Journal of Information and Organizational Sciences
Volume 49, Number 2 (2025)
Journal homepage: jios.foi.hr

JIOS

DOI: 10.31341/jios.49.2.3
Open Access

UDC 004.77:004.021:004.85
Original Scientific Paper

The Evolution of Cloud through SJF-ML Hybrid Scheduling

Prathamesh Vijay Lahande[1*]

[1]DES Pune University, Pune, India
*Correspondence: prathamesh.lahande23@gmail.com

PAPER INFO

Paper history:
Received 02 May 2025
Accepted 22 July 2025

Citation:
Lahande, P. V. (2025). The Evolution of Cloud through SJF-ML Hybrid Scheduling. In Journal of Information and Organizational Sciences, vol. 49, no. 2, pp. 193-211

Copyright:
© 2025 The Authors. This work is licensed under a Creative Commons Attribution BY-NC-ND 4.0. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

ABSTRACT

Purpose: The author proposes sixteen Shortest Job First - Machine Learning (SJF-ML) hybrid algorithms, combining the cloud's SJF scheduling algorithm with four ML algorithm categories, with cloud evolution through ML intelligence as the primary objective. The four categories include: SJF-CA, SJF-ELA, SJF-PM, and SJF-RA. The developed SJF-ML algorithms by the author perform pattern recognition of the tasks that are to be computed, to improve decision-making during task computations in the cloud. These sixteen SJF-ML algorithms include: SJF-ADAB, SJF-BAY, SJF-DT, SJF-KNN, SJF-LAS, SJF-LDA, SJF-LGB, SJF-LN, SJF-MLP, SJF-NAV, SJF-PLY, SJF-RDG, SJF-RF, SJF-RBST, SJF-SVM, and SJF-XGB. Performance Metrics: Cost, Time, Energy, and LB are utilized to compare the developed algorithms with baseline SJF, along with comparing them within their respective SJF-ML categories. Dataset: The real-time Google Big Data Task (BDT) dataset, comprising tasks ranging from one hundred to one thousand across nineteen files, was computed using the SJF-ML and SJF algorithms. Experiment: Open-source CloudSim simulator with VM counts of 20, 40, 60, 80, and 100 were utilized to compute the BDTs, outputting results across the considered metrics. Results: The algorithms SJF-XGB and SJF-LN provided the best results, with SJF-DT, SJF-LAS, and SJF-LDA providing poor results. Findings: Hybridization of the cloud's scheduling algorithms with ML provides improved intelligence and performance, resulting in the evolution of the cloud.

Keywords: Cloud-Computing, Hybrid-Algorithm, Machine-Learning, Scheduling, SJF

1. Introduction

1.1. Problem Formulation

Cloud development is profound in today's world, where several users are inclined towards using it, rather than using alternative platforms for computing purposes (Chaudhary et al. and Chung et al., 2025). On the one hand, its computing power is impressive; however, the cloud encounters several limitations in handling its resources, where current scheduling methods struggle without any intelligence mechanism to handle Big Data Tasks (BDTs) (Chaudhary et al., 2025). Without any modern intelligence mechanism to deal with these uncertain BDTs, the current scheduling methods of the cloud will always output results whose highest threshold is limited (Kathole et al., 2025). This puts a pause on evolving the cloud, which is contrary to the current modern evolution trend observed in several systems through modern Machine Learning (ML) mechanisms. With the technological advancements and results obtained with ML techniques, the scheduling algorithms of the cloud need to be provided with its intelligence to cope up its performance-related gaps and ensure its performance is elevated, thereby providing an evolution to the cloud as compared with the classical

cloud systems (Kathole et al. and Liu, 2025). With the need for BDT computations in the modern day, the cloud faces decision-making challenges too, with its classical scheduling methods, which function without ML intelligence (Sanjalawe et al., Sonia et al., and Ye et al., 2025). Without any integration of ML methods, these current scheduling algorithms fail to withstand these challenging BDTs, leading to higher cost and time consumption (Ali et al. and Almurshed et al., 2024). Additionally, the existing cloud's scheduling methods do not possess a pattern recognition facility which includes detection of the irregularities in BDTs, trends, and classification of incoming BDTs which are to be computed, predicting execution times of BDTs, leading to an improper Load Balancing (LB) mechanism in the cloud, which often increases energy consumption (Alsubaei et al., 2024; Bartakke et al., 2024; and Hayyolalam et al., 2024). Hence, the author has focused on these issues to hybridize with the cloud's ideal performing Shortest Job First (SJF) scheduling algorithm with the vast range of ML category of algorithms to provide an improved version of scheduling policies. In this paper, the author presents sixteen SJF-ML scheduling methods across four ML categories by using the preemptive SJF scheduling mechanism of selecting the BDT with the least computing time among the rest and using ML intelligence to allocate the cloud's resources to compute the BDTs. Here, the hybridization process includes combining the approaches of SJF scheduling with the ML technique to improve the scheduling approach. These SJF-ML algorithms give the cloud a fair chance to make better decisions with the BDTs, output better results, and ultimately lead to its evolution through a systematic and balanced BDT computation. Lastly, the author has also proposed a second-level hybridization in this research paper by combining the best-performing algorithms in the first level from each SJF-ML category to form a second-level hybridization to further improve cloud performance as a part of future research direction.

Figure 1 shows the Venn diagram of combining SJF with ML intelligence to develop the SJF-ML algorithms for improving its performance and providing a next-level cloud-evolved computing environment.

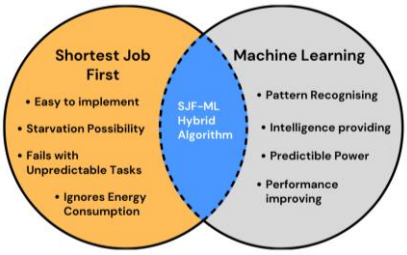

Figure 1. Individual characteristics SJF and ML algorithms for Cloud Evolution.

Figure 1 shows that the individual characteristics provided by the SJF scheduling algorithm are combined with those of ML algorithms to provide a hybridized stronger approach for better scheduling results.

1.2. Proposed Solution

The author proposes an experimental-based research work that includes hybridizing the best-performing cloud scheduling algorithm, Shortest Job First (SJF) in its preemptive form, with ML categories of algorithms to develop algorithms across four different ML algorithm categories, termed SJF-ML, to ensure higher performance is obtained and the cloud evolves through ML intelligence. Through this evolution, the proposed SJF-ML algorithms by the author ensure that a better Quality of Service (QoS) is provided to end users.

The author presents the following SJF-ML algorithm categories:

- **Shortest Job First - Classification Algorithm (SJF-CA) Category**: The classical SJF is combined with ML's Classification Algorithms (CA) to develop the SJF-CA category of algorithms.
- **Shortest Job First - Ensemble Learning Algorithm (SJF-ELA) Category**: The classical SJF is combined with ML's Ensemble Learning Algorithms (ELA) to develop the SJF-ELA category of algorithms.
- **Shortest Job First - Probabilistic Model (SJF-PM) Category**: The classical SJF is combined with ML's Probabilistic Model (PM) Algorithms to develop the SJF-PM category of algorithms.
- **Shortest Job First - Regression Algorithm (SJF-RA) Category**: The classical SJF is combined with ML's Regression Algorithms (RA) to develop the SJF-RA category of algorithms.

The author has explored the above four SJF-ML categories to combine the algorithms from each ML category with SJF in preemptive mode, which are suitable for scheduling mechanisms in the cloud

environment in their own unique way or working, leading to the development of sixteen SJF-ML hybrid algorithms as follows:

- **SJF-CA Algorithms:** The SJF is combined with four ML CAs algorithms to define the SJF-CA category. These include: Decision Tree (DT), which is ideal for rule-based VM allocation decisions at run-time; K-Nearest-Neighbors (KNN), which adapts to the dynamic workload of the BDTs using grouping techniques; Linear Discriminant Analysis (LDA), which is optimal for task prioritization with reduced dimensionality for the BDTs; and Support Vector Machine (SVM), which is effective for handling the non-linearity of the BDTs distributions (Talwani, S. et al. 2022 and Mo, Y. et al. 2015). These algorithms are hybridized with SJF to develop SJF-DT, SJF-KNN, SJF-LDA, and SJF-SVM hybrid SJF-CA scheduling algorithms, respectively.
- **SJF-ELA Algorithms:** The SJF is combined with four ML ELAs algorithms to define the SJF-ELA category. These include: AdaBoost (AB), which is effective in reducing the biasness of BDT computing time predictions; Light Gradient Boosting Machine (LGB), which has a high-speed memory efficient scheduling for the BDTs; Random Forrest (RF), which is useful with volatile workloads; and Extreme Gradient Boosting (XGB), which is highly useful for skewed BDT distributions (Liu, Z. 2025). These algorithms are hybridized with SJF to develop SJF-AB, SJF-LGB, SJF-RF, and SJF-XGB hybrid SJF-ELA scheduling algorithms, respectively.
- **SJF-PM Algorithms:** The SJF is combined with two PMs, including Bayesian Network (BAY), which uses probabilistic models to handle dependencies in the BDT scheduling process, and Naïve Bayes (NAV), which is lightweight in its nature and uses real-time probability for scheduling short-term BDTs (Chauhan, N. et al. 2022). These algorithms are combined with SJF to develop SJF-BAY and SJF-NAV hybrid SJF-PM scheduling algorithms, respectively.
- **SJF-RA Algorithms:** The six RAs, including Lasso (LAS) which is helpful in the high-dimensional and challenging BDTS, Linear (LN) which provides ideal real-time prediction for short BDTs, Multi-Layer Perceptron Neural Network (MLP) which captures the non-linearities useful for LB, Polynomial (PLY) which models the learning mechanism through its curvilinear BDT trends, Ridge (RDG) which takes care of multi-collinear VM features, and Robust (RBST) which ensures that no outlier BDTs are observed (Huymajer, M. et al. 2024). These algorithms are combined with SJF to develop SJF-LAS, SJF-LN, SJF-MLP, SJF-PLY, SJF-RDG, and SJF-RBST, respectively under the SJF-RA category.

