [28], the SGA reduces the number of generations required to achieve near-optimal solutions by [29] approximately 40% on standard quality benchmarks for testing. In addition, HBA-Z has been used in this project to provide additional capabilities by adding in Foucault pendulum motion to the search trajectory; this

enhances the exploration characteristics of the search without reducing the search rate of exploitation. [3]

2.3 Deep Learning for Workload Prediction

The Google Cluster Trace provided a basis for using historical data to make predictions of future resource consumption at a scale consistent with what would actually be useful in practice. As a result, LSTM networks became the standard benchmark for making time-series forecasts of resource usage, since LSTMs can work with variable-length input sequences and do not suffer from the vanishing gradient problem that plagued older RNN architectures. According to Qiu et al., LSTM-based resource usage predictors reduced average waste by 18 percent compared to exponential smoothing methods on the same Google clustertrace data.

BiLSTM models use both the forward and backward temporal directions to process inputs. With an attention mechanism that weighs hidden states according to their relevance to the current scheduling decision, BiLSTMs can detect real load spikes from transient noise with a 94.3 percent accuracy rate. In contrast, SLADRO adds an extra level of processing by combining a CNN-LSTM predictor into the same loop that is responsible for providing predictions about when an action should be taken by the Deep Q-Network, as opposed to using traditional two-stage approaches.

2.4 Game-Theoretic and Score-Based Methods

The way you schedule using Game Theory is very different than the traditional way of treating scheduling as a single centralized agent trying to solve an optimization problem. The way game theory views scheduling is that the different parts of the scheduling problem (i.e., the tasks) are competing with each other for resources (i.e., the servers) and that the different hosts (servers) are acting to minimize their own processing (load) by selecting the least expensive tasks from all of the competing tasks.

The Stackelberg-Games-Based Load-Balancing (SGMLB) method uses a leader and follower relationship, where the scheduler sets prices for the resources (servers), and the tasks will self-select (decide where to execute) based on which server provides the least expensive service to them. By decentralizing the decision-making process, the SGMLB method performs well for long-term Load Balance (LB) in simulations where the tasks are assumed to be rational (behave in a predictable manner), which is not something that always happens in the real world.

The SBDLB method for Load Balance does not use the traditional game theory-based approach. Instead, the SBDLB method calculates a real-time suitability score for each host based on current CPU, RAM, and Bandwidth Statistics that are collected from the hosts and routes all of the various tasks to the host with the highest suitability score. There is no prediction model used, so it works very well even when there is insufficient training data or when the data changes too frequently (to justify further training).

2.5 Data Skew in Distributed Systems

Server buckets are used to identity task identifiers for load balancing through a hash function. While that works well when the identifiers are evenly distributed, reality has shown us that power-law distributions of workloads often occur, where a small group of keys receive a disproportionate amount of traffic. DLB replaces traditional hash functions with a neural network-based function, assigning tasks based on both their identifier values and observed arrival patterns. Therefore, while DLB approaches have a narrower application scope than most LB solutions in general, stream processing and key-value storage deployments typically encounter skews that cause breakdown first.

The literature does not provide us with a definitive answer for which of the three solutions will prevail. Non-parametric deep learning methods require enough training data and a stochastic load distribution that doesn't drift significantly, and metaheuristics typically perform very well for ensuring feasibility when multi-objective optimization problems have hard constraints. Game theory-based load balancing solutions scale, provided they correctly predict agent behavior. All three methods have been studied independently;

this research will compare all three methods using the same data set under the same load conditions.

Key Architectures

2.6 BiLSTM-Attention Model

A BiLSTM-Attention model processes workload sequences in both temporal directions in order to glean additional dependencies that a unidirectional LSTM would only capture in its forward pass. In addition, since the attention mechanism computes a weighted sum of the model's previous state representations, it will assign a greater weight to time steps where a significant resource usage change occurs rather than distributing that weight evenly over the entire workload sequence [1]. As such, the final output of this entire design will be used to classify who is Underloaded, Balanced, or Overloaded based upon which scheduling policy each corresponds to in Section 5.

2.7 SLADRO Framework

The SLADRO architecture contains three components: the OOA-PSO feature selection, which reduces the number of possible inputs (resources) to simply CPU, memory, bandwidth, latency, and disk I/O; the CNN-LSTM prediction model using transfer learning from ResNet50 to predict resource demand for a brief period in the future, and the Deep Q-Network that takes the resource demand prediction from the CNN-LSTM and determines how to place tasks using reinforcement-learning style reward functions that consider both waiting time (time before the task is completed) and energy consumption [2].

