National College of Ireland

Task Scheduling in Cloud Datacenters: Comparison of Heuristic, Metaheuristic, and Machine Learning Approaches

MSc Research Project
Cloud Computing

Amit Kumar Chhabiraj Yadav
Student ID: 23294795

School of Computing
National College of Ireland

Supervisor: Prof. Aqeel Kazmi

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Amit Kumar Chhabiraj Yadav
Student ID:	23294795
Programme:	Cloud Computing
Year:	2025
Module:	MSc Research Project
Supervisor:	Prof. Aqeel Kazmi
Submission Due Date:	11/12/2025
Project Title:	Task Scheduling in Cloud Datacenters: Comparison of Heuristic, Metaheuristic, and Machine Learning Approaches
Word Count:	XXX
Page Count:	21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Amit kumar Yadav
Date:	28th January 2026

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Task Scheduling in Cloud Datacenters: Comparison of Heuristic, Metaheuristic, and Machine Learning Approaches

Amit Kumar Chhabiraj Yadav
23294795

Abstract

Cloud datacenters are under pressure to reduce energy consumption and maintain high performance. As a result, task scheduling remains a major challenge for scheduling algorithms such as traditional heuristic algorithms, which struggle to handle workload diversity, task to VM allocation, and bursty task arrival patterns. Although heuristic, metaheuristic, and machine-learning schedulers have been studied individually, few works have compared all three paradigms under a common experimental setup. But this project addresses this gap by evaluating seven algorithms: heuristic (Round Robin, Min-Min, Max-Min), metaheuristic (GA, PSO, ACO), and a Deep Q-Network (DQN) within a CloudSim Plus environment, which work across six realistic workload scenarios. A total of 210 simulations were run to record energy consumption, makespan, latency, and throughput for all scenarios, and their results show DQN achieves the lowest mean energy usage in five scenarios (496.67 Wh) and outperforms Min-Min by 26% and Round Robin by 49%, while ACO performs best under highly bursty workloads. These findings demonstrate that reinforcement learning is a strong, energy efficient scheduling approach, though its sensitivity to extreme workload variability remains a limitation. In practice, up to 6.3 kWh/day of such RL-based optimization might be saved in a mid-sized datacenter, assuming that there are significant benefits to operations and the environment.

1 Introduction

1.1 Background

Cloud computing has emerged as one of the main technologies for delivering on-demand computing services to organizations. Hyperscale data centres with hundreds of thousands of servers are operated by large cloud providers such as Amazon Web Services, Google Cloud Platform, and Microsoft Azure. These data centres are paving the way for global digital expansion, but they also bring serious operational and environmental concerns. It has been reported that data centres use 200-250 TWh of power every year, which makes up about 1.5 per cent of the electricity consumption globally, and energy expenses take up 30-50 per cent of the total operational expensesKoomey et al. (2011). One of the key elements that affects this energy usage is task scheduling, which is the process by which incoming tasks (cloudlets) are distributed to the virtual machines (VMs) on

the physical hosts. Poor scheduling leads to congested VMs, idle hosts and wastage of resources. Actual production loads make this issue even more complicated: empirical evidence Reiss et al. (2011) indicates that cloud workloads are very skewed and bursty, not uniform, and simple or fixed-point scheduling methods are not appropriate in the present-day datacenters. This generates an urgent demand of energy conscious scheduling policies which can deal with the heterogeneous and unpredictable workloads and still ensure performance efficiency.

1.2 Motivation

The energy efficiency of cloud data centers has now become one of the economic and environmental priorities. A minor change in scheduling efficiency can result in massive cost savings and carbon emission reductions when implemented on a large scale. Most existing methods, however, consider a single performance metric, such as throughput, response time, or makespan, with energy consumption treated as auxiliary. Simultaneously, the scheduling research environment is very fragmented. Heuristic techniques are simple but frequently inefficient; metaheuristics offer better optimization, but these methods are very computationally intensive, and no work has been done to investigate them under realistic workloads. Machine learning–based techniques are promising, but there has not been much evaluation of these methods on realistic optimization problems in the data centre. Moreover, results in the literature are not comparable, as they are based on different data center setups, workloads, and evaluation procedures. Considering such gaps, a systematic, cross-paradigm comparison is needed to determine which scheduling strategy is most effective in energy optimisation, and in different workload characteristics.

1.3 Question and Objectives

1.3.1 Research Question

Which task scheduling approach- heuristic, metaheuristic, or machine learning, provides the highest energy efficiency for allocating tasks to virtual machines in cloud datacenters under diverse and realistic workload conditions?

1.3.2 Research Objectives

In order to answer the research question, the following objectives are defined:

1. To design and implement a simulation framework by using CloudSim Plus for evaluating multiple scheduling algorithms performance for assiging the task to virtual machine under identical experimental conditions.

2. To develop and integrate seven representative scheduling algorithms, including three heuristic algorithms, three metaheuristic algorithms, and one machine learning based algorithm using Deep Q-Network (DQN).

3. To conduct 210 controlled simulation experiments (7 algorithms x 6 scenarios) with energy consumption as the primary performance metric that have supported by makespan, latency and throughput as secondary metrics.

4. To perform a statistical comparison of all scheduling algorithms and identify the most energy-efficient scheduling paradigm under different workload conditions.

While metrics such as makespan and throughput are important, now energy consumption has now become a main concern due to the rapid growth of large-scale datacenters and their environmental and economic impact. Energy-aware scheduling that is directly influences operational cost and sustainability which is making it a more critical optimization objective for modern cloud infrastructures compared to performance-only metrics.

1.4 Limitations

This study is based entirely on simulation using CloudSim Plus and therefore does not capture a few real-world effects, such as hardware overhead, thermal variations, or network jitter, and the Deep Q-Network (DQN) approach requires extensive training and may not fully generalize to workload patterns that were not observed during training. The use of AWS Learner Lab also limits the large-scale physical experimentation due to restricted runtime and instance availability, the energy model used in this study is based on a linear CPU utilization–power relationship, which simplifies real-world datacenter behavior by excluding factors such as cooling, memory power, and I/O energy. While this abstraction is common in CloudSim-based studies and sufficient for relative comparison, future work could extend the model to incorporate more detailed power components.

1.5 Structure of the Report

The thesis is organized in the following way. The chapter Literature Review discusses available literature on heuristic, metaheuristic, and machine learning based scheduling models. The Research Methodology chapter provides a description of the general structure of the experiment, workload design, energy modelling methodology, and evaluation metrics. Design Specification outlines the simulation setup, such as datacenter setup, VM setup, and workload scenario setup. The chapter Implementation is a description of how the scheduling algorithms were developed and integrated into the simulation framework. The Evaluation chapter shows experimental results,and compared each algorithm in the context of energy consumption, makespan, latency, throughput, and statistical analysis. These results are interpreted in the Discussion chapter and compared with the results. The last chapter is the Conclusion and Future Work that recaps the key contributions of this work and provides potential future research directions.

2 Literature Review

The study of task scheduling in cloud computing is not new and has been actively investigated for over ten years. With the constant expansion of cloud services, the data centers are under pressure to utilize their resources effectively with a decrease in energy usage. Most current scheduling methods primarily target the performance variables of makespan, throughput, and response time, but not the energy usage, which is a secondary or non-objective variable. This literature review is also a summary of the key research streams in the heuristic, metaheuristic, and machine learning approaches to cloud scheduling, their strengths and limitations, and the gaps that still remain unaddressed, which drive this thesis.