Figure 2 represents the SJF-ML algorithms across the four categories.

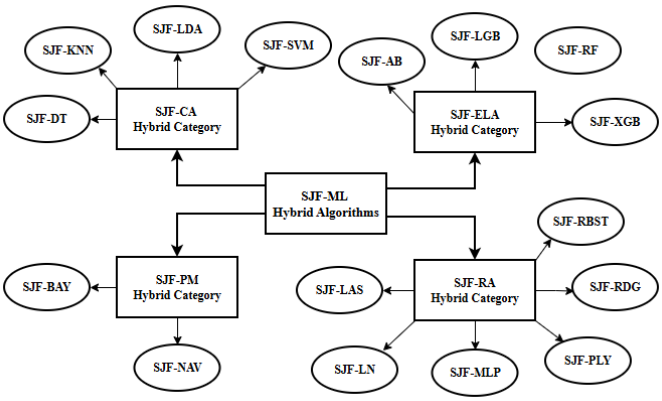

Figure 2. Proposed SJF-ML Algorithms.

2. Literature Review

2.1. Survey of Existing Work

This section includes the LR, where several authors have explored and provided various hybrid techniques using the SJF with other intelligent and prominent methods to focus on cloud limitations and its challenges. The authors of this paper have proposed an AI-enhanced framework that integrates deep workload prediction

and RL with SJF and traditional queueing theory to optimize results in the cloud environment (Chaudhary et al., 2025). This study presents a lightweight sampling fine-grained GPU scheduling method that uses suspend-resume mechanisms to reduce energy wastage in cloud systems (Chung et al., 2025). This work includes a detailed review of scheduling resources, which consists of a thorough explanation of scheduling techniques from the past decades and offering gaps from current research (Kathole et al., 2025). The ML's EL algorithms are overviewed to provide a clear pathway for their practical implementation in the cloud (Liu, 2025). This paper includes a review based on AI-based hybrid models, where the work analyses the need for energy-efficient systems for scheduling jobs along with providing future directions (Sanjalawe et al., 2025). The LB techniques are discussed in this research paper, where their significance in ML implementations is provided for multiple computing platforms (Sonia et al., 2025). The author of this research paper has focused on minimizing cloud performance parameters, such as cost and energy, by considering the sustainability of LB in the cloud (Verma, 2025). The Asynchronous Control-based Aggregation Transport Protocol (AC-ATP) rule is provided for reducing the deadlines in scheduling techniques by reducing the time performance metrics (Ye et al., 2025). A multiple-level fuzzy approach is integrated into the Internet of Things (IoT) platform to improve the operational efficiencies experienced by scheduling jobs in the healthcare domain (Ali et al., 2024). The authors have provided a framework named Enhanced Optimized-Greedy Nominator Heuristic (EO-GNH) for AI placement in computing sectors to improve time and resource allocation (Almurshed et al., 2024). The K-means technique is hybridized with Heterogeneous Earliest End Time (HEFT), providing a dual ML framework for task scheduling and improvements in results (Alsubaei et al., 2024). Zero-trust hybrid strategies have been provided by the authors to provide a structured evaluation framework for the cloud environment for selecting the best solutions and offering guidance on cost-effectiveness and implementation (Bartakke et al., 2024). The authors have presented a Chaos theoretical model hybridized with the Black Widow Optimization algorithm (CBWO) to focus on LB issues in cloud environments (Hayyolalam et al., 2024). A priority-based method in the cloud using DRL named PH-DRL is presented for better scheduling of jobs and improved time metrics (He et al., 2024). The authors have presented a scheduling technique for a fog-computing environment to improve QoS (Hosseini et al., 2024). A comparative evaluation of RAs is performed where seventeen RAs are studied in contrast with conventional methods (Huymajer et al., 2024). ML techniques focused on cloud optimization for better decision-making (Kanchetti et al., 2024).

This research work hybridized SJF with priority and deployed the Enhanced Shortest Job First with Priority (ESJFP) algorithm to improve classical SJF with task complexities having ML capabilities and improve resource and time metrics (Laha et al., 2024). A multi-objective flexible job-shop scheduling problem (MOFJSP) scheduler is presented in this paper to imbibe graphs with modern prominent methods for complex problem solutions (Li et al., 2024). A Graph-based Denoising Diffusion Probabilistic Model (G-DDPM) is presented for time-series forecasting, where the method works with PMs to reduce energy metrics (Miraki et al., 2024). The LB issues are handled using the presented Modified Parallel Particle Swarm Optimization (MPPSO) method for task scheduling, reducing the time parameters in the cloud (Pradhan et al., 2024). The author has implemented ML algorithms like DT, RF, and KNN to provide solutions related to posture classification, achieving better accuracy and Interpretability (Rahimi et al., 2024). The SJF is hybridized with a Multi-Level Memory-Based Framework (SJF-MMFB) for enhanced cloud scheduling by incorporating multi-queues addressing starvation Issues and improving fairness in cloud systems (Rekha et al., 2024). Hybrid ML models are implemented for cryptocurrency predictions and handling real-time predictions with achieved profitability (Salehi et al., 2024). Big data Integrated cloud systems are used to improve decision-making and operational performance for the ERP systems (Saraswat et al., 2024). The challenges associated with the cloud are focused on ML techniques to improve LB, the system's efficiency, and decision-making (Zende et al., 2024). The authors have proposed a dual-objective algorithm to improve results concerning the scheduling domain (Asghari et al., 2023). Post validations across twelve scenarios, the presented algorithm provides better results when compared to the other state-of-the-art algorithms. The author has used the SVM scheme in the two-cloud platform to reduce computing and communication overheads (Hu et al., 2023). Modern statistical models are utilized to investigate the cloud performance in this work (Kumar et al., 2024). This work surveys task scheduling methods and compares them with various performance parameters for optimal task management (Nayak et al., 2023). This work demonstrates the use of ML to improve real-time scheduling algorithms for cluster environments, optimizing task scheduling for dynamic environments (Zhang et al., 2023). This work explores how ML impacts energy optimization, dynamic load balancing, task scheduling, and security in cloud computing, improving resource allocation, security, and VM migration (Kumar et al., 2022). The EAs are utilized for the prediction of diseases in cocoa crops, offering early and accurate predictions based on climatic parameters, and benefiting farmers with proactive disease management (Olofintuyi et al., 2022).

This work evaluates hybrid scheduling and allocation algorithms in cloud computing, showing that combining SJF with RR results in the best response time, while SJF with a novel length-wise allocation (LwA)

provides the least CPU utilization (Sahkhar et al., 2022). The author has combined two optimization methods to strengthen the prominent cloud performance parameters and enhance the task scheduling process in the cloud (Verma, 2022). The authors have examined the use of RR and SJF algorithms for improving efficiency in cloud environments, suggesting an approach for optimal scheduling and resource allocation (Kumar et al., 2021). A detailed comparison is made with SJF and Longest Job First (LJF) with ML-based scheduling approaches using CloudSim, showing SJF's efficiency while highlighting ML's potential for further optimization in dynamic environments (Murad et al., 2021). This work reviews studies on ML in cloud security, identifying SVM as key and emphasizing hybrid methods for threat detection using metrics like True Positive Rate (Nassif et al., 2021). The authors propose hybrid SJF-Min-Min Best Fit (MMBF) and SJF-Extreme Learning Machine (ELM) scheduling models using Particle Swarm Optimization (PSO), optimizing job hosting and minimizing starvation in dynamic cloud settings (Rekha et al., 2021). This work combines SJF-MMBF with ELM energy-aware scheduling, addressing energy and security challenges (S Rekha et al., 2021).

This paper introduces a scheduling method for enhancing the task allocation process in cloud environments (Tanha, M. et al., 2021). The authors have designed an Enhanced SJF (ESJF) for stable task management by introducing ESJF with time-slicing between shortest and longest jobs, outperforming SJF in unstable environments by reducing starvation and delays (Younis et al., 2021). This work enhances VM scheduling using SJF-MMBF and SJF-ELM models with PSO, improving performance in high-load, dynamic cloud environments (Rekha et al., 2020). This work uses DRL and Long Short-Term Memory (LSTM) for scheduling, outperforming SJF, RR, and PSO by reducing CPU/RAM usage and task delays on real-world cloud workloads (Rjoub et al., 2020). A SJF-MMBF method is applied for efficient VM scheduling, combining SJF and MMBF with the Queue-Length MaxWeight policy to prevent starvation and improve job throughput (Guo et al., 2019). This work includes DRL used with LSTM to reduce CPU and RAM usage, outperforming SJF, RR, and PSO in big data task scheduling (Rjoub et al., 2019). CAs are discussed in this paper to ensure that better cloud system performance is achieved (Samie et al., 2019). A Q-learning-based method integrated with the heterogeneous earliest finish time (HEFT) is deployed to reduce makespan and response time, outperforming traditional scheduling algorithms in efficiency (Tong et al., 2019). This work compares heuristic SJF-MMBF with reinforcement-learning-based SJF-RL, showing SJF-RL excels in minimizing job delay and preventing starvation (Guo et al., 2018). This work includes an SRDQ algorithm combining SJF and RR with an adjustable time quantum to reduce time and avoid long task starvation (Elmougy et al., 2017).