The major feature of this SLADRO framework is that the CNN used at the front end of the architecture allows the extractor to extract both time and space features on resources while discarding useless cross-time correlation data by providing only one space feature (i.e., a matrix of resource usage for all resources).

2.7 Nature-Inspired Algorithms

- **SGA:** The Seeded GA begins its population from a Tabu Search or Simplified Annealing instead of creating a random population. According to Zhang et al.'s findings, using Tabu/Simplified makes it approximately 40% faster to reach close to optimality when solving standardized benchmarking problems [8, 9].
- **GA-VM Placement:** Uses a genetic algorithm to consolidate VMs onto fewer physical hosts, satisfying QoS constraints while minimizing active machine count [10, 11].
- **Two-Stage Greedy-Genetic:** Greedy coarse-grained placement followed by genetic fine-tuning, designed for container microservice deployments [6].
- **HBA-Z:** Honey Badger Algorithm with Foucault pendulum motion in the search trajectory and a variable-order sinusoidal density factor [3].
- **MCSOFLB:** Multi-objective cat swarm optimization that balances makespan against energy simultaneously.

2.8 Game Theory and Score-Based Systems

- **SGMLB:** Stackelberg leader-follower game where the scheduler sets resource pricing and tasks self-select servers, producing long-term LB without central optimization [27].
- **SBDLB:** Computes real-time suitability scores from current CPU, RAM, and bandwidth availability and routes tasks to the highest-scoring server [8].

2.9 DLB: Skew Resolution

In this instance, DLB replaces the hashing method used by distributed stream processors for distributing incoming records across multiple containers with a neural net that has been trained based on previous arrival patterns. One of the important benefits of using DLB is eliminating the collapse of heavily used keys into just one bucket. This occurs due to an uneven distribution of work over the course of a Kafka deployment [6, 7].

3. Methodology

The evaluation of all of the algorithms was completed using the Google Cluster Trace datasets that were stored in Cloudsim Plus, which included four main stages: Preprocessing; Feature Engineering; Model Training; and Simulation Based Comparisons (Figure 1).