2.1 Approaches to Scheduling by Heuristic

The earliest method of load balancing in the clouds was heuristic algorithms, which are easy, predictable, and economical in their calculations. Classical policies like Round Robin (RR) and First-Come-First-Served (FCFS) were introduced by Rahman et al. (2014). to distribute tasks. These procedures are effective in stable situations, but the policies remain fixed and do not take into account the dynamics of the resource situation. They consequently end up developing imbalanced loads, and this results in the waste of energy.

Deepa and Cheelu (2017) compared both Static and dynamic heuristics and discovered that approaches like Min-Min and Max-Min are more adaptable to dynamic loads; however, they did not test actual energy consumption. Kalra and Singh (2015) also demonstrated that RR does not take into account the heterogeneity of VM, which may create bottlenecks in performance and also waste power. All in all, heuristic schemes are rapidly deployed and simple, yet they are greedy to the point that they cannot be used to globally optimize or schedule in a way that is energy sensitive.

2.2 Improved Heuristics and SDN-based Scheduling

Some of the studies enhanced conventional heuristics with dynamic thresholds or Software-Defined Networking (SDN). Govindarajan and Kumar (2017) suggested an SDN-based intelligent load balancer that enhanced throughput, and Kriushanth and Arockiam (2015) suggested rules of auto-scaling to decrease the cases of overload. Setiyani and Safitri (2023) and Surendhar et al. (2023) altered RR with a dynamic quantum in order to enhance fairness and reduce waiting time.

These enhanced heuristics do not consider host-level energy or power consumption models, despite having better performance compared to classical methods.hey respond to load changes but are not energy efficient, which is a significant drawback.

2.3 Metaheuristic Scheduling Methods

The metaheuristics are a significant enhancement of heuristics since it is capable of exploring the global search space and can optimise the complex task-VM mappings. Most widely used are the Genetic Algorithms (GA), Particle Swarm Optimisation (PSO) and the Ant Colony Optimisation (ACO).

2.3.1 Genetic Algorithm (GA)

In multi-objective optimisation of makespan and energy, with GA, Madni et al. (2017) obtained energy savings of 25-30 per cent. when compared to RR. Their findings indicate that the evolutionary search assists in avoiding the local minima. GA is, however, inefficient in real-time scheduling, slow and requires a large population and numerous generations. Dai et al. (2015) suggests a hybrid GA and ACO scheduling algorithm, which is a combination of the broad search ability of Genetic Algorithms and the constructive heuristic behaviour of the Ant Colony Optimisation to meet a set of QoS constraints in cloud environments. The study, however, does not cover energy optimisation since it does not consider any host-level power model in order to measure the energy consumption. Moreover, the assessment does not have a statistical validation, no multi-seed experiments, no significant test- constrained faith in the strength of the presented results.

2.3.2 Particle Swarm Optimisation (PSO)

It is a widely used type of metaheuristic algorithm, and it is a popular variety of metaheuristic.

Tsai and Rodrigues (2014) examined PSO and emphasised its rapid convergence. Nevertheless, in discrete scheduling, PSO may prematurely converge in case swarm diversity is lost. Zuo et al. (2014) demonstrated that PSO could generate good makespan performance, but it has to be properly tuned in to prevent local-optimum traps. Many existing PSO-based schedulers only optimise the performance but not energy.

2.3.3 Ant Colony Optimisation (ACO)

ACO is the most energy-conscious in the literature. Investigations conducted by Zhang et al. (2016) indicated that ACO could save upto 30 percent of energy applied through an elite-ant mechanism, and it enhanced ACO by regulating the bounds of pheromones and tested the technique based on real hosts power models. Nonetheless, metaheuristics are typically computationally intensive and need parameter adjustment, and the number of workload cases that are experimented with is commonly two to four, which do not allow the assumption of generalisation.

To conclude, metaheuristics are more optimised than heuristics but also more costly in terms of runtime and can be parameter sensitive.

2.4 Machine Learning and Reinforcement Learning Methods

Predictive machine learning models which forecast the results or outcomes of a specific event or action once the data is provided to them.

2.4.1 Predictive machine learning models

The models that make predictions regarding the outcome or the results of a particular event or action when the information is presented to them.

The initial research utilized regression based models to predict VM load. Jaykrushna et al. (2018) predicted the load by means of linear regression, which enhances the response time but presupposes the linear relationships that do not apply in real cloud systems. Senthil Pandi et al. (2024) used Azure Machine Learning as a means to optimise costs.

2.4.2 Deep RL and Reinforcement Learning

The reason why reinforcement learning (RL) has become an enticing trend is that it does not need human-made rules; however, it is capable of learning the scheduling rules. Mao et al. (2016) proposed DeepRM and found that Deep Q-Network (DQN) can speed up the job completion time more than heuristics. Nevertheless, DeepRM had to train on 10,000 and more training episodes, and it was computationally costly.G and K (2023) suggested a hybrid RL-Bonobo optimiser in order to accelerate convergence, although they never assessed energy. Xiao et al. (2024) used DRL to edge computing scheduling in containers, but latency was the primary criterion instead of power consumption.

The literature primarily assesses scheduling strategies in isolated paradigms such as heuristic, metaheuristic or machine learning-based strategies, but using different assumptions, workloads, and metrics. Such a fragmented evaluation complicates a direct comparison and restricts the effective advice to cloud operators. Contrary to that, this paper

combines the paradigms by performing an analysis of the paradigms in the same experimental environment, allowing for a fair and reproducible comparison of the strengths and limitations of the paradigms under realistic workload conditions.

2.5 Simulation Workload- Studies and Frameworks

Large-scale schedulers require simulation in their evaluation. The most popular frameworks are CloudSim Buyya et al. (2009) and CloudSim Plus with models of VMs, hosts, and applications that are detailed. The linear CPU-power correlation was confirmed by Beloglazov and Buyya (2015) and is used today as the default of energy-conscious simulations.

Nevertheless, the majority of the preceding researches employ simplified/artificial workloads. Empirical trace analyses of real workloads by Reiss et al. (2011) (Google cluster data) revealed that real workloads are very skewed, with numerous short tasks and a few extremely long tasks.Ali-Eldin et al. (2014) empirically proved that cloud workloads tend to be extremely bursty and have sudden spikes that are challenging to foresee. Such behaviour makes the process of scheduling more complicated and makes it clear that there is a need to test algorithms in various and realistic workload conditions.

2.6 Identified Research Gaps

In the reviewed literature, a number of gaps are consistent: None of the studies compares heuristics, metaheuristics, and RL in identical conditions of an experiment. Few works consider energy as the primary measure. The majority of studies employ single-run analysis that does not involve statistical testing. Workloads can be unrealistic, without burstiness or skew or variant of real cloud traces. RL-based method is rarely found in large-space training cost and energy optimization. No work done for finding across six heterogeneous scenarios with multiple seeds.

3 Research Methodology

This research paper adheres to an experimental research approach that is based on simulation to compare and assess heuristic, metaheuristic, and machine learning based task scheduling methods in cloud datacenters. The entire experiment can be performed in a controlled and repeatable environment with the help of CloudSim Plus. Similar workload situations, a single simulation framework, and the datacenter configurations set and standardized performance measurements are employed to provide fairness, comparability, and statistical validity of results. The experimental approach involves four key steps, namely simulation environment preparation, workload scenario creation, controlled performance of scheduling algorithms, and statistical results analysis. Every phase is carefully planned to minimize bias and external variation. In addition, the fixed hyperparameters and optimization budgets were used for all metaheuristic and learning based schedulers to ensure consistency, reproducibility, and fairness across different scheduling paradigms.