The above-mentioned contributions from the researchers all around the globe who have presented their hybrid methodologies involving SJF. This work helped the author to understand how the classical and traditional SJF scheduling algorithms have been used with the modern ML methods. The next sub-section of the LR dives deeper into understanding the contributions of these authors and how they helped the author of this paper to contribute to improving cloud performance through hybridizations.

2.2. LR concerning Technical Overview

This sub-section includes the LR concerning Technical Overview, where Important aspects of author contributions are studied and represented in terms of the algorithm presented, its features, along with the enhancements it offers and limitations it possesses, presented in detail in Table 1.

Ref. No.	Algorithm	Features	Enhancements	Limitations
[1]	AI-enhanced framework	Dual-layer neural network	Adaptive scheduling	High training time, overhead
[2]	Lightweight sampling	Job migration, sampling	Boosts efficiency, fairness	Potential migration overhead
[3]	Meta-heuristic algorithms	Categorization, review	Identifies research gaps	Lacks novel implementation
[4]	Ensemble Learning	Combines multiple learners	Increases robustness, accuracy	Computationally intensive
[5]	AI-based hybrid models	AI decision-making cloud	Improves energy, fault handling	Complex adaptation to real-time
[6]	LB techniques	Real-time workload adaptation	Optimizes resource distribution	Hybrid LB in cloud
[9]	AC-ATP protocol	Aggregation and congestion	Reduces traffic, accelerates training	Straggler delays

[10]	Fuzzy Approach	IoT integration	Enhances healthcare predictions	Relies on expert labelling
[11]	EO-GNH	Optimized placement, parallelism	Boosts edge-cloud performance	Scaling complexity
[12]	K-means, HEFT	Task priority scheduling	Improves LB	Tuning required for K-values
[13]	Zero-Trust Framework	Identity/context verification	Fine-grained access, real-time policies	Legacy system updates needed
[14]	CBWO	Chaos theory, task distribution	Reduces energy usage	Needs parameter tuning
[15]	PH-DRL	DRL-based hybrid scheduler	Balances performance	Limited generalizability
[17]	Regression Models	Comparative feature study	Enhances prediction, tuning strategies	Risk of overfitting
[18]	General ML methods	Data retrieval, automation	Boosts decision-making, scalability	Infrastructure reliability
[19]	ESJFP	Prioritizes tasks	Reduces wait time	Lacks real-time adaptability
[20]	MO-GARL	Combines GAT with RL	Generalizable solutions obtained	Compute-heavy training
[21]	G-DDPM	Graph learning	Enhances accuracy	Resource-intensive
[22]	MPPSO	PSO for scheduling	Reduces delays, boosts throughput	Large-scale setting complexity
[23]	DT, RF, KNN	Time-frequency	Transparent feature influence	Inconsistent accuracy
[24]	SJF-MMFB with PSO	Integrates PSO, SJF	Balances load, prevents starvation	Overhead in large environments
[25]	ML methods	Applied to crypto price prediction	Combines fuzzy logic and ML	Some models over-fit
[26]	Big data with Cloud	Integration of big data in ERPs	Cost-effective	Lack of real-world case studies
[27]	ML methods	ML for LB	Improves cloud efficiency, security	Resource-demanding
[29]	SVM	Privacy-preserving ML	Resilience to malicious behaviour	Encryption complexity
[30]	Statistical Models	Survey of cloud apps	Insight into cloud app preferences	No qualitative analysis
[31]	Scheduling methods	Scheduling Methods Analysis	Optimize scheduling	Context-sensitive performance
[32]	FCFS, SJF	Predictive ML	Accurate burst time forecasts	Limited to real-time constraints
[33]	ML methods	Predictive ML	AI improves efficiency, and automation	Human expertise is still needed
[34]	Ensemble Learning	Time-series	High-accuracy forecasting	Region-specific data limitations
[35]	SJF, FCFS, RR, LwA	Various scheduling algorithms	Best response time with SJF + RR	Neglects other metrics
[36]	RR, SJF	Pre-emptive RR, SJF	Minimizes waiting time	High waiting time
[37]	SJF, LJF	SJF, LJF for Scheduling	Increases throughput	Starvation observed
[38]	SVM	Cloud security threats	Improve accuracy	High training time
[39]	PSO, SJF, MMBF, ELM	Hybrid queue scheduling	Prevents starvation	Scalability issues
[40]	SJF, MMBF, ELM	Hybrid scheduling	Improves efficiency	Complex hybrid approach

[42]	ESJF	Time-slice strategy	Reduces waiting times	Increased scheduling complexity
[43]	PSO, SJF, MMBF, ELM	Prevents starvation	Optimized scheduling performance	Computational overhead
[44]	RL, DQN, RNN-LSTM	LSTM	Reduces CPU/RAM usage	High computational expense
[45]	MMBF, SJF	SJF with QMW Scheduling	Improves fairness	Complexity in resource scheduling
[46]	DRL, LSTM	Predict VM assignment for tasks	Reduces resource consumption	Complex implementation
[47]	Classification Techniques	ML techniques for IoT	Improves data management	Scalability concerns
[48]	QL-HEFT	Combines Q-learning and HEFT	Improves makespan	High computational overhead
[49]	SJF-MMBF	Hybrid SJF with RL	Adapts to dynamic workloads	Adds computational complexity
[50]	SRDQ	Hybrid SJF and RR	Balances SJF and RR	Complexity in fine-tuning

Table 1. Literature Review concerning Technical Overview.

The conducted LR helped the author to identify the gaps and place the foundations for this research.

2.3. Gaps

The following gaps were identified from the above conducted LR:

- Existing studies have not fully developed hybrid scheduling methods that integrate a diverse range of ML algorithms beyond currently explored techniques, restricting cloud evolution.
- Current approaches lack the integration of BDT computations within ML-enhanced SJF, despite their need in the modern computing world.
- A comprehensive comparison between SJF and its improved versions is missing, particularly in assessing the entire spectrum of cloud performance metrics.
- Existing methods lead to increased energy consumption and expenses despite improved results.
- Contributing to a second level of hybridization using the best obtained multiple algorithms is absent in the same research contribution.

The identified gaps helped the author to develop the SJF-ML algorithms and make suitable contributions for evolving the cloud, further represented in section 3.

3. Contributions

The following includes the author's contributions from this research work:

- The SJF method has been enhanced through hybridization with four distinct and diverse categories of ML algorithms, resulting in the development of sixteen SJF-ML algorithms.
- Real-time BDTs were employed to evaluate the performance of the developed SJF-ML algorithms with baseline SJF, ensuring their applicability to evolve cloud environments.
- A comprehensive analysis was conducted to compare the SJF with hybridized SJF-ML methods, considering a broad spectrum of cloud performance metrics, including cost, time, energy, and LB.
- From the best-performing algorithm in each ML category, an architecture consisting of a second-level hybridization approach is provided to further refine scheduling efficiency and contrast its effectiveness against other SJF-ML variants.
- The proposed SJF-ML methodologies aim to enhance cloud performance, improve QoS, and contribute to more efficient, scalable scheduling solutions in high-demand computing environments.

The above contributions are made through this unbiased research work through extensive experiments in an open-source simulator where several real-time cloud components are deployed and BDTs are computed using all the developed SJF-ML algorithms.

4. Dataset, Experimental Setup and Performance Metrics

4.1. Dataset

Google's BDT was utilized by the classical SJF method along with the developed sixteen SJF-ML hybrid algorithms across four categories for computing. This dataset consists of nineteen input files across task lengths from one hundred tasks to one thousand tasks, each task having AT and CT as the significant features for the BDT. These features are fed to the SJF-ML algorithms to improve the cloud's scheduling decisions. The author has provided data analysis for the input BDTs in the form of descriptive statistics for these BDTs, including Mean (μ), Median (MDN), and Standard Deviation (σ) for a better understanding of the BDTs used. Table 2 includes the descriptive statistics for the BDTs.

Descriptive Statistics		**BDT-AT (s)**			**BGT-CT (s)**		
Sr. No.	**BDT Length**	**μ**	**MDN**	**σ**	**μ**	**MDN**	**σ**
1	100	543.1000	549.5000	273.6093	135.10	91.00	167.42
2	150	557.8733	587.0000	253.7945	147.87	97.00	185.55
3	200	573.3700	585.5000	275.2022	136.10	87.00	178.60
4	250	536.9560	537.5000	254.4490	119.91	89.00	152.95
5	300	545.1067	552.0000	265.7443	138.42	89.00	182.21
6	350	551.8571	527.0000	262.9433	135.05	95.00	162.92
7	400	552.8725	564.5000	263.0676	129.67	95.00	150.59
8	450	549.8956	560.5000	261.5785	123.83	91.00	151.95
9	500	556.3660	560.5000	258.0876	130.00	93.00	153.38
10	550	538.8127	527.5000	272.4410	130.29	93.00	168.00
11	600	542.4350	538.5000	258.2551	137.24	93.00	175.03
12	650	544.5800	549.5000	257.2584	136.16	93.00	168.88
13	700	550.8814	550.0000	255.6384	124.99	93.00	150.30
14	750	551.6360	545.5000	251.5622	136.07	93.00	171.25
15	800	560.9663	567.0000	256.9907	123.70	91.00	149.87
16	850	547.2741	537.5000	261.1188	125.12	91.00	149.47
17	900	542.2711	538.0000	256.2634	133.39	93.00	161.45
18	950	543.3053	544.5000	260.4067	124.61	94.00	150.46
19	1000	552.9770	552.5000	248.5605	129.66	93.00	162.63
	AVG	**549.6072**	**551.2895**	**260.3669**	**131.4305**	**92.3158**	**162.7847**

Table 2. Descriptive Statistics for the BDTs.