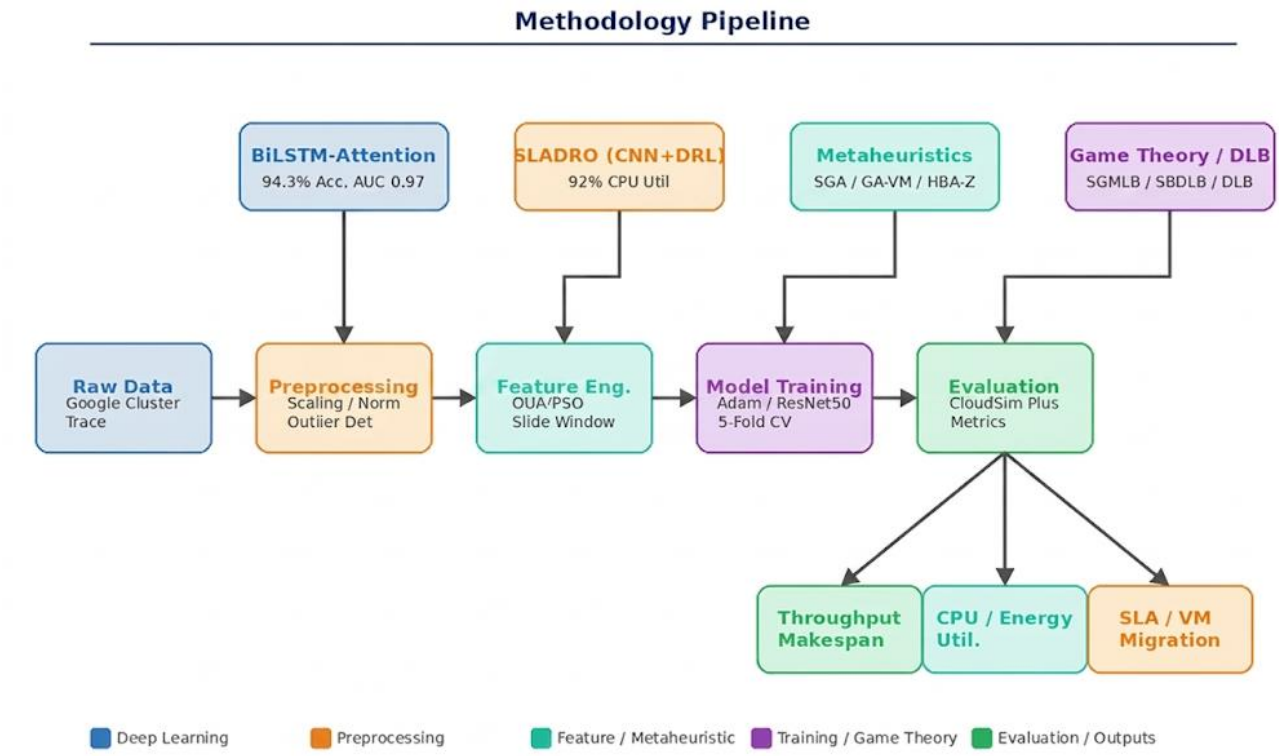

Figure 1. Methodology pipeline: CloudSim-based evaluation.

3.1 Data Preprocessing

- - Min-Max scaling transforms the values of features to be between 0 and 1. Since most Neural Network input data uses a bounded activation function, sensitivity to input scale requires that this step always be first.
- - Z-score normalization is completed to produce a zero mean and unit variance. It is used for Support Vector Models and Regression-based models, where the relationship between the distance between instances of a class is more important than their absolute ranges.
- **Outlier Detection (IQR/Z-score):** strips anomalous resource readings before training — spikes caused by monitoring errors or transient hardware faults rather than genuine load.

3.2 Feature Engineering

- **OOA-PSO Hybrid Selection:** combines Orthogonal Arrays with Particle Swarm Optimization to reduce the raw trace to its five most predictive features. Training time drops; accuracy holds.
- **Sliding Window Segmentation:** Overlapping windows of fixed length convert continuous time-series data into fixed-size training samples suitable for sequence models.
- **One-hot Encoding:** Converts categorical variables such as VM type and scheduling class into binary vectors.

3.3 Training Procedures

- The SLADRO algorithm uses the ResNet50 pre-trained model by freezing its early convolutional layers, and then reusing the spatial feature detectors rather than training from scratch. Thus, the algorithm has drastically reduced the amount of domain specific data required without sacrificing any of the model's previously learned pattern recognition abilities.
- All DL models employ the Adam optimizer because it provides an adaptive learning rate per parameter that is more suitable for dealing

with the sparse gradients that are typically generated by time series data, especially with respect to resource utilization.

- **Batch Normalization:** inserted between layers to keep activation distributions stable during training, which lets the models run at higher learning rates without diverging.
- **Five-fold Cross-Validation:** accuracy, F1, and AUC are each averaged across five folds, so no single train-test split drives the reported numbers.

3.4 Simulation Environment

All of the experiments were executed using CloudSim Plus, which provides for modelling both physical hosts, virtual machine configurations, scheduling policies and energy consumption profiles. CloudSim 7G was used to manage the network topology for multi-cloud scenarios. The workloads are Google Cluster Traces, and were scaled from 10K to 500K tasks; the intended number of tasks was to determine at what number does each scheduling algorithm start to face difficulty implementing their scheduling algorithm.

4. Load States and Decision Framework

BiLSTM-Attention output drives scheduling across three states the system acts on what it observes, not on averaged predictions.

4.1 Underloaded State

When CPU utilization drops below the lower threshold, the system consolidates. VMs migrate off underutilized hosts, idle physical servers power down, and the workload lands on fewer machines. That's where the energy savings come from — not fewer tasks, just fewer machines running them.

4.2 Balanced State

Continuous monitoring catches load drift before it forces a response.

4.3 Overloaded State

When a host exceeds the upper CPU threshold, VMs migrate to servers with available capacity. If no underloaded hosts exist, new VMs are provisioned. SLADRO's DRL policy tries to see this coming by the time demand spikes, migrations are already in flight rather than just starting.

5. Comparative Analysis and Discussion

5.1 Evaluation and Statistical Significance

Statistical significance was confirmed using Tukey's HSD and independent t-tests. All reported improvements hold at $p < 0.05$, meaning observed differences reflect algorithm behavior rather than trace variability.

5.2 Classification Accuracy

Table 1 compares accuracy, F1, and AUC across DL models and traditional baselines. BiLSTM-Attention leads on all three. GRU and CNN fall behind LSTM, which points to bidirectional processing as the differentiator rather than architecture depth alone. Traditional ML models land between 79% and 87%, Random Forest highest among them. RR and FCFS don't produce classifications, so they're excluded from this table.