3.1 Tools and Software required for development

CloudSim Plus 7.x was the simulation engine that was used to implement the entire scheduling framework. All the scheduling algorithms, workload generators, and brokers were implemented in Java (JDK 17), which could allow standardized object-oriented integration with the CloudSim Plus architecture. The Deep Q-Network model was built and trained with the help of the DeepLearning4J library, which has deep reinforcement learning features. Statistical processing and visualisation were done in Python, based on NumPy, Pandas, SciPy, and Matplotlib. All the simulations and model training were more than satisfied with a MacBook Pro with an Apple M4, which offered the stability needed to run large experimental batches and the long-term reinforcement-learning training phase.

3.2 Simulation Environment Setup

All experiments are performed in one standardized cloud-simulation environment, which is developed with the use of CloudSim Plus. All experimental use a homogeneous datacenter configuration to guarantee that performance variations are only brought about by scheduling strategies and not by hardware variability. The same virtual machines, hosts, and system configurations are preserved in all the experiments. Energy models that validate CPU-based energy models are used to measure energy consumption in the CloudSim Plus framework. Besides energy, other performance indicators such as makespan, average latency, and system throughput are noted in each and every experiment. Controlled experimentation is realized by ensuring that all the scheduling algorithms are exposed to the same conditions of simulation.

3.3 Workload Scenario Design

Six workload scenarios are designed in order to represent realistic cloud operating conditions. These conditions are different attributes of the tasks that are usually encountered in production datacenters, such as balanced work loads, skewed workloads, significant variability in workload size, and bursty arrival schedules. The number of tasks generated by each scenario is the same, but the distribution of tasks and the behavior of arrivals differ. To exclude the bias of randomness, five independent random seeds are used per workload situation in order to ensure statistical reliability. In each experiment, the same workloads are given to all the scheduling algorithms to ensure that they have been fairly compared.

3.4 Scheduling algorithms under evaluation

This paper compares seven representative scheduling algorithms of three leading paradigms. There are three heuristic algorithms (Round Robin, Min-Min, and Max-Min), three metaheuristic algorithms (Genetic Algorithm, Particle Swarm Optimization, and Ant Colony Optimization), and one machine learning-based algorithm on the platform of Deep Q-Network (DQN). All the scheduling algorithms are executed in the same conditions of the system and with the same inputs of the workload. There is no algorithm that receives a privilege in terms of resources or extra information on the system. This makes sure that performance differences observed are only based on the behavior of the scheduling strategy and not environmental benefits.

3.5 Experimental Execution Process

This is a completely automated experimentation process that has a centralized simulation execution pipeline. Every experimental run is based on a standard procedure: the datacenter is initialized, the desired workload set-up is loaded, the desired scheduling algorithm is started, the tasks are submitted, and the simulation is run. The formula is used to conduct 210 experiments: 7 algorithms x 6 scenarios x 5 random seeds = 210 simulations. Each simulation is automatically monitored and all its total energy consumption, makespan, average latency, and throughput are recorded and stored in the folder where all CSV files are. These files would be used as the raw data to be used in the performance analysis. After the completion of all 210 simulation runs, the generated CSV result files were stored in the results directory, which are manually run by using two standalone Python analysis scripts. These scripts are executed separately to generate performance graphs, to perform four statistical validation tests for algorithm comparison, and to produce the final winner and ranking CSV files used in the evaluation.

3.6 Reproducibility and Statistical Validations

Deterministic generation of workload and constant random seeds in every experimental run guarantee its reproducibility. There is a Python based two scripts files that use fixed input CSV files to ensure that all statistical results and performance graphs are fully reproducible. The same input conditions ensure the possibility of repeating any experiment. The reliability of observed differences in performance is statistically tested to validate the performance differences. Several statistical tests are employed, such as the Friedman test to rank in general, the Wilcoxon signed-rank test to compare two, and effect size analysis to quantify the practical significance. These tests justify that the results which are observed are not merely by chance, but they constitute statistically significant performance differences between. Additionally, the source code for all the scheduling algorithms is provided in the GitHub link, which is available in the footnote. The complete simulation source code is accessible in the GitHub repository.[1]

4 Design Specification

This chapter introduces the workflow design of the proposed simulation structure that is created to determine the task scheduling approaches in cloud datacenters. The design is based on a sequential and modular execution cycle in order to provide reproducibility, controlled experiment, and fair comparison of various scheduling paradigms. The entire workflow is structured into five steps: workload generation, scheduling decision layer, simulation execution, energy and performance monitoring, and experiment automation and post-processing analysis. The workflow diagram 1 represents the general process.

4.1 Scenario Engine (WorkLoad Phase)

The Scenario Engine start the workflow and is in charge of creating cloud workload traces that are used as input in the simulation. The module generates cloudlets, which are user tasks with fixed execution periods, arrival times, and processing demands. There are 6 workload conditions created, such as Balanced, ShortHeavy, LongHeavy, HighVariance,

[1]GitHub Repository: `https://github.com/Amit-NCI/Energyconsumption`

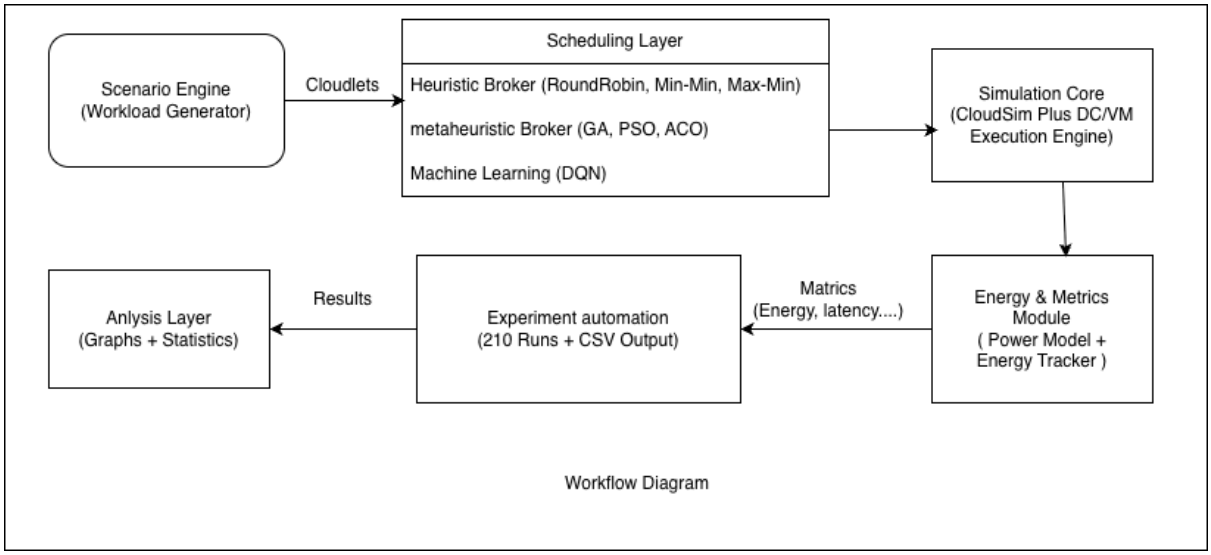


Figure 1: WORKFLOW DIAGRAM

RealisticRandom, and BurstyRealistic. Even though the overall number of tasks is the same in all scenarios, the statistics of the sizes of tasks and their arrival behavior will be different in each case to capture realistic cloud workload behavior. The resulting cloudlets are sent to the scheduling decision layer for assigning the tasks to each scheduling algorithm.