Table 2 shows that the descriptive statistics of the ATs possess a considerable amount of variability, making it difficult to predict the SJF and develop SJF-ML algorithms to be computed. On the other hand, the descriptive statistics of the BDA-CTs showcase a slight amount of stability with lower fluctuations, providing a fair chance to all the algorithms for computing. At a higher task length, both times showcase balanced behavior, suggesting there will be a steady execution with higher loads. These tasks were input into a real-time computing environment to the SJF and SJF-ML algorithms across several VM-based experimental scenarios to ensure unbiased results are obtained.

4.2. Working of SJF-ML hybrid algorithms

The SJF and ML algorithms integrate with each other to make the crucial scheduling decisions at runtime. The SJF-ML scheduling algorithms executed the BDTs based on the feature of BDT-CTs. The working mechanism includes loading the challenging BDTs in the cloud queue, followed by the feature extraction with

the label defined by the BDT-CTs. Further, the BDTs are split into 80 % training and 20 % testing sets and fed to each of the presented sixteen SJF-ML algorithms for training and testing in all the experimental scenarios. Later, the BDTs are sorted considering CTs and scheduled with the ideal available VM at runtime. Figure 3 represents the general block diagram of the working of all the SJF-ML algorithms.

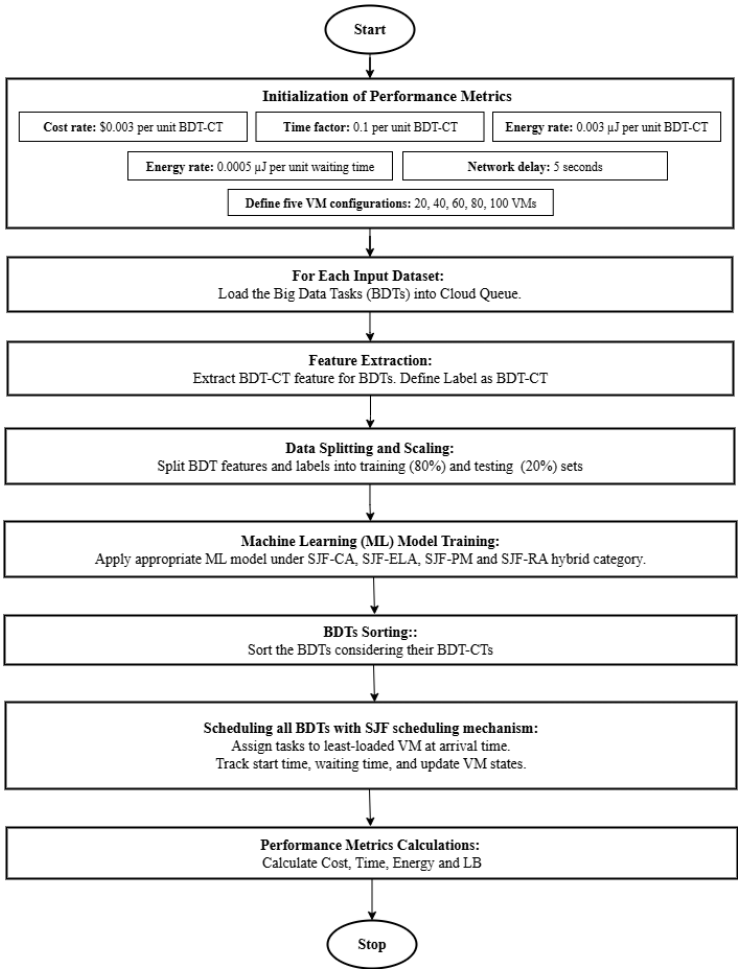

Figure 3. Block diagram of Working of the SJF-ML Algorithms.

5. Experimental Setup and Performance Metrics, and Architecture of Experiment

5.1. Experiment Setup and Performance Metrics

The experiment was set up in the open-source CloudSim environment where the classical SJF and sixteen developed SJF-ML hybrid algorithms across four SJF-ML categories were Incorporated In this simulation environment. The experiment simulates computing the BDTs in a multiple-VM environment. The performance metrics considered for the study are: cost ($), time (seconds 's'), energy (microJules 'μJ'), and LB. LB here is defined as a measurement that uses the coefficient of variation, considering workloads across all the VMs post-scheduling all the tasks, considering a certain scenario. All the SJF-ML algorithms provide dynamic LB, unlike the existing SJF, through enhancements of predictive intelligence mechanisms. The ML component during the scheduling process allocates VMs by considering their queue lengths, resource utilization patterns, and estimation of BDT-CTs, which is significant for SJF. Through continuous real-time feedback against

fluctuating workloads, the SJF-ML ensures a stable distribution of all BDTs evenly among the considered VMs. The computing cost rate was set to $0.003 per unit BDT-CT. The Execution Time Factor was set to 0.1 per unit BDT-CT. The Energy Consumption Rate and Idle Energy Consumption Rate were set to 0.003μJ per unit BDT-CT and 0.0005μJ per unit waiting time. With the possibility of errors and faults in the network, a constant network delay of 5s is considered. The experiment was conducted in five scenarios with the VM count as: 1: 20 VMs; 2:40 VMs; 3:60 VMs; 4:80 VMs, and 5:100 VMs. The main reason to choose these four metrics is that they cover the entire spectrum of parameters for the cloud. Time and Cost play a major role in scheduling in the cloud and are critical for judging and comparing algorithms. Energy ensures the cloud's resources are consumed optimally without having to sacrifice time and cost to do so. Lastly, each cloud resource needs to be utilized to the fullest, and an uneven workload allotment ensures high time, cost, and energy for the ones most used. Hence, LB as a metric is calculated in the form of a coefficient of variance which provides a balance to ensure appropriate energy is consumed with Ideal cost and time metrics. Together, these metrics form an Ideal comparative framework for the classical SJF to be compared with the developed SJF-ML hybrid algorithms. If a certain algorithm provides ideal results across all four metrics, then other metrics also significant to the cloud will be optimal.

5.2. Architecture of the Experiment

This sub-section provides the experiment's architecture with the deployed SJF-ML hybrid scheduling algorithms. The experiment was conducted where the BDTs were submitted for computing from the user environment to the cloud environment. Upon reaching the cloud's platform, the BDTs are added to the cloud queue, where they wait until their ATs are reached. This queue represents a storage from which the SJF-ML algorithm will choose the BDT for execution. This BDT's size acts like a main feature for the SJF-ML algorithms to select the tasks from the queue, considering its least CT. Here, the developed SJF-ML algorithms deployed will ensure appropriate pattern recognition through eradicating irregularities of the BDTs, classifying them according to their trends, thereby provide an intelligence mechanism before submitting the BDTs to the cloud VMs. From its initial submission until getting selected and assigned to being computed and finally getting completed and closed, a BDT undergoes several stages in its lifetime. Post all completion of BDT computations, the results are computed and provided to the user for the considered metrics. Figure 4 represents the architecture of the experiment with the deployment of SJF-ML hybrid scheduling algorithms.

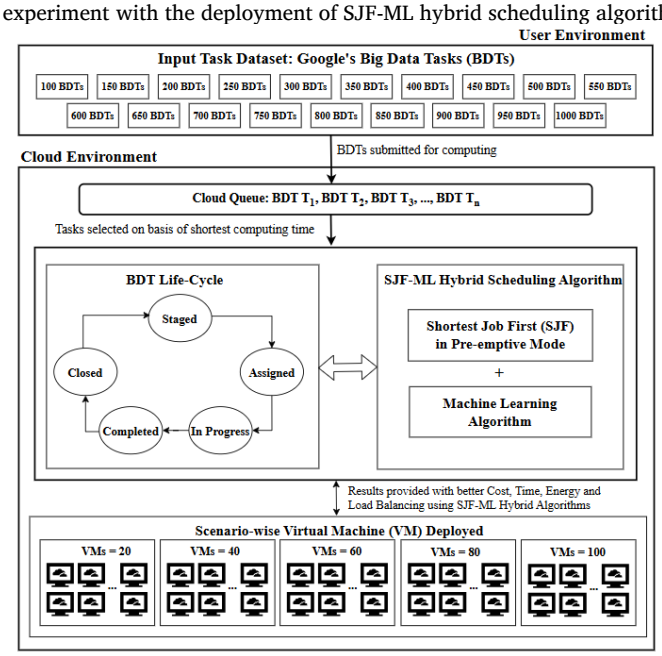

Figure 4. Architecture of the Experiment with SJF-ML Algorithms.

6. Results and Discussions

This section includes the results and discussions for the experiment conducted, presented in four sub-sections: 6.1, 6.2, 6.3, and 6.4, including cost, time, energy, and LB results compared in terms of average (AVG) and improvement percentage (IMPR %) compared against SJF AVG values. The performance rank of a certain algorithm is represented with a circled number, representing its performance rank within its respective SJF-ML hybrid category. Additionally, a green-circled asterisk is represented for the best performance, considering all SJF-ML algorithms and the SJF algorithm.