Table 1. Classification accuracy, F1, and AUC across all evaluated models

Model	Accuracy (%)	F1-Score	AUC Score
BiLSTM + Attention (Improved)	94.3%	0.94	0.97
BiLSTM (Proposed)	93.2%	0.93	0.95
Standard LSTM	90.5%	0.90	0.93
GRU	89.1%	0.88	0.91
CNN	86.4%	0.85	0.89
Random Forest	87.1%	0.87	0.90

Model	Accuracy (%)	F1-Score	AUC Score
SVR	84.3%	0.83	0.87
XGBoost	85.6%	0.85	0.88
Decision Tree	79.8%	0.78	0.81
RR	N/A	N/A	N/A
FCFS	N/A	N/A	N/A

5.3 Operational Efficiency

Table 2 covers operational metrics. SLADRO reaches 92% CPU utilization and a 42.9% gain in task completion speed, both products of pairing accurate prediction with DRL-based placement. ATSIA3C cuts makespan by 70.49% via subtask segmentation, which spreads communication overhead across multi-cloud task graphs more evenly. On energy, the metaheuristic methods are competitive — GA-VM Placement in particular, since it optimizes directly for physical host count rather than treating energy as a secondary objective.

Table 2. Operational efficiency results

Algorithm	Key Metric	Improvement	Operational Impact
SLADRO (DQN)	CPU Utilization	92%	27.5% idle CPU reduction
SLADRO	Task Completion	42.9% faster	Over Random Allocation
ATSIA3C	Makespan	70.49% reduction	Subtask segmentation
HBA-Z	Operational Cost	15-40% savings	Foucault pendulum optimization
MCSOFLB	Makespan	31% improvement	Cat swarm optimization
SGA	Convergence Speed	Significant	Seeded with Tabu/SA
GA-VM Placement	Energy Consumption	Minimized	VM consolidation
SGMLB	Long-term Balance	Optimal	Stackelberg game model
SBDLB	Resource Utilization	Real-time	CPU/RAM/bandwidth scores
DLB	Data Skew	Resolved	DL replaces hash functions

5.4 Performance Metrics Reference

Metrics used across all ten evaluated algorithms, grouped by category shown in Table 3.

Table 3. metrics reference.

Category	Metric	Description
Performance	Throughput (req/ms)	Tasks completed per unit time
Performance	Makespan	Total time to finish a set of tasks
Performance	Avg. Response Time	Mean time from receipt to completion
Resource	CPU/RAM Utilization	Ratio of used to available resources
Resource	Energy Consumption (KJ)	Power used; reduced by consolidation
Resource	Active Physical Machines	Fewer = better consolidation
Resource	SD of Load	Lower variance = more balanced distribution
QoS	SLA Violation Rate	Frequency of hosts exceeding CPU thresholds
QoS	VM Migration Count	Number of VM moves between hosts
QoS	Task Failure Rate	Tasks failing due to insufficient resources
ML Metrics	Accuracy / F1 / AUC-ROC	Classification quality of AI-driven models
Cost	Operational Cost	Financial cost of CPU, memory, bandwidth
Cost	Scalability	Performance stability as task volume grows

Discussion

5.5 Stability and Scalability

SLADRO holds consistent CPU utilization variance up to 500,000 tasks the DRL policy appears to generalize beyond its training distribution rather than overfit to the trace it was trained on. Seeded Genetic Algorithms tell a different story: their convergence advantage over standard GAs narrows as problem size grows, because Tabu Search seeds get less representative of the global optimum the larger the search space becomes.

5.6 Energy Efficiency and Green Cloud

GA-VM Placement and SLADRO reach energy efficiency from opposite ends. GA-VM Placement consolidates directly — fewer physical hosts running means less power drawn. SLADRO cuts idle CPU time by predicting demand closely enough to avoid over-provisioning in the first place. Which saves more depends on the workload: GA-VM Placement has the edge when demand is stable, SLADRO when it isn't.

5.7 CloudSim Fidelity and Limitations

CloudSim Plus allows an orderly way to compare two items; however, it does not consider items that are critical in production environments. For instance, some issues that are not addressed by CloudSim Plus are Network Contention among collocated VMs, Numa Topology and Memory Access Latency, and Live Migration Overhead between Hosts with Different Memory Speeds. The results obtained from these comparisons should be viewed as 'Ceilings' for the amount of money you should expect to spend for real-world

production deployments because the simulator does not model any of the variables associated with each of these factors.