4.2 Decision-Making Layer (Scheduling Layer)

The generated cloudlets are sent to the Scheduling Layer, which allocates the tasks to the VM with three types of scheduling brokers. The Heuristic Broker applies Round Robin, Min-Min, and Max-Min scheduling algorithms that are based on rules. The Metaheuristic Broker uses population-based global optimisation methods such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO). The Machine Learning Broker uses a deep Q-Network (DQN) model, where the cloudlets are appropriately assigned to the VM are based on policies learned around the state-action space. The scheduling decisions are generated separately by each broker and sent to a simulation execution core as a final task-to-VM mapping.

4.3 CloudSimPlus Execution Engine

The Simulation Core is developed on the framework of CloudSim Plus and serves as the virtual environment of a cloud datacenter. The entire cloud platform is developed at the start of every simulation run with the CommonInfrastructure module. It involves the establishment of two datacenters, each comprising of several physical hosts, and provisioning of virtual machines (VMs) on the physical hosts with predetermined CPU, memory, and bandwidth resources. When the infrastructure is initiated, cloudlets forwarded by the scheduling layer are run on the respective VMs with the discrete event-based simulation framework of CloudSim Plus. This step is able to model task queueing, execution delay, and resource utilisation, hence a realistic simulation of the system behaviour during the runtime under various strategies of scheduling.

4.4 Energy and Performance monitoring Layer

The Energy and Metrics Module is responsible for continuously tracking the CPU usage of every host at each point of the simulation and calculating the energy usage based on the proven linear host CPU power model offered as CloudSim Plus. Simultaneously, it documents the key performance indicators such as the total energy used (Wh), makespan (seconds), average latency (seconds), and throughput (tasks per second). These metrics are exported after every simulation run in the form of structured CSV files, and they constitute the raw dataset to be further statistically analyzed.

4.5 Automation and Analysis Layer

The Experiment Automation Layer is used to control the entire experiment. It performs all combinations of 7 scheduling algorithms, 6 workload scenarios, and 5 independent random seeds in a systematic way, giving a total of 210 simulation runs. During each run, the automation layer starts the simulation environment, loads the workload under consideration, starts the chosen scheduler, runs the simulation, and saves the performance metrics gathered during the simulation. Once all simulation runs have been completed, the resulting CSV files are processed manually with two Python based analysis scripts. These scripts produce performance graphs in terms of comparison, four statistical validation tests, an algorithm ranking file, and scenario-based winner summary CSV files. The resultant processed findings are then incorporated in the Evaluation and Discussion chapters, where performance interpretation is carried out in more detail.

5 Implementation

This chapter describes the technical implementation of the simulation framework on CloudSim Plus for seven scheduling algorithms, six types of workload generation, and analysis tools. The simulation focused on the three types of paradigms: heuristic, meta-heuristic, and Machine learning (DQN). Without providing the source code details, this chapter explains how each system component was built, setup and used to generate the repeatable experiment results. This chapter is divided into different sections

5.1 VM and Datacenter Configurations

CloudSim Plus uses a centralized CommonInfrastructure module to ensure consistency across all experiments. The simulated environment comprises 2 datacenters, each having 4 homogeneous physical hosts, for a total number of 8 host. All hosts will have 12 CPU cores (PEs), 65,536 MB RAM, 200,000 bandwidth, and 2,000,000 storage units. Linear CPU power model is used to measure the energy consumption of each host by applying PowerModelHostSimple class with an idle power of 70 W and maximum power of 200 W. Virtual machines (VMs) are generated in multiplicity of 24, 3 heterogeneous VMs each on a host having number of CPU cores (6,4,2) therefore, the number of cores available on each host is used completely. Each VM is constrained by a set of resources to ensure that all the scheduling algorithms are tested under the same completely controlled infrastructure conditions before any task scheduling decisions are applied.

5.2 Workload Scenario Engine

A custom Scenario Engine has been applied in the form of the WorkloadGenerator class to create six realistic workload scenarios, i.e., Balanced, ShortHeavy, LongHeavy, HighVariance, RealisticRandom, and BurstyRealistic. In both scenarios, a set of controlled cloudlets is generated, with each scenario defining the length of tasks (in MI), the amount of CPU utilized, the number of required processing elements (PEs), the size of files, and the size of the output, such that cloud workloads are realistic to model. The Balanced case makes evenly distributed tasks of moderate CPU load, whereas there are mainly short-duration tasks in the ShortHeavy case and large computational tasks in the LongHeavy case. HighVariance scenario combines very small and mediate and enormous jobs to model heterogeneous behavior of execution. The RealisticRandom case is based on the distribution of workloads on a probability distribution in the model of realistic production environments based on micro, small, medium, large, and huge jobs. The BurstyRealistic scenario presents bursts of arrivals of tasks to simulate flash-crowd and peak-load effects that are usually found in actual cloud systems. Deterministic random seeds have been used on all cases to ensure reproducibility between five independent experimental cases. All the cloudlets created are delivered directly to the scheduling brokers, such that each of the scheduling algorithms is tested on the same workload conditions to make a fair and unbiased evaluation.

5.3 Broker Scheduling and Integrating Algorithms

To reflect the heuristic, metaheuristic, and reinforcement-learning paradigms, three types of scheduling brokers were applied.

5.3.1 Heuristic

The heuristic layer of the scheduling in the given research comprises three well-known deterministic scheduling algorithms, which include Round Robin, Min-Min, and Max-Min, and are implemented with specific CloudSim Plus extensions of brokers. The Round Robin algorithm has a basic cyclic VM selection algorithm such that the incoming cloudlet is allocated to the next available virtual machine, in a circular sequence, without thinking of the execution time or the load on the system, and thus, equal distribution of tasks is achieved with a minimal computational overhead. Min-Min algorithm works by initially estimating the time needed to execute each unscheduled cloudlet on each available VM by using the cloudlet length of instruction and VM processing capacity, choosing per cloudlet the VM that completes it with the least time, and finally placing the cloudlet with the lowest time to complete it on its VM; execution time is remembered after each mapping in order to maintain the accuracy of the execution order. By contrast, the Max-Min algorithm uses an inverse selection strategy according to which the algorithm calculates minimum completion time of each unscheduled cloudlet on all VMs and then picks a cloudlet with the maximum among these minimum completion times to ensure long tasks are scheduled first to eliminate the possibility of shorter jobs starving the long task. Both Min-Min and Max-Min explicitly measure VM ready time with the help of internal scheduling maps and allocate cloudlets to VMs before the simulation actually runs, and let CloudSim Plus follow the scheduling decisions. These heuristic algorithms do not need any tunable hyperparameters, and have a baseline behavior of scheduling behaviour to compare the optimization-based and learning-based methods.