6.1. Cost Results

Table 3 includes the Cost Results for the developed SJF-ML hybrid algorithms compared to SJF.

Cost ($) Results		VM Counts					AVG	IMPR% vs SJF
Category	Algorithm	20	40	60	80	100		
Classical	SJF	3138.54	840.47	399.99	240.77	164.47	956.85	-
SJF-CAs vs SJF	SJF-DT④	5188.89	1280.00	569.58	328.71	212.67	1515.97	-58.4334
	SJF-KNN③	5028.80	1249.88	557.97	318.10	207.78	1472.51	-53.8914
	SJF-LDA①	3704.52	977.66	458.97	272.85	184.36	1119.67	-17.0163
	SJF-SVM②	3754.84	972.62	447.66	260.50	172.02	1121.53	-17.2106
SJF-ELAs vs SJF	SJF-AB④	4952.65	1236.87	557.00	318.68	209.15	1454.87	-52.0479
	SJF-LGB③	3704.52	977.66	458.97	272.85	184.36	1119.67	-17.0163
	SJF-RF②	3149.31	843.07	401.09	241.36	164.83	959.93	-0.3219
	SJF-XGB①	3138.55	840.46	399.99	240.77	164.47	956.85	0.0000
SJF-PMs vs SJF	SJF-BAY①	4532.35	1116.82	522.63	292.29	195.22	1331.86	-39.1921
	SJF-NAV②	4980.60	1240.91	555.99	317.71	207.68	1460.58	-52.6446
SJF-RAs vs SJF	SJF-LAS⑤	5069.30	1285.20	577.98	329.59	214.76	1495.37	-56.2805
	SJF-LN①⊛	3138.52	840.46	399.99	240.77	164.47	956.84	0.0010
	SJF-MLP④	3150.74	843.71	402.86	241.69	165.11	960.82	-0.4149
	SJF-PLY②	3138.56	840.47	399.99	240.77	164.47	956.85	0.0000
	SJF-RDG③	3142.46	841.42	400.39	240.99	164.60	957.97	-0.1171
	SJF-ROB②	3138.55	840.46	399.99	240.77	164.47	956.85	0.0000

Table 3. Cost Results for SJF-ML hybrid algorithms compared to SJF algorithm.

Table 3 shows the following cost comparison of the SJF-ML hybrid algorithms compared to the SJF algorithm:

- **SJF-CA vs SJF**: SJF > SJF-LDA > SJF-SVM > SJF-KNN > SJF-DT
- **SJF-ELA vs SJF**: [SJF ≈ SJF-XGB] > SJF-RF > SJF-LGB > SJF-AB
- **SJF-PM vs SJF**: SJF > SJF-BAY > SJF-NAV
- **SJF-RA vs SJF**: SJF-LN⊛ > SJF > [SJF-PLY ≈ SJF-ROB] > SJF-RDG > SJF-MLP > SJF-LAS

Figure 5 shows Cost IMPR % graph for SJF-ML Hybrid Algorithms compared to SJF Algorithm.

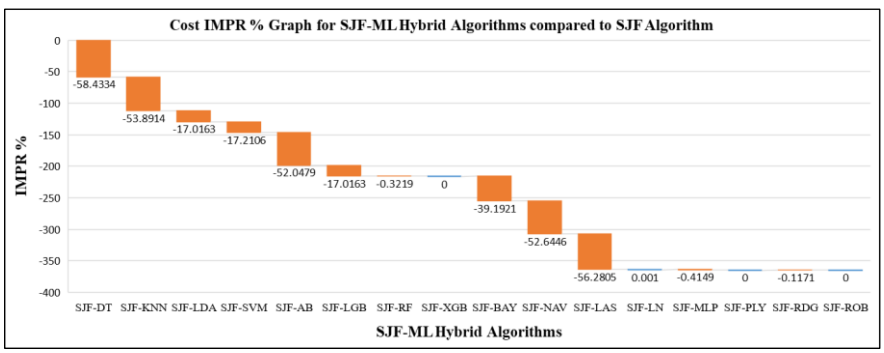

Figure 5. Cost IMPR % graph for SJF-ML Hybrid Algorithms compared to SJF Algorithm.

The cost results convey the SJF-LDA of the SJF-CA category, SJF-XGB of the SJF-ELA category, SJF-BAY of the SJF-PM category, and SJF-LN of the SJF-RA provide the best performance within their respective categories. Additionally, the SJF-LN hybrid algorithm of the SJF-RA category provides the best cost results across all the SJF-ML developed hybrid algorithms, along with the SJF algorithm.

6.2. Time Results

Table 4 includes the time comparison of SJF with presented SJF-ML Hybrid Algorithms.

Time (s) Results		VM Counts					AVG	IMPR% vs SJF
Category	Algorithm	20	40	60	80	100		
Classical	SJF⊛	104.62	28.02	13.33	8.03	5.48	31.90	-
SJF-CAs vs SJF	SJF-DT④	172.96	42.67	18.99	10.96	7.09	50.53	-36.8692
	SJF-KNN③	167.63	41.66	18.60	10.60	6.93	49.08	-35.0041
	SJF-LDA①	123.48	32.59	15.30	9.09	6.15	37.32	-14.5230
	SJF-SVM②	125.16	32.42	14.92	8.68	5.73	37.38	-14.6602
SJF-ELAs vs SJF	SJF-AB④	165.09	41.23	18.57	10.62	6.97	48.50	-34.2268
	SJF-LGB③	123.48	32.59	15.30	9.09	6.15	37.32	-14.5230
	SJF-RF②	104.98	28.10	13.37	8.05	5.49	32.00	-0.3125
	SJF-XGB①⊛	104.62	28.02	13.33	8.03	5.48	31.90	0.0000
SJF-PMs vs SJF	SJF-BAY①	166.02	41.36	18.53	10.59	6.92	48.68	-34.47
	SJF-NAV②	182.62	45.5	19.64	11.54	7.47	53.35	-40.2062
SJF-RAs vs SJF	SJF-LAS④	168.98	42.84	19.27	10.99	7.16	49.85	-36.0080
	SJF-LN①⊛	104.62	28.02	13.33	8.03	5.48	31.90	0.0000
	SJF-MLP③	105.02	28.12	13.43	8.06	5.50	32.03	-0.4059
	SJF-PLY①⊛	104.62	28.02	13.33	8.03	5.48	31.90	0.0000
	SJF-RDG②	104.75	28.05	13.35	8.03	5.49	31.93	-0.0940
	SJF-RBST①⊛	104.62	28.02	13.33	8.03	5.48	31.90	0.0000

Table 4. Time comparison of SJF with presented SJF-ML Hybrid Algorithms.

Table 4 shows the following time comparison of the SJF-ML hybrid algorithms compared to the SJF algorithm:

- **SJF-CA vs SJF**: SJF⊛ > SJF-LDA > SJF-SVM > SJF-KNN > SJF-DT
- **SJF-ELA vs SJF**: [SJF⊛ ≈ SJF-XGB⊛] > SJF-RF > SJF-LGB > SJF-AD
- **SJF-PM vs SJF**: SJF⊛ > SJF-BAY > SJF-NAV
- **SJF-RA vs SJF**: [SJF⊛ ≈ SJF-LN⊛ ≈ SJF-PLY⊛ ≈ SJF-RBST⊛] > SJF-RDG > SJF-MLP > SJF-LAS

Figure 6 shows Time IMPR % graph for SJF-ML Hybrid Algorithms compared to SJF Algorithm.

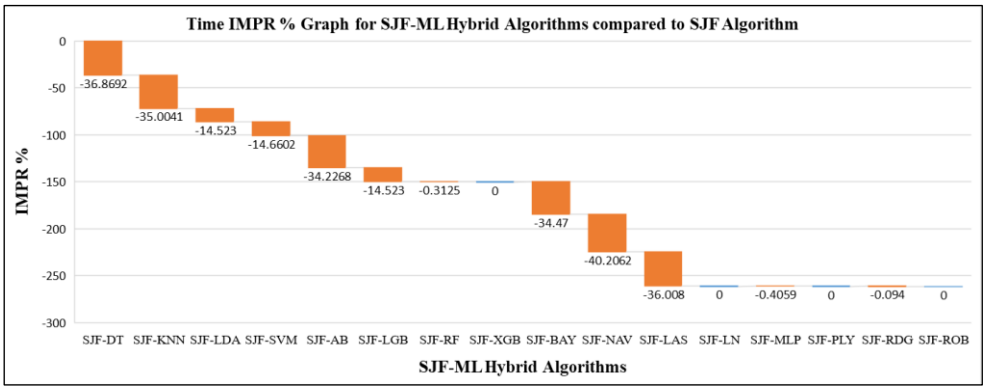

Figure 6. Time IMPR % graph for SJF-ML Hybrid Algorithms compared to SJF Algorithm.

The time results convey the SJF-LDA of the SJF-CA category, SJF-XGB of the SJF-ELA category, SJF-BAY of the SJF-PM category, and SJF-LN, SJF-PLY, and SJF-RBST algorithms of the SJF-RA provide the best performance within their respective categories. Additionally, the existing SJF algorithm, SJF-XGB of the SJF-SJF-ELA category, and SJF-LN, SJF-PLY, and SJF-RBST hybrid algorithms of the SJF-RA category provide the best time results across all the SJF-ML developed hybrid algorithms, along with the SJF algorithm.

6.3. Energy Results

Table 5 includes the energy comparison of SJF with presented SJF-ML Hybrid Algorithms.