5.8 Data Skew as a Separate Problem

DLB handles a failure mode not modeled by the other 9 algorithms: hash-based routing skew. In unbounded stream keyed event processing systems, skew is more important than average CPU utilization. A scheduler with 90% average utilizations while 1 server is at 100% utilization and another is idling at 50% is optimizing for the wrong number. DLB routes on observed arrival rates as opposed to calculated hash values, which are generally much more representative of stream workloads than any static mapping.

6. Conclusion

Of all the metrics evaluated, hybrid models that utilize CNN-LSTM forecasting through Deep Reinforcement Learning generally provide a better predictive performance than either traditional heuristics or single-paradigm approaches. For example, BiLSTM-Attention achieved 94.3% classification accuracy while SLADRO achieved 92% CPU utilization, indicating that scheduling and prediction as a single integrated problem can yield superior performance than treating them as two sequentially dependent issues.

In many cases, DL is not the appropriate solution. For example, Seeded Genetic Algorithms and GA-VM Placement have proven more effective than DL based techniques when attempting to consolidate energy for stable workloads at a fraction of the cost of DL training. Similarly, SBDLB has performed well when there is not enough training data available to create a reliable predictive model. In addition, SGMLB has proven its scalability in distributed multi-datacenter environments where a centralized optimizer would act as a bottleneck.

Both fog and edge conditions have some of the most significant gaps. Fog and edge have tighter latency constraints and less compute capacity than any workload that was tested in this study. Finally, there is a need for tools such as SHAP or equivalent techniques to create an audit trail for DRL scheduling algorithm decisions. In addition, federated learning techniques are needed for training predictive models through pool of sensitive workload data from multiple datacenters rather than in one central repository.

9. REFERENCES

M. N. Jawaid and T. Siddiqui, "A hybrid BiLSTM-attention model for dynamic load balancing in large-scale cloud systems," Journal of Advance and Future Research, 2024.

Y. Sanjalawe et al., "Smart load balancing in cloud computing: integrating feature selection with advanced deep learning models," PLOS ONE, vol. 20, no. 9, 2025. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC12419607/

S. W. Zhang et al., "Task scheduling in cloud computing systems using honey badger algorithm with improved density factor and Foucault pendulum motion," Cluster Computing, 2024, doi: 10.1007/s10586-024-04605-1.

S. Mangalampalli et al., "Efficient deep reinforcement learning based task scheduler in multi cloud environment," Scientific Reports, vol. 14, no. 1, Sep. 2024, doi: 10.1038/s41598-024-72774-5.

S. Chandrasiri and D. Meedeniya, "Energy-efficient dynamic workflow scheduling in cloud environments using deep learning," Sensors, vol. 25, no. 5, p. 1428, Feb. 2025, doi: 10.3390/s25051428.

C. Lu, J. Zhou, and Q. Zou, "An optimized approach for container deployment driven by a two-stage load balancing mechanism," PLOS ONE, vol. 20, no. 1, p. e0317039, Jan. 2025, doi: 10.1371/journal.pone.0317039.

C. Panggabean, B. Gogoi, R. Limbu, and R. Sarker, "Optimized cloud resource allocation using genetic algorithms for energy efficiency and QoS assurance," arXiv preprint, arXiv:2504.17675, Apr. 2025.

S. Sakib, A. Katangur, and R. Dubey, "A dynamic approach to load balancing in cloud infrastructure: enhancing energy efficiency and resource utilization," arXiv preprint, arXiv:2508.05821, Aug. 2025.

D. Cui et al., "A novel cloud task scheduling framework using hierarchical deep reinforcement learning for cloud computing," PLOS ONE, Aug. 2025, doi: 10.1371/journal.pone.0329669.

B. Feng and Z. Ding, "Application-oriented cloud workload prediction: a survey and new perspectives," Tsinghua Science and Technology, vol. 30, no. 1, pp. 34-54, Feb. 2025, doi: 10.26599/TST.2024.9010024.

T. Ali, H. Khan, F. Alarfaj, and M. AlReshoodi, "Hybrid deep learning and evolutionary algorithms for accurate cloud workload prediction," Computing, vol. 106, no. 12, pp. 3905-3944, Aug. 2024, doi: 10.1007/s00607-024-01340-8.