5.3.2 Metaheuristic

The Genetic Algorithm (GA) broker is used as a population based evolutionary optimizer in an effort to minimize a weighted objective function of energy usage and makespan during task-to-VM scheduling. The chromosome will encode a complete scheduling solution with an integer gene array, with each gene being the VM assignment of a cloudlet. The GA has a population size of 50 chromosomes that have evolved in 100 generations with a crossover rate of 0.8 and a mutation rate of 0.1 to keep diversity alive. Parent selection is done using tournament selection with a size of 3, and elitism to protect the best 2 solutions in each generation is used; otherwise, high-quality candidates will be wasted. Fitness of individual chromosome is calculated as a weighted mean as energy (weight = 0.6) + makespan (weight = 0.4), where the energy is approximated as a linear average power model with idle power of 70 W and maximum power of 200 W. Evolution is done by single-point crossover and probabilistic mutation till convergence. The optimal chromosome in the whole world is used in the final stage of the evolution, where the cloudlets are bound to the virtual machines of their choice, becoming an optimal solution of scheduling. **Particle Swarm Optimization (PSO)** broker is a swarm intelligence-based metaheuristic optimizer for minimizing the weighted objective of energy consumption and makespan in the task-to-VM scheduling. The swarm size of the PSO is 50 particles that are developed through 100 iterations. The individual particles are a full schedule (discrete task-to-VM position) vector and a continuous velocity vector. A sigmoid (non-linear) adaptive inertia weight is used so that it declines from 0.95 to 0.35 to balance the exploration and exploitation in the search process. The cognitive and social learning coefficients take the form of asymmetrically set. c1 = 1.7 and c2= 2.1, which provides more weight to the global best solution. The dynamic velocity clamping is such that the maximum velocity used is reduced from 6.0 to 2.0 to allow coarse global search during the initial phases, and fine grained convergence during the late phases. The fitness function is calculated as a weighted sum of normalized energy (weight = 0.6) and makespan (weight = 0.4) with energy modeled as a linear average power model with idle power of 70 W and maximum power of 200 W. In order to prevent premature convergence, advanced strategies such as learning on elites (top 10 percent leading the bottom 20 percent) and diversity enhancement by aggressive perturbation of 25 percent of the worst particles are applied when a stagnation is observed over 8 consecutive iterations. Once convergence is achieved, the worldwide optimum particle is used on the CloudSim Plus simulator by attaching each cloudlet to its optimal virtual machine, yielding the ultimate PSO-based scheduling model. **Ant Colony Optimization (ACO**) broker is applied as a population-based bio-inspired metaheuristic to optimize a weighted goal of total energy usage and makespan in task to VM scheduling. The colony is a collection of 30 ants that are the result of 100 optimization cycles. Every ant builds a full-fledged scheduling solution by probabilistically placing each of the cloudlets into a virtual machine guided by a pheromone-heuristic decision rule, where probabilities of selections are controlled by pheromone importance alpha = 1, beta =2. The heuristic data are obtained by the consideration of the inverse cost model, which indicates the performance of estimated time execution and energy expenditure. The evaporation rate of pheromones is used in an adaptive manner, whereby the rate of evaporation rises linearly with the rate of evaporation. p = 0.1 to p = 0.3 in an iteration, which allows a gradual movement in between exploration and exploitation. Pheromone updates use an elitist approach to the deposit of pheromones, and each ant deposits a proportional pheromone deposit of Q= 100 or

deposits a supplementary elite-weighted reinforcement of factor 2.0. Min-max pheromone limits are imposed to avoid premature convergence and stagnation in the range of 0.01 to 10.0, the fitness of every ant is calculated as weighted sum of normalized energy (weight 0.6) and makespan (weight 0.4) where energy is estimated with a linear average power model, idle power = 70 W and maximum power = 200 W. Once convergence has taken place, the solution of the globally best ant is applied to the CloudSim Plus environment through binding each cloudlet to its optimal assignment of a virtual machine.

5.3.3 Machine Learning

The Deep Q- Network (DQN) broker can be used as a RL based scheduling agent to optimize the task to VM assignment through an energy-aware execution. DQN training takes place in 2000 episodes with 2 repeat training passes per episode to improve learning stability. It is a neural network with three fully connected hidden layers containing 128, 128, and 64 neurons; the activation function is ReLU, and the output layer is Q-values of all the available virtual machines. It is trained with the Adam optimizer, continuing with a learning rate of 0.0003 and a discount factor (gamma) of 0.99. Epsilon-greedy exploration strategy is utilized, and exploration rate (epsilon) fades away linearly between 1.0 and 0.05 throughout training. There is a prioritized experience replay buffer of capacity 200000 samples with a mini-batch size of 128 samples, so that there is stability in the gradient update. To ensure that Q-value divergence does not occur, a target network is aligned with the online network after every 1000 training steps. The task length, the number of processing elements required, the remaining workload ratio, and the current virtual machine load distribution are used to code the system state. The reward function is described as the execution time of the negative tasks as an incentive in order to make more energy efficient and faster scheduling decisions. After the training, the acquired scheduling policy is used deterministically to assign cloudlets to the virtual machines of the CloudSim Plus environment. The trained model is stored the zip files, which are reused over simulation runs in order to provide uniform and reproducible performance assessment.

5.4 Simulation Runner automation

To control the 210 simulations that are needed by the experimental design, an automation controller was created. The controller repeatedly did the combination of each of the seven algorithms, each of the six workload scenarios, and each of the five random seeds. Each run of the controller re-initialised the datacenter, recreated the VM pool, recreated or loaded the workload scenario, and called the relevant scheduling broker. After completion, the metrics that were produced, including the total energy consumption, makespan, latency, and throughput, were exported into structured CSV files in scenario-specific directories. This design was such that it provided 100 percent reproducibility and zero chances of manual error. After all simulations run are completed, the generated CSV files are manually processed using Python based analysis scripts to generate plots for all results, such as energy consumption, latency, throughput, makespan, and statistical validation results for algorithm ranking summaries and comparison of paradigm box plot.

5.5 Statistical Analysis and Visual Production of output

Once all the simulation runs are completed, the detailed and summary CSV files are analyzed (using an external analysis module written in Python). The CSV files has been generated by the MetricsCollector class files, which provides both task-level and absolute performance metrics such as total energy, average latency, maximum latency, makespan, throughput, and average execution time. The energy values employed in this analysis come directly out of the special EnergyTracker module that samples power consumption of the host at runtime and fairly allocates the energy usage to each virtual machine according to proportional CPU utilization. The Python analysis module summarizes these metrics on all the runs, makes scenario-wise and algorithm-wise summary tables, and makes comparative performances plots and boxplots to determine variability. Moreover, to check the statistical significance and practical use of the results of the differences in the performance, several statistical validation tests, such as the Friedman test to determine the overall ranking, the Wilcoxon signed-rank test to compare the differences between two variables, paired t-tests, and measures of the effects, e.g., Cohen's d and coefficient of variation, are utilized. These are processed numeric and visual findings that are used as the basis of the Evaluation and Discussion chapters, where conclusions are made in accordance with empirical and statistical findings.

6 Evaluation

The analysis of the 210 controlled simulations run on seven scheduling algorithms on six workload conditions and with five independent seeds per experiment is given. Because each of the simulations was performed with the same CloudSim Plus datacenter settings, the only discrepancy in the performance is the scheduling strategies. The energy consumption is addressed as the primary evaluation parameter since it is relevant to datacenter sustainability, whereas makespan, latency, and throughput are applied to total scheduling performance measurements. This evaluation aims to identify which algorithmic paradigm, namely the heuristic, metaheuristic, or reinforcement learning, provides the most energy-performance tradeoff and the impact of the workload characteristics on the aptness of the algorithm.

6.1 Total Energy Consumption

The initial analysis is on the overall energy usage of each and every algorithm and scenario. Table 1 summarizes the average amount of energy used by each algorithm when using each pattern of workload (in Wh).

The findings indicate that DQN is the lowest energy user in the Balanced, Short-Heavy, LongHeavy, HighVariance, and RealisticRandom settings. DQN uses nearly half the amount of energy as Round-Robin and over a quarter as Min-Min, on average. The savings are due to its capability to learn task-VM affinity patterns, as well as scheduling tasks so as to reduce the CPU variations and idle time on hosts. The metaheuristic algorithms, particularly the ACO and GA, also have significant savings, but their optimisation procedure take more time to compute.