Energy (μJ) Results		VM Counts					AVG	IMPR% vs SJF
Category	Algorithm	20	40	60	80	100		
Classical	SJF	835.24	296.15	170.71	118.17	89.84	302.02	-
SJF-CAs vs SJF	SJF-DT④	1176.96	369.41	198.98	132.82	97.87	395.21	-30.8556
	SJF-KNN③	1150.28	364.39	197.04	131.05	97.06	387.96	-28.4551
	SJF-LDA②	929.57	319.02	180.54	123.51	93.16	329.16	-8.9862
	SJF-SVM①⊛	845.06	271.73	147.69	98.23	72.52	287.05	4.9566
SJF-ELAs vs SJF	SJF-AB④	1137.59	362.22	196.88	131.15	97.29	385.03	-27.4849
	SJF-LGB③	929.57	319.02	180.54	123.51	93.16	329.16	-8.9862
	SJF-RF②	836.71	296.42	170.79	118.18	89.84	302.39	-0.1225
	SJF-XGB①	835.24	296.15	170.71	118.17	89.84	302.02	0.0000
SJF-PMs vs SJF	SJF-BAY①	1050.87	326.60	179.01	120.51	88.31	353.06	-16.8995
	SJF-NAV②	1142.25	362.89	196.71	130.99	97.04	385.98	-27.7995
SJF-RAs vs SJF	SJF-LAS④	1157.03	370.27	200.38	132.97	98.22	391.77	-29.7166
	SJF-LN①	835.23	296.15	170.71	118.17	89.84	302.02	0.0000
	SJF-MLP③	838.32	297.24	171.87	118.54	90.18	303.23	-0.4006
	SJF-PLY①	835.24	296.15	170.71	118.17	89.84	302.02	0.0000
	SJF-RDG②	835.89	296.31	170.78	118.2	89.86	302.21	-0.0629
	SJF-RBST①	835.24	296.15	170.71	118.17	89.84	302.02	0.0000

Table 5. Energy comparison of SJF with presented SJF-ML Hybrid Algorithms.

Table 5 shows the following energy comparison of SJF-ML hybrid algorithms compared to the SJF algorithm:

- **SJF-CA vs SJF**: SJF-SVM⊛ > SJF > SJF-LDA > SJF-KNN > SJF-DT
- **SJF-ELA vs SJF**: [SJF ≈ SJF-XGB] > SJF-RF > SJF-LGB > SJF-AB
- **SJF-PM vs SJF**: SJF > SJF-BAY > SJF-NAV
- **SJF-RA vs SJF**: [SJF ≈ SJF-LN ≈ SJF-PLY ≈ SJF-RBST] > SJF-RDG > SJF-MLP > SJF-LAS

Figure 7 shows Energy IMPR % graph for SJF-ML Hybrid Algorithms compared to SJF Algorithm.

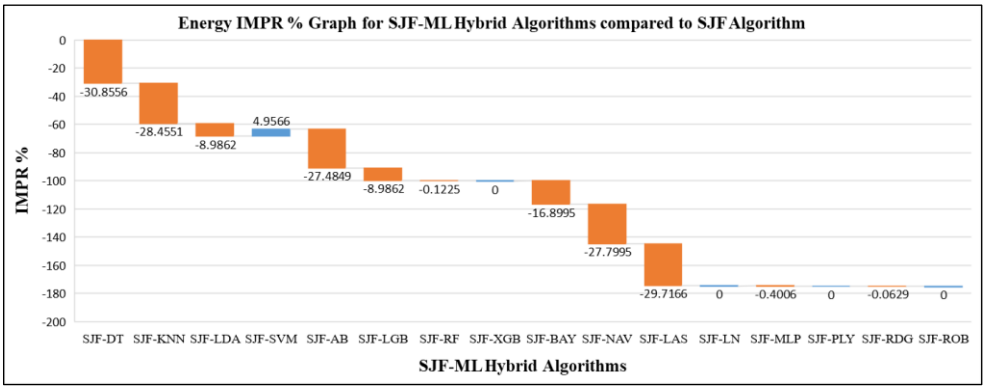

Figure 7. Energy IMPR % graph for SJF-ML Hybrid Algorithms compared to SJF Algorithm.

The energy results convey the SJF-SVM of the SJF-CA category, SJF-XGB of the SJF-ELA category, SJF-BAY of the SJF-PM category, and SJF-LN, SJF-PLY, and SJF-RBST hybrid algorithms of the SJF-RA provide the best performance within their respective categories. Additionally, the SJF-SVM hybrid algorithm of the SJF-CA category provides the best energy results across all the SJF-ML developed hybrid algorithms, along with the SJF algorithm.

6.4. LB Results

Table 6 includes the LB comparison of SJF with presented SJF-ML Hybrid Algorithms.

LB Results		VM Counts					AVG	IMPR% vs SJF
Category	Algorithm	20	40	60	80	100		
Classical	SJF	2.30	4.56	6.50	8.40	9.99	6.35	-
SJF-CAs vs SJF	SJF-DT②	1.48	3.08	4.8	6.31	7.83	4.70	25.9843
	SJF-KNN③	1.18	3.23	4.91	6.51	7.99	4.76	25.0394
	SJF-LDA④	2.21	4.5	6.51	8.48	10.16	6.37	0.3150
	SJF-SVM①⊛	0.23	0.75	1.24	1.62	1.14	1.00	84.2520
SJF-ELAs vs SJF	SJF-AB①	1.65	3.51	5.25	6.78	8.19	5.08	20.0000
	SJF-LGB④	2.21	4.5	6.51	8.48	10.16	6.37	0.3150
	SJF-RF②	2.29	4.54	6.47	8.37	9.96	6.33	0.3150
	SJF-XGB③	2.30	4.56	6.50	8.40	9.99	6.35	0.0000
SJF-PMs vs SJF	SJF-BAY①	1.48	3.28	5.03	6.61	8.04	4.89	16.3780
	SJF-NAV②	1.63	3.54	5.38	7.14	8.84	5.31	22.9921
SJF-RAs vs SJF	SJF-LAS①	1.23	2.79	4.44	6.07	7.70	4.45	29.9213
	SJF-LN④	2.30	4.56	6.50	8.40	9.99	6.35	0.0000
	SJF-MLP②	2.25	4.49	6.35	8.34	9.84	6.25	1.5748
	SJF-PLY④	2.30	4.56	6.50	8.40	9.99	6.35	0.0000
	SJF-RDG③	2.29	4.54	6.48	8.37	9.95	6.33	0.3150
	SJF-RBST④	2.30	4.56	6.50	8.40	9.99	6.35	0.0000

Table 6. LB comparison of SJF with presented SJF-ML Hybrid Algorithms.

Table 6 shows the following LB comparison of the SJF-ML hybrid algorithms compared to the SJF algorithm:

- **SJF-CA vs SJF**: SJF-SVM⊛ > SJF-DT > SJF-KNN > SJF > SJF-LDA
- **SJF-ELA vs SJF**: SJF-AB > SJF-RF > [SJF ≈ SJF-XGB] > SJF-LGB
- **SJF-PM vs SJF**: SJF-BAY > SJF-NAV > SJF
- **SJF-RA vs SJF**: SJF-LAS > SJF-MLP > SJF-RDG > [SJF ≈ SJF-LN ≈ SJF-PLY ≈ SJF-RBST]

Figure 8 shows LB IMPR % graph for SJF-ML Hybrid Algorithms compared to SJF Algorithm.

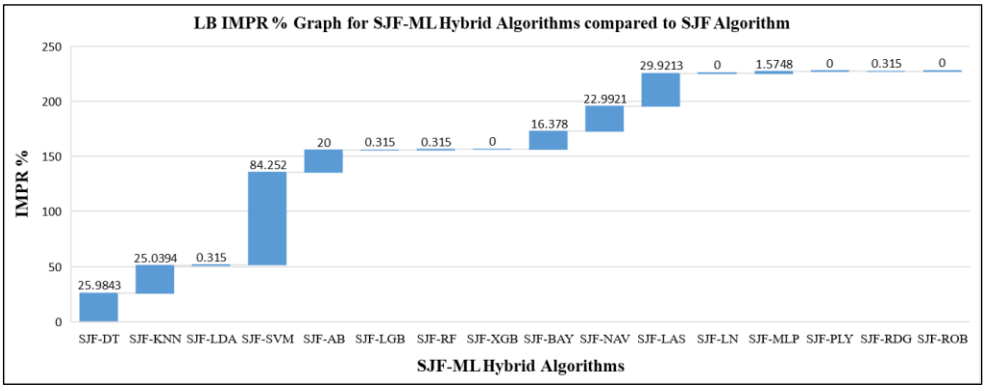

Figure 8. LB IMPR % graph for SJF-ML Hybrid Algorithms compared to SJF Algorithm.

The LB results convey the SJF-SVM of the SJF-CA category, SJF-AB of the SJF-ELA category, SJF-BAY of the SJF-PM category, and SJF-LAS of the SJF-RA provide the best performance within their respective categories. Additionally, the SJF-SVM hybrid algorithm of the SJF-CA category provides the best LB results across all the SJF-ML developed hybrid algorithms, along with the SJF algorithm.

7. Cumulative Results and Implications

This section includes filtering the better-performing SJF-ML hybrid algorithms obtained post the experiment along with their implications. Table 7 includes the comparative matrix representing the cumulative experimental results with performance of algorithms categorized as Best, Good, Average, Poor, or Worst.