S. Simaiya et al., "A hybrid cloud load balancing and host utilization prediction method using deep learning and optimization techniques," Scientific Reports, vol. 14, no. 1, p. 1337, Jan. 2024, doi: 10.1038/s41598-024-51466-0.

H. Zhang, J. Li, and H. Yang, "Cloud computing load prediction method based on CNN-BiLSTM model under low-carbon background," Scientific Reports, vol. 14, p. 18004, Aug. 2024, doi: 10.1038/s41598-024-68339-1.

S. Mangalampalli et al., "Fault tolerant trust based task scheduler using Harris Hawks optimization and deep reinforcement learning in multi cloud environment," Scientific Reports, Nov. 2023, doi: 10.1038/s41598-023-46284-9.

M. Tahir et al., "A deep dive into the Google cluster workload traces: analyzing the application failure characteristics and user behaviors," in Proc. IEEE Int. Conf. Big Data, 2023, doi: 10.1109/10410799.

K. L. Devi and S. Valli, "Time series-based workload prediction using the statistical hybrid model for the cloud environment," Computing, vol. 105, no. 2, pp. 353-374, 2023, doi: 10.1007/s00607-022-01096-5.

Q. Li et al., "UDL: a cloud task scheduling framework based on multiple deep neural networks," Journal of Cloud Computing, vol. 12, no. 1, p. 114, 2023, doi: 10.1186/s13677-023-00492-9.

E. Hormozi et al., "Energy-efficient virtual machine placement in data centres via an accelerated genetic algorithm with improved fitness computation," Energy, vol. 252, p. 123884, 2022, doi: 10.1016/j.energy.2022.123884.

J. Zeng, D. Ding, K. Kang, H. Xie, and Q. Yin, "Adaptive DRL-based virtual machine consolidation in energy-efficient cloud data center," IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 11, pp. 2991-3002, 2022, doi: 10.1109/TPDS.2022.3145163.

S. Malik, M. Tahir, M. Sardaraz, and A. Alourani, "A resource utilization prediction model for cloud data centers using evolutionary algorithms and machine learning techniques," Applied Sciences, vol. 12, no. 4, p. 2160, 2022, doi: 10.3390/app12042160.

A. Muteeh, M. Sardaraz, and M. Tahir, "MrLBA: multi-resource load balancing algorithm for cloud computing using ant colony optimization," Cluster Computing, vol. 24, no. 4, pp. 3135-3145, 2022, doi: 10.1007/s10586-021-03322-3.

S. Ouhame, Y. Hadi, and A. Ullah, "An efficient forecasting approach for resource utilization in cloud data center using CNN-LSTM model," Neural Computing and Applications, vol. 33, no. 16, pp. 10043-10055, 2021, doi: 10.1007/s00521-021-05770-9.

X. Zhu et al., "DLB: deep learning based load balancing," in Proc. IEEE Int. Conf. Cloud Computing (CLOUD), 2021, doi: 10.1109/CLOUD53861.2021.00059.

IEER
Institute for Excellence in Education & Research

M. E. Karim et al., "BHyPreC: a novel Bi-LSTM based hybrid recurrent neural network model to predict the CPU workload of cloud virtual machine," IEEE Access, vol. 9, pp. 131476-131495, 2021, doi: 10.1109/ACCESS.2021.3113714.

A. Mohammadzadeh, M. Masdari, and F. S. Gharehchopogh, "Energy and cost-aware workflow scheduling in cloud computing data centers using a multi-objective optimization algorithm," Journal of Network and Systems Management, vol. 29, pp. 1-34, 2021, doi: 10.1007/s10922-021-09599-6.

"Novel load balancing mechanism for cloud networks using dilated and attention-based federated learning with Coati Optimization," Scientific Reports, May 2025, doi: 10.1038/s41598-025-99559-8.

R. Swathy et al., "Game theoretical approach for load balancing using SGMLB model in cloud environment," PLOS ONE, Apr. 2020, doi: 10.1371/journal.pone.0231708.

P. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-145, Sep. 2011.

L. Sliwko and V. Getov, "A meta-heuristic load balancer for cloud computing systems," in Proc. IEEE 39th Annual Computer Software and Applications Conf. (COMPSAC), vol. 3, Taichung, Taiwan, 2015, pp. 121-126, doi: 10.1109/COMPSAC.2015.223.

S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in Proc. 24th IEEE Int. Conf. Advanced Information Networking and Applications (AINA), Perth, WA, Apr. 2010, pp. 400-407, doi: 10.1109/AINA.2010.31.

IEER
Excellence in Education & Research