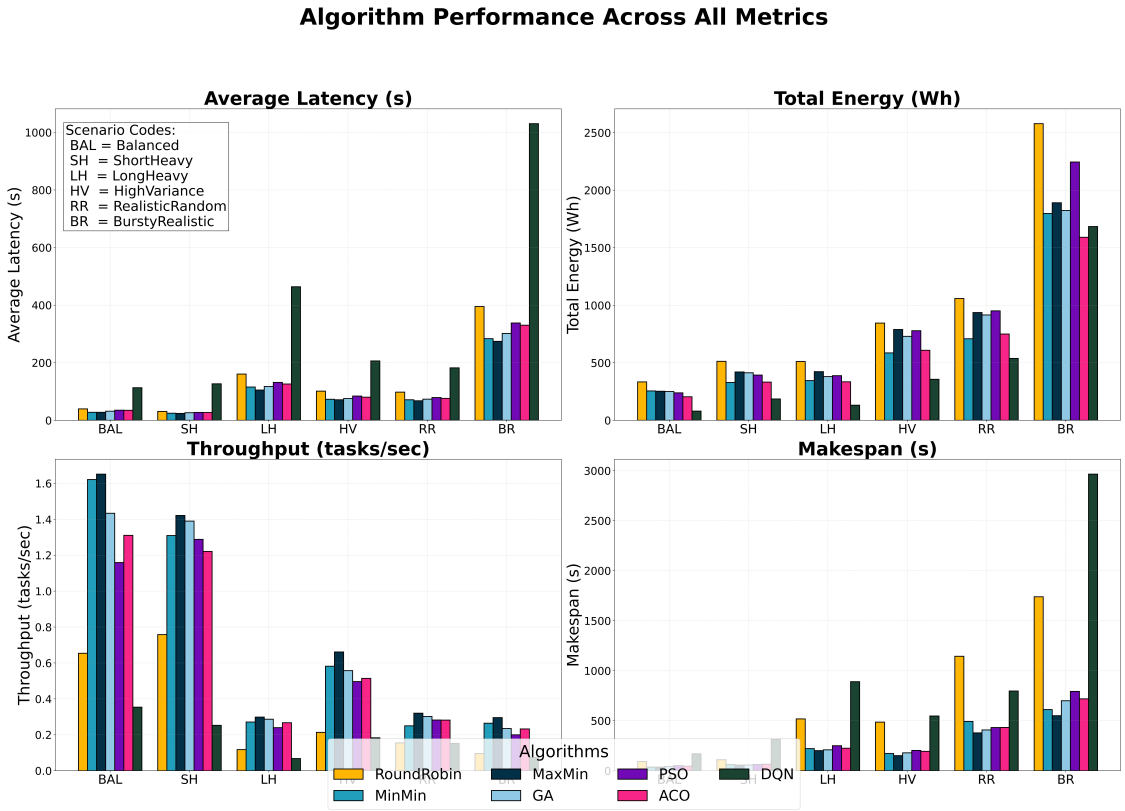
The BurstyRealistic situation generates another result. In this case, ACO turns out to be the most efficient algorithm, in terms of consumption in the average as compared to DQN. The adaptation of ACO is more adaptive since the pheromone-based construct-

Table 1: Energy Consumption Comparison Across Algorithms scenario-wise

Scenario	DQN	ACO	GA	PSO	MinMin	MaxMin	RR
Balanced	80.39	205.52	250.40	239.19	255.73	252.99	334.43
ShortHeavy	186.44	332.49	413.39	393.55	329.36	420.70	513.10
LongHeavy	132.31	334.91	380.75	388.51	345.83	423.17	511.15
HighVariance	357.18	608.78	730.44	779.26	586.12	789.81	845.17
RealisticRandom	538.72	750.09	916.00	951.90	709.42	937.13	1059.73
BurstyRealistic	1684.98	1591.58	1824.38	2245.29	1797.91	1891.40	2578.42

ive search is responsive to abrupt bursts of incoming tasks, whereas the trained policy under DQN falters every time the conditions under evaluation change significantly with respect to its training environment. This example demonstrates a factor well known to reinforcement learning models, that they do not generalise well when workload patterns are extremely irregular.

ACO and GA have narrow distributions of energy, which implies stability in performance. There is an extensive dispersion and high-energy outliers in heuristic methods. DQN has demonstrated high performance on most of the runs, but it is sensitive to some random seeds, particularly when there are irregular workloads. Such distributions indicate the capabilities and restrictions of the three paradigms and lay the basis of further evaluation scenario-by-scenario.



6.2 Analysis of Energy, Latency, Throughput, and Makespan

The section performs a combined multi-metric comparison analysis of the behavior of the energy consumption, average latency, throughput, and makespan in six workload scenarios with the combined multi-metric comparison graph **??**. The scheduling in each of the workloads is different: ShortHeavy and LongHeavy have skewed task lengths, HighVariance has extreme heterogeneity, and BurstyRealistic has bursts of tasks that burst the scheduler with the need to adapt. In all the Balanced, ShortHeavy, LongHeavy, and High Variance cases, DQN has the lowest energy consumption compared to any other algorithm. This is as certains its robust capability to learn energy-efficient task to VM mappings at constant workloads and at moderately changing workloads. DQN has a distinct energy edge even in HighVariance, which is characterized by a much more pronounced complexity in scheduling because of its capability to reproduce non-linear interactions between workload and resource. Nonetheless, in the BurstyRealistic case, ACO performs the best in terms of energy consumption compared to the rest of the algorithms. This implies that pheromone-directed constructive search has greater flexibility to abrupt changes in workload. However, the trained DQN policy exhibits lower generalization in the case of highly irregular workloads, which also leads to high energy consumption and latency. The behavior of latency comes immediately after the energy behavior. Owing to its inflexible policy that is rule-based, heuristic algorithms have low latency with stable workloads but scale to substantially lower levels with bursty workloads and highly variable workloads. Metaheuristic algorithms, in general, ACO and PSO, offer a moderate latency under the majority of workloads. DQN exhibits relatively increased latency during highly changing and bursty conditions due to the explicit purpose of its policy of optimizing long-term energy usage, rather than short-term reactivity. In terms of throughput and makespan, GA and PSO always have the smallest makespan and largest throughput in most cases. The globalization optimization techniques of their population allow the efficient load balancing of virtual machines, as a result in a rise in the velocity of execution and the productivity of the system. ACO has a trade-off in completeness that is not too fast at the cost of better energy efficiency. The heuristic algorithms show reasonable performance in Balanced and ShortHeavy workloads and are significantly worse in LongHeavy, HighVariance, and BurstyRealistic workloads because they have rigid rules of scheduling.

DQN has lower throughput and higher makespan, especially when loaded with BurstyRealistic workload. This is not surprising, as the learned policy is aimed at minimizing power utilization, as opposed to maximizing the speed of execution, and so slower, albeit more efficient, scheduling decisions are made.

In general, this combined analysis proves the existence of a basic trade-off between the speed of execution and power consumption. Metaheuristic algorithms are the most balanced in all the metrics; heuristic algorithms are simple and limited in performance, and DQN is very effective in minimizing energy at the throughput and makespan cost.