Metric	Best	Good	Average	Poor	Worst
Cost	SJF SJF-XGB SJF-LN	SJF-BAY SJF-LDA SJF-PLY SJF-RF SJF-ROB	SJF-LGB SJF-NAV SJF-SVM SJF-RDG	SJF-KNN SJF-AB SJF-MLP	SJF-DT SJF-LAS
Time	SJF SJF-XGB SJF-LN SJF-PLY SJF-RBST	SJF-LDA SJF-RF SJF-BAY SJF-RDG	SJF-SVM SJF-LGB SJF-NAV SJF-MLP	SJF-KNN SJF-AB SJF-LAS	SJF-DT
Energy	SJF-SVM SJF SJF-XGB SJF-LN SJF-PLY SJF-RBST	SJF SJF-RF SJF-BAY SJF-RDG	SJF-LDA SJF-LGB SJF-NAV SJF-MLP	SJF-KNN SJF-AB SJF-LAS	SJF-DT
LB	SJF-SVM SJF-AB SJF-BAY SJF-LAS	SJF-DT SJF-RF SJF-NAV SJF-MLP	SJF-KNN SJF SJF-XGB SJF SJF-RDG	SJF SJF-LGB SJF-LN SJF-PLY SJF-RBST	SJF-LDA

Table 7. Comparative Matrix representing cumulative results for the experiment.

Table 7 shows that the developed SJF-ML hybrid algorithms, such as existing SJF, SJF-XGB of the SJF-ELA category, and SJF-LN of the SJF-RA category, provide the best cost results. The hybrid developed algorithms SJF-DT of the SJF-CA category and the SJF-LAS of the SJF-RA category are least suitable for the cost and can be avoided for hybridizations and implementations in the cloud. In terms of time performance metric, the existing SJF, along with SJF-XGB of the SJF-ELA category, SJF-LN, SJF-PLY, and SJF-RBST algorithms of the SJF-RA category provide the best results concerning time. Concerning time, the hybrid-developed SJF-DT algorithm of the SJF-CA category can be avoided for implementation since it provides poor results concerning time. In terms of energy, SJF-SVM of the SJF-CA category, existing SJF, SJF-XGB of the SJF-ELA category, and SJF-LN, SJF-PLY, SJF-RBST algorithms of the SJF-RA category provide the best results concerning energy. Like the time performance metric, the SJF-DT performance metric provides poor results concerning energy. Lastly, the SJF-SVM of the SJF-CA category, SJF-AB of the SJF-ELA category, SJF-BAY of the SJF-PM category, and SJF-LAS of the SJF-RA category provide the best results concerning LB. Here, the SJF-LDA of the SJF-CA category provides poor LB results. All the other developed hybrid algorithms provide either good, average, or poor performance compared within their SJF-ML categories, along with the SJF algorithm. Among the best SJF-ML performers, the SJF-XGB and SJF-LN algorithms perform better across maximum parameters, and the cloud can deploy these SJF-ML hybrid algorithms to ensure better results are obtained against the baseline of the SJF algorithm. Thus, the integration of ML with SJF provides fruitful results, improving the required cost, time, energy, and LB-considered performance metrics, and can be used to provide a better decision-making ability to the cloud to evolve itself from existing versions and provide better QoS.

8. Conclusion and Future Research Direction

This research paper provided hybrid scheduling methods by combining the currently existing best Shortest Job First (SJF) scheduling algorithm with sixteen Machine Learning (ML) algorithms. The ML algorithms within the categories of Classification Algorithms (CA), Ensemble Learning Algorithms (ELA), Probabilistic Models (PM), and Regression Algorithms (RA) were used to develop SJF-CA, SJF-ELA, SJF-PM, and SJF-RA as the four different categories of SJF-ML hybrid algorithms. From these sixteen SJF-ML hybrid categories, SJF-ML hybrid algorithms including SJF-AB, SJF-BAY, SJF-DT, SJF-KNN, SJF-LASSO, SJF-LDA, SJF-LGBM, SJF-LIN, SJF-MLPNN, SJF-NAVBAY, SJF-POLY, SJF-RDGE, SJF-RF, SJF-ROBST, SJF-SVM, and SJF-XGB are developed across the four SJF-ML categories. The main aim was to embed the cloud's scheduling process with ML techniques and improve the scheduling by enhancing the considered performance metrics such as cost, time, energy, and LB. The existing SJF algorithm was used as a baseline to compare it with all the developed SJF-ML algorithms across all the mentioned metrics. The real-time Google big data tasks were computed using all the SJF-ML hybrid algorithms across five different experimental scenarios of VM counts. From the experiment conducted, it can be conveyed that the SJF-XGB of the SJF-ELA category and the SJF-LN of the SJF-RA category provided better results across the maximum considered parameters. The cloud can deploy these variants for scheduling, rather than the SJF algorithm. A major limitation of the presented SJF-ML hybrid algorithms is the complexity of integration, where the coordination of both algorithms needs to be carefully executed. An additional limitation is the overhead of time spent in training the SJF-ML algorithms, which adds to the unwanted latency and energy consumption. The experiment findings conclude that the development of ML algorithms with scheduling approaches in the cloud provides better results through their improved decision-making, pattern recognition of BDTs, and overall allocation of resources, therefore providing a better, enhanced, and evolved version of itself. As a part of future research direction, the author aims to develop a second-level hybridization where the best performers, SJF-XGB and SJF-LN algorithms, will be integrated to ensure further improvements can be made for the cloud evolution. This second-level algorithm will be named SJF-XL, a combination of SJF-XGB and SJF-LN, where the individual first-level hybrid algorithm characteristics will be used to provide better and enhanced scheduling results to the first-level evolved cloud. Figure 9 represents the architecture where the SJF-XL second-level hybrid algorithm will be used by the cloud.

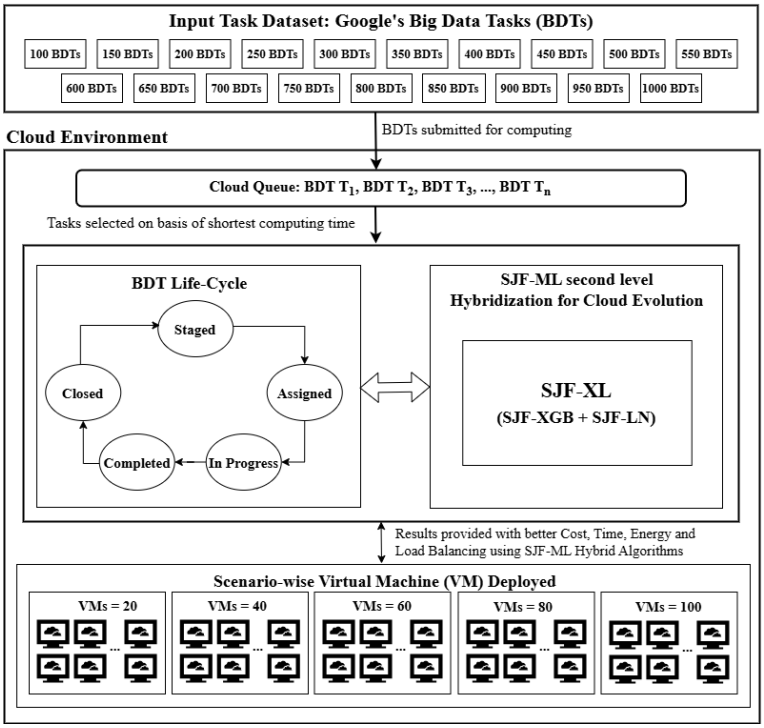

Figure 9. Second-Level SJF-ML Architecture.

References

Chaudhary, H., Sharma, G., Nishad, D. K., & Khalid, S. (2025). AI-enhanced modelling of queueing and scheduling systems in cloud computing. Deleted Journal, 7(4).

Chung, W., Tong, J., & Chen, Z. (2025). A fine-grained GPU sharing and job scheduling for deep learning jobs on the cloud. The Journal of Supercomputing, 81(2).

Kathole, A. B., Vhatkar, K., Lonare, S., & Kshirsagar, A. P. (2025). Optimization-based resource scheduling techniques in cloud computing environment: A review of scientific workflows and future directions. Computers & Electrical Engineering, 123, 110080.

Liu, Z. ". (2025). Ensemble Learning. In Springer (pp. 221–242).

Sanjalawe, Y., Al-E'mari, S., Fraihat, S., & Makhadmeh, S. (2025). AI-driven job scheduling in cloud computing: a comprehensive review. Artificial Intelligence Review, 58(7).

Sonia, N., & Nath, R. (2025). A systematic review of various load balancing approaches in cloud computing utilizing machine learning and deep learning. International Journal of Data Science and Analytics.

Verma, G. (2025). Sustainable Cost-Energy Aware load balancing in cloud environment using intelligent optimization. Sustainable Computing Informatics and Systems, 101115.

Ye, J., Peng, Y., Li, Y., Li, Z., & Huang, J. (2025). Asynchronous Control based Aggregation Transport Protocol for Distributed Deep Learning. IEEE Transactions on Computers, 1–14.

Ali, A. A., Umamaheswari, S., Khan, A. B. F., & Jayabrabu, R. (2024). Fuzzy rules-based Data Analytics and Machine Learning for Prognosis and Early Diagnosis of Coronary Heart Disease. Journal of Information and Organizational Sciences, 48(1), 167–181.

Almurshed, O., Kaushal, A., Meshoul, S., Muftah, A., Almoghamis, O., Petri, I., Auluck, N., & Rana, O. (2024). Enhancing performance of machine learning tasks on edge-cloud infrastructures: A cross-domain internet of things based framework. Future Generation Computer Systems, 166, 107696.

Alsubaei, F. S., Hamed, A. Y., Hassan, M. R., Mohery, M., & Elnahary, M. K. (2024). Machine learning approach to optimal task scheduling in cloud communication. Alexandria Engineering Journal, 89, 1–30.

Bartakke, J., & Kashyap, R. (2024). The usage of clouds in Zero-Trust security strategy. Journal of Information and Organizational Sciences, 48(1), 149–165.

Hayyolalam, V., & Özkasap, Ö. (2024). CBWO: A novel multi-objective load balancing technique for cloud computing. Future Generation Computer Systems, 107561.

He, H., Gu, Y., Liu, Q., Wu, H., & Cheng, L. (2024). Job scheduling in hybrid clouds with privacy constraints: A deep reinforcement learning approach. Concurrency and Computation Practice and Experience.

Hosseini, S. M., Shirvani, M. H., & Motameni, H. (2024). Multi-objective discrete Cuckoo search algorithm for optimization of bag-of-tasks scheduling in fog computing environment. *Computers & Electrical Engineering*, *119*, 109480.

Huymajer, M., Filzmoser, P., Mazak-Huemer, A., Winkler, L., & Kraxner, H. (2024). Opportunities and pitfalls of regression algorithms for predicting the residual value of heavy equipment — A comparative analysis. Engineering Applications of Artificial Intelligence, 141, 109599.

Kanchetti, D., Munirathnam, R., & Thakkar, D. (2024). Integration of Machine Learning Algorithms with Cloud Computing for Real-Time Data Analysis. Journal for Research in Applied Sciences and Biotechnology, 3(2), 301–306.

Laha, J., Pattnaik, S., Chaudhury, K. S., & Palai, G. (2024). Reducing makespan and enhancing resource usage in cloud computing with ESJFP Method: a new Dynamic Approach. Internet Technology Letters.

Li, Y., Zhong, W., & Wu, Y. (2024). Multi-objective flexible job-shop scheduling via graph attention network and reinforcement learning. The Journal of Supercomputing, 81(1).

Miraki, A., Parviainen, P., & Arghandeh, R. (2024). Probabilistic forecasting of renewable energy and electricity demand using Graph-based Denoising Diffusion Probabilistic Model. Energy and AI, 19, 100459.

Pradhan, A., Das, A., & Bisoy, S. K. (2024). Modified parallel PSO algorithm in cloud computing for performance improvement. Cluster Computing, 28(2).

Rahimi, N., Kamankesh, A., Amiridis, I. G., Daneshgar, S., Sahinis, C., Hatzitaki, V., & Enoka, R. M. (2024). Distinguishing among standing postures with machine learning-based classification algorithms. Experimental Brain Research, 243(1).

Rekha, S., Mohanapriya, D., Selvi, S., Banupriya, C., & Gobi, M. (2024). A Reinforced PSO Algorithm With SJF-MMBF For Allocating Work In Cloud Computing. IEEE, 1–6.

Salehi, S. (2024). Comparative analysis of machine learning techniques for cryptocurrency price prediction. Journal of Information and Organizational Sciences, 48(2), 341–352.

Saraswat, J. K., & Choudhari, S. (2024). Integrating big data and cloud computing into the existing system and performance impact: A case study in manufacturing. Technological Forecasting and Social Change, 210, 123883.

Zende, S., Singh, T., & Suryavanshi, M. (2024). Comprehensive review on Machine learning applications in Cloud Computing. International Journal of Innovative Research in Computer Science & Technology, 12(4), 16–24.

Asghari Alaie, Y., Hosseini Shirvani, M. & Rahmani, A.M. (2023). A hybrid bi-objective scheduling algorithm for execution of scientific workflows on cloud platforms with execution time and reliability approach. J Supercomput 79, 1451–1503.

Hu, C., Zhang, C., Lei, D., Wu, T., Liu, X., & Zhu, L. (2023). Achieving Privacy-Preserving and verifiable support vector machine training in the cloud. IEEE Transactions on Information Forensics and Security, 18, 3476–3491.

Kumar, R. (2023). Usage of Cloud Computing Application by Students from Kurukshetra University. Journal of Information and Organizational Sciences, 47(1), 189–201.

Nayak, A. A., & Shetty, S. (2023). A Systematic Analysis on Task Scheduling Algorithms for Resource Allocation of Virtual Machines on Cloud Computing Environments. IEEE Xplore, 1–6.

Zhang, R., Liu, Z., Tian, G., Lu, Y., & Sujatha, K. (2023). A Study of Real-Time Scheduling Algorithms in Cluster Environment Based on Machine Learning. IEEE, 682–686.

Chauhan, N., & Agrawal, R. (2022). Probabilistic Optimized Kernel Naive Bayesian Cloud Resource Allocation System. Wireless Personal Communications, 124(4), 2853–2872.

Kumar, Y., Kaul, S., & Hu, Y. (2022). Machine learning for energy-resource allocation, workflow scheduling and live migration in cloud computing: State-of-the-art survey. Sustainable Computing Informatics and Systems, 36, 100780.

Olofintuyi, S. S. (2022). Early Cocoa Blackpod Pathogen Prediction with Machine Learning Ensemble Algorithm based on Climatic Parameters. Journal of Information and Organizational Sciences, 46(1), 1–14.

Sahkhar, L., Balabantaray, B. K., & Yadav, S. S. (2022). Efficient cloudlet allocation to virtual machine to impact cloud system performance. International Journal of Information System Modeling and Design, 13(6), 1–21.

Talwani, S., Singla, J., Mathur, G., Malik, N., Jhanjhi, N. Z., Masud, M., & Aljahdali, S. (2022). Machine-Learning-Based Approach for Virtual Machine Allocation and Migration. Electronics, 11(19), 3249.

Verma, G. (2022). Hybrid optimization model for secure task scheduling in Cloud: Combining Seagull and Black Widow Optimization. *Cybernetics & Systems*, *55*(8), 2489–2511.

Kumar, S., & Dumka, A. (2021). Load Balancing with the Help of Round Robin and Shortest Job First Scheduling Algorithm in Cloud Computing. Springer Nature, 213–223.

Murad, S. A., Azmi, Z. R. M., Muzahid, A. J. M., & Al-Imran, M. (2021). Comparative Study on Job Scheduling Using Priority Rule and Machine Learning. IEEE Xplore, 1–8.

Nassif, A. B., Talib, M. A., Nasir, Q., Albadani, H., & Dakalbab, F. M. (2021). Machine Learning for Cloud Security: A Systematic Review. IEEE Access, 9, 20717–20735.

Rekha, S., & Kalaiselvi, C. (2021). Load balancing using SJF-MMBF and SJF-ELM approach. International Journal of Scientific Research in Computer Science Engineering and Information Technology, 74–86.

S, R., & C, K. (2021). Secure And Energy Aware Task Scheduling In Cloud Using Deep Learning And Cryptographic Techniques. ICTACT Journal on Communication Technology, 12(2), 2434–2441.

Tanha, M., Shirvani, M. H., & Rahmani, A. M. (2021). A hybrid meta-heuristic task scheduling algorithm based on genetic and thermodynamic simulated annealing algorithms in cloud computing environments. *Neural Computing and Applications*, *33*(24), 16951–16984.

Younis, M. F. (2021). ESJF algorithm to improve cloud environment. Iraqi Journal of Science, 4171–4180.

Rekha, S., & Kalaiselvi, D. (2020). Multi Level Queue Scheduling With Particle Swarm Optimization (Mlqs-Pso) Of Vms in Queueing Heterogeneous Cloud Computing Systems. International Journal of Recent Technology and Engineering (IJRTE), 8(5), 664–672.

Rjoub, G., Bentahar, J., Wahab, O. A., & Bataineh, A. S. (2020). Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems. Concurrency and Computation Practice and Experience, 33(23).

Guo, M., Guan, Q., Chen, W., Ji, F., & Peng, Z. (2019). Delay-Optimal Scheduling of VMs in a Queueing Cloud Computing System with Heterogeneous Workloads. IEEE Transactions on Services Computing, 15(1), 110–123.

Rjoub, G., Bentahar, J., Wahab, O. A., & Bataineh, A. (2019). Deep Smart Scheduling: A Deep Learning Approach for Automated Big Data Scheduling Over the Cloud. IEEE Xplore, 189–196.

Samie, F., Bauer, L., & Henkel, J. (2019). From Cloud Down to Things: An overview of Machine learning in Internet of Things. IEEE Internet of Things Journal, 6(3), 4921–4934.

Tong, Z., Deng, X., Chen, H., Mei, J., & Liu, H. (2019). QL-HEFT: a novel machine learning scheduling scheme base on cloud computing environment. Neural Computing and Applications, 32(10), 5553–5570.

Guo, M., Guan, Q., & Ke, W. (2018). Optimal scheduling of VMs in queueing cloud computing systems with a heterogeneous workload. IEEE Access, 6, 15178–15191.

Elmougy, S., Sarhan, S., & Joundy, M. (2017). A novel hybrid of Shortest job first and round Robin with dynamic variable quantum time task scheduling technique. Journal of Cloud Computing Advances Systems and Applications, 6(1).

Mo, Y., Kontonatsios, G., & Ananiadou, S. (2015). Supporting systematic reviews using LDA-based document representations. Systematic Reviews, 4(1).

Reiss, C., Tumanov, A., Ganger, G. R., Katz, R. H., & Kozuch, M. A. (2012). Heterogeneity and dynamicity of clouds at scale. *ACM*.