6.3 Statistical Significance and Reliability

A number of statistical tests have been performed to confirm the differences of performance observed between the scheduling algorithms. The Friedman test establishes that the difference is statistically significant at all workloads with p-values of below 0.01. Pairwise Wilcoxon signed-rank test results indicate that in the BurstyRealistic scenario, ACO is significantly superior to heuristic and DQN-based schedulers, and that in the other scen-

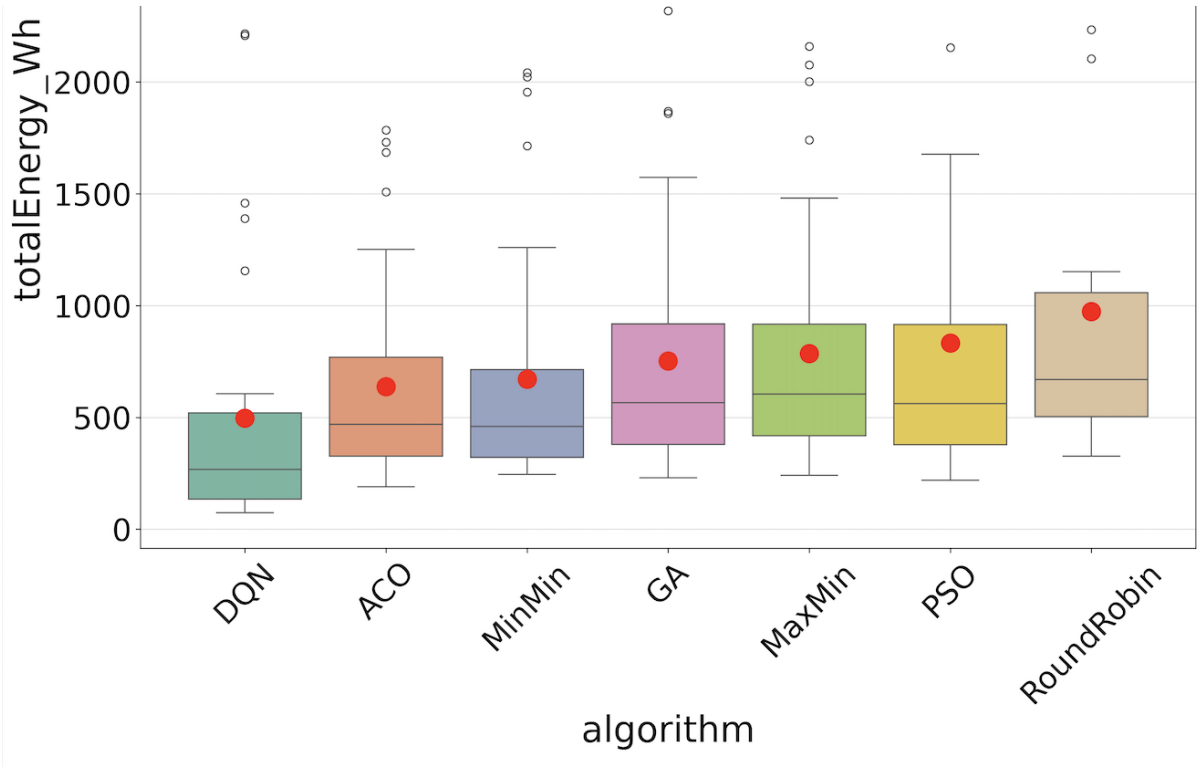


Figure 2: Statistical analysis(Energy Distribution) boxplot 1 for all algorithms.

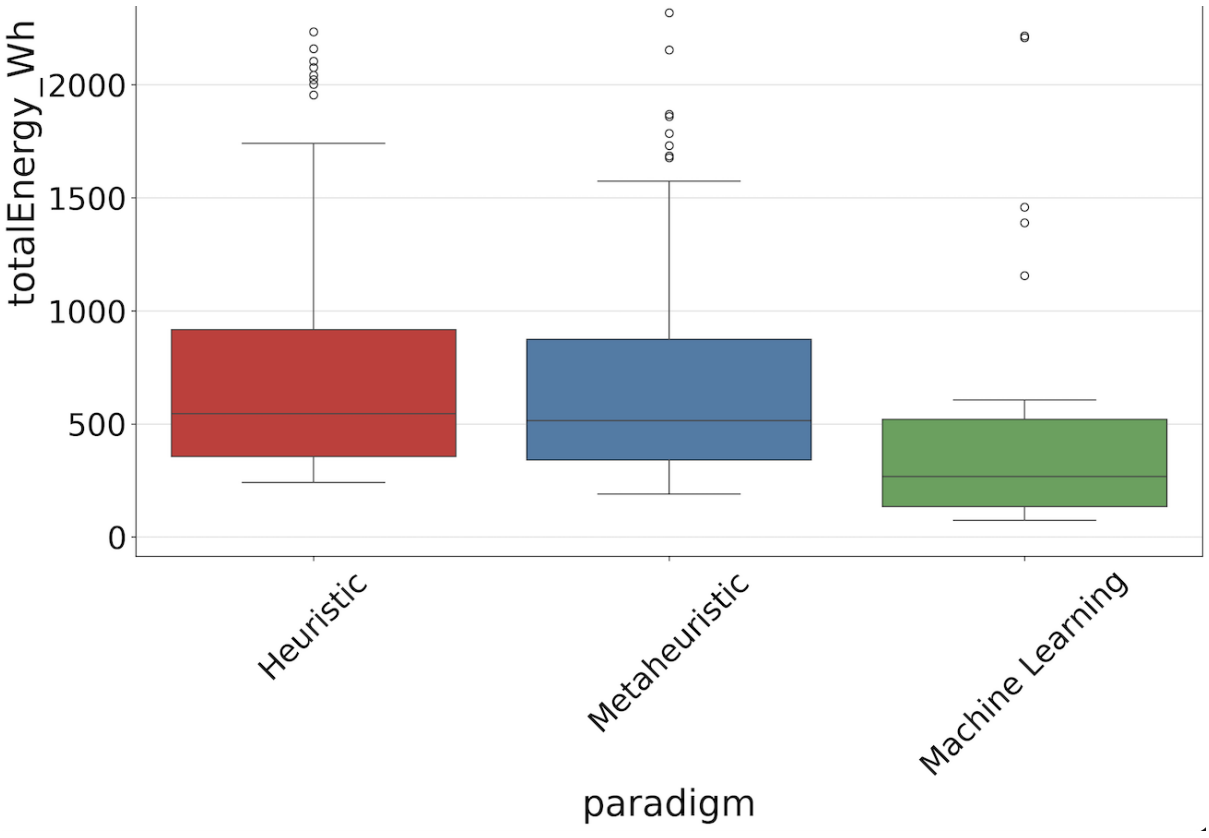


Figure 3: Statistical analysis (Energy Distribution) boxplot 2 for all paradigms.

arios, DQN is statistically efficient in terms of energy usage as compared to heuristic methods. As well, paired t-tests confirm the fact that GA, PSO, and ACO have much better makespan and latency than heuristic algorithms. These differences are further validated to mean something practically significant and not as a result of randomness, as shown by the box-plot distributions and the effect size analysis.

6.4 Discussion

The experimental findings of 210 controlled experiments clearly show that there are marked differences in behavior between heuristic, metaheuristic, and reinforcement learning-based scheduling paradigms in varied workload situations. Metaheuristic algorithms, especially ACO, demonstrate the most consistent and sound level of energy-efficiency in highly variable and bursty workload, which confirms the power of the pheromone-based global searching in the adaptation to the sudden workload changes. However, heuristic methods, despite being computationally cheap, are always poorly adaptable and consume more energy because of their inflexible and shortsighted task allocation methods, which are inappropriate in a dynamic datacenter environment. The DQN-based scheduler uses the least amount of energy in five of six cases, which confirms that reinforcement learning can be used to train deep energy-aware scheduling policies with structured and moderately varying workloads. Nonetheless, the poor performance in the BurstyRealistic case illustrates the generalization weakness of reinforcement learning that is commonly recognized in situations where the test conditions are vastly different from the distributions during training. In general, the findings support the conclusion that is the best scheduling paradigm is the most irregular and bursty driven metaheuristics is superior in the irregular workload, reinforcement learning is most efficient in a structured one, and heuristics is merely a low-complexity baseline paradigm. These results respond directly to the research question by showing that workload variability, burstiness, and task heterogeneity are the key factors behind the design of the best scheduling strategy.

7 Conclusion and Future Work

This paper included a cross-paradigm comparison of seven cloud scheduling algorithms based on heuristic, metaheuristic, and reinforcement learning models across six workload conditions that are realistic using CloudSim Plus. The results are based on 210 statistically controlled simulations, and they prove that algorithmic choice decisively affects datacenter energy efficiency and performance. DQN scheduler had the lowest energy consumption in five of the six situations, and was vastly superior to classical heuristics, and ACO was more robust with highly bursty workloads. Metaheuristic algorithms, including GA and PSO, were always in the strongest area of responsiveness, with the best makespan and latency performance. In general, the findings demonstrate that no universal optimality of a scheduling strategy can exist and that the most appropriate paradigm highly depends on the nature of the workload, including burstiness, variance, and task-length distribution.

The next research should be directed to the creation of hybrid scheduling models that will merge the global exploration with metaheuristics and the rapid inference ability of reinforcement learning to address the limitations faced by high bursty workloads. Transfer learning and offline reinforcement learning might play a great role in lowering the cost of training and enhancing the generalization of a variety of datacenter settings. Moreover,

implementation of the given framework on actual cloud platforms like AWS or Azure would also allow testing it on actual thermal, network, and power limits. Lastly, the model should be extended to include carbon-intensity-aware scheduling and multi-datacenter optimization to make the model more sustainable and allow the practical implementation of energy-aware scheduling in large-scale cloud systems.

References

Ali-Eldin, A., Seleznjev, O., Sjöstedt-de Luna, S., Tordsson, J. and Elmroth, E. (2014). Measuring cloud workload burstiness, *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pp. 566–572.
URL: *https://ieeexplore.ieee.org/document/7027554*

Beloglazov, A. and Buyya, R. (2015). Openstack neat: a framework for dynamic and energy-efficient consolidation of virtual machines in openstack clouds, *Concurrency and Computation: Practice and Experience* **27**(5): 1310–1333.
URL: *https://clouds.cis.unimelb.edu.au/papers/OpenStack-Neat-CCPE.pdf*

Buyya, R., Ranjan, R. and Calheiros, R. N. (2009). Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities, *2009 International Conference on High Performance Computing Simulation*, pp. 1–11.
URL: *https://ieeexplore.ieee.org/document/5192685*

Dai, Y., Lou, Y. and Lu, X. (2015). A task scheduling algorithm based on genetic algorithm and ant colony optimization algorithm with multi-qos constraints in cloud computing, *2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics*, Vol. 2, pp. 428–431.
URL: *https://ieeexplore.ieee.org/document/7335004*

Deepa, T. and Cheelu, D. (2017). A comparative study of static and dynamic load balancing algorithms in cloud computing, *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pp. 3375–3378.
URL: *https://ieeexplore.ieee.org/document/8390086*

G, A. M. and K, B. (2023). Reinforcement learning-based bonobo optimizer for efficient load balancing in cloud computing, *2023 3rd International Conference on Intelligent Technologies (CONIT)*, pp. 1–5.
URL: *https://ieeexplore.ieee.org/document/10205866*

Govindarajan, K. and Kumar, V. S. (2017). An intelligent load balancer for software defined networking (sdn) based cloud infrastructure, *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pp. 1–6.
URL: *https://ieeexplore.ieee.org/document/8117881*

Jaykrushna, A., Patel, P., Trivedi, H. and Bhatia, J. (2018). Linear regression assisted prediction based load balancer for cloud computing, *2018 IEEE Punecon*, pp. 1–3.
URL: *https://ieeexplore.ieee.org/document/8745409*

Kalra, M. and Singh, S. (2015). A review of metaheuristic scheduling techniques in cloud computing, *Egyptian Informatics Journal* **16**(3): 275–295.
URL: *https://www.sciencedirect.com/science/article/pii/S1110866515000353*

Koomey, J. et al. (2011). Growth in data center electricity use 2005 to 2010, *A report by Analytical Press, completed at the request of The New York Times* **9**(2011): 161.
URL: *https://alejandrobarros.com/wp-content/uploads/old/4363/Growth$_i$n$_D$ata$_C$enter$_E$lectricity$_u$se*

Kriushanth, M. and Arockiam, L. (2015). Load balancer behavior identifier (lobbi) for dynamic threshold based auto-scaling in cloud, *2015 International Conference on Computer Communication and Informatics (ICCCI)*, pp. 1–5.
URL: *https://ieeexplore.ieee.org/document/7218115*

Madni, S. H. H., Abd Latiff, M. S., Abdullahi, M., Abdulhamid, S. M. and Usman, M. J. (2017). Performance comparison of heuristic algorithms for task scheduling in iaas cloud computing environment, *PLoS ONE* **12**(5): e0176321.
URL: *https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0176321*

Mao, H., Alizadeh, M., Menache, I. and Kandula, S. (2016). Resource management with deep reinforcement learning, *HotNets '16: Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pp. 50–56.
URL: *https://dl.acm.org/doi/10.1145/3005745.3005750*

Rahman, M., Iqbal, S. and Gao, J. (2014). Load balancer as a service in cloud computing, *2014 IEEE 8th International Symposium on Service Oriented System Engineering*, pp. 204–211.
URL: *https://ieeexplore.ieee.org/document/6830907*

Reiss, C., Wilkes, J. and Hellerstein, J. L. (2011). Google cluster-usage traces: format+ schema, *Google Inc., White Paper* **1**: 1–14.

Senthil Pandi, S., Kumar, P. and Salman Latheef, T. A. (2024). Cloud platform optimization using azure machine learning to improve performance and reliability, *2024 International Conference on Computational Intelligence for Green and Sustainable Technologies (ICCIGST)*, pp. 1–6.
URL: *https://ieeexplore.ieee.org/document/10717550*

Setiyani, L. and Safitri, C. (2023). Dc - drc optimization using load balancer (case study: Karawang regional hospital), *2023 Eighth International Conference on Informatics and Computing (ICIC)*, pp. 1–6.
URL: *https://ieeexplore.ieee.org/document/10382047*

Surendhar, K., Chamarthy, L. S., Priyadharshini, D., Keerthana, G., Sinha, G. and Kumar, A. V. (2023). Enhancement of round robin algorithm with dynamic quantum in cloud computing, *2023 International Conference on Inventive Computation Technologies (ICICT)*, pp. 799–803.
URL: *https://ieeexplore.ieee.org/document/10133995*

Tsai, C.-W. and Rodrigues, J. J. P. C. (2014). Metaheuristic scheduling for cloud: A survey, *IEEE Systems Journal* **8**(1): 279–291.
URL: *https://ieeexplore.ieee.org/document/6516911*

Xiao, Z., Liu, K., Hu, M. and Wu, D. (2024). Deepcts: A deep reinforcement learning approach for ai container task scheduling, *Proceedings of the 2024 ACM Conference on Advances in Computational Machine Learning (CACML)*, Association for Computing

Machinery, New York, NY, USA.
URL: *https://dl.acm.org/doi/10.1145/3654823.3654885*

Zhang, L., Wang, Y., Zhu, L. and Ji, W. (2016). Towards energy efficient cloud: an optimized ant colony model for virtual machine placement, *Journal of Communications and Information Networks* **1**(4): 116–132.
URL: *https://ieeexplore.ieee.org/document/8932322*

Zuo, X., Zhang, G. and Tan, W. (2014). Self-adaptive learning pso-based deadline constrained task scheduling for hybrid iaas cloud, *IEEE Transactions on Automation Science and Engineering* **11**(2): 564–573.
URL: *https://ieeexplore.ieee.org/document/6574281*