



AI-Driven VM Load Balancing Using Fuzzy Logic and Support Vector Machine

Ali Kidar^{1*}, Mostefa Hamdani², Abdelmalek Douara³

¹ Faculty of Science and Technology, University El Wancharissi of Tissemsilt – Tissemsilt, 38000, Algeria

* Corresponding Author Email: a.kidar@univ-tissemsilt.dz, - ORCID: [0009-0002-1084-201X](https://orcid.org/0009-0002-1084-201X)

² Laboratory of Electronics, Computer Science, and Applied Mathematics Faculty of Science and Technology, University El Wancharissi of Tissemsilt – Tissemsilt, 38000, Algeria

Email: mustapha.hamdani@gmail.com - ORCID: [0000-0002-8710-283X](https://orcid.org/0000-0002-8710-283X)

³ Laboratory of Electronics, Computer Science, and Applied Mathematics Faculty of Science and Technology, University El Wancharissi of Tissemsilt – Tissemsilt, 38000, Algeria

Email: douara.abdelmalek@univ-tissemsilt.dz - ORCID: [0009-0006-1086-201X](https://orcid.org/0009-0006-1086-201X)

Article Info:

DOI: 10.22399/ijcesn.5290

Received: 13 December 2025

Revised: 29 March 2026

Accepted: 28 April 2026

Keywords

Cloud Computing
Load Balancing
Fuzzy Logic
Support Vector Machine
SVM
Virtual Machine
Resource Optimization

Abstract:

Cloud computing has become an important solution for providing flexible and on-demand computing resources. As the number of users and requests increases, load balancing becomes a key challenge because tasks must be assigned to virtual machines in a way that reduces response time and improves resource use. This study proposes an intelligent Active VM Load Balancing algorithm based on Fuzzy Logic and Support Vector Machine, SVM. Fuzzy Logic is used to evaluate virtual machines according to their main characteristics, including memory size, processor speed, and number of processors. Then, SVM is applied to classify the most suitable virtual machine for each incoming task. The proposed approach aims to improve task allocation, avoid overloaded and underloaded virtual machines, and enhance overall cloud performance. The method is evaluated using CloudAnalyst and compared with common load balancing algorithms such as Round Robin, Throttled and Equally Spread Current Execution Load. The results show that the proposed Fuzzy-SVM approach improves response time, data center processing time, and resource utilization, making it a promising solution for smarter cloud load balancing.

1. Introduction

With the rapid development of Internet technology in recent years, Cloud Computing has been widely adopted as one of the most popular paradigms in both industrial and academic fields. The main idea of Cloud Computing is to provide services according to user demands on a pay-as-you-go basis, through the sharing of hardware resources and software [1]. Cloud computing faces many challenges as the number of users continues to grow, since the demand for resource sharing and resource use is increasing rapidly. Therefore, load balancing between resources is an important challenge. Load balancing distributes the workload across multiple entities, which helps achieve optimal resource utilization, maximize throughput, minimize response time, and

avoid overload [2]. Fuzzy logic, which is viewed as an extension of classical logic systems, provides an effective conceptual framework for addressing the problem of knowledge representation in an environment marked by uncertainty and imprecision [3]. Despite the large number of research works conducted in the field of Cloud Computing, it still faces several current issues and challenges. One of the most important challenges is load balancing, which requires optimization methods and models to improve performance and satisfy users. Machine learning techniques are useful for creating groups of virtual machines according to their hardware characteristics, such as RAM, processors, and other resources [4]. Support Vector Machine, SVM, is a supervised machine learning technique used for classification and prediction tasks. It works by

finding the best boundary that separates data into different classes while keeping the maximum distance between them. This makes SVM effective when dealing with complex data and limited training samples. In cloud load balancing, SVM helps classify virtual machines or predict the most suitable one for each incoming task based on features such as memory, processor speed, number of processors, and current load. Different load balancing algorithms have been proposed in Cloud Computing to optimize performance parameters. However, no algorithm provides a fully efficient solution to these problems because they are NP-complete [5]. The load balancing problem is highly complex and is known to be NP-complete. It is necessary, as far as possible, to avoid situations where some nodes are overloaded while others remain underloaded or completely idle. Although load balancing has been widely studied, current load balancing strategies still require optimization methods and models to further improve execution performance and meet user needs. Organizing nodes according to their hardware characteristics, such as processors, RAM, and other resources, and assigning tasks to virtual machines based on the performance of each node is one of the techniques widely adopted in different load balancing algorithms. This technique may be improved by considering additional parameters, such as the number of resource allocations, and by using artificial intelligence approaches, including machine learning algorithms. In this paper, we combine the techniques mentioned above to design a load balancing algorithm based on fuzzy logic and SVM. The proposed approach assigns tasks to the most efficient virtual machines, while also considering those with a lower number of allocations. We compared the proposed algorithm with Round Robin, Throttled, and Equally Spread Current Execution Load algorithms. In this comparison, we considered the following main criteria: response time, data center processing time, and the total cost of virtual machines. The proposed algorithm was implemented in Java using the CloudAnalyst simulator. The experimental results show that the proposed algorithm is highly competitive compared with existing approaches. The remainder of this paper is divided into five sections. The second section introduces the related work and the third section depicts the proposed approach. In the fourth section, we compare the proposed algorithm against Equally Spread Current Execution Load, Throttled algorithm and Round Robin algorithm. The fifth section displays the detailed analysis of the proposed approach. Finally, the sixth section is dedicated to a conclusion and some future works.

2. Literature Review

Load balancing is a set of techniques used to distribute a workload among different machines in a distributed computing system in order to minimize execution time and improve resource utilization [6], while avoiding situations where some nodes are heavily loaded and others remain idle or perform little work. It is therefore a resource optimization problem that aims to manage available computing resources according to the requirements of the computation to be performed or the characteristics of the tasks to be executed [7]. Many load balancing algorithms have been developed to distribute workloads evenly across computing nodes. In this section, we discuss selected studies on load balancing that have been conducted to improve cloud performance. Weighted Round Robin (WRR), derived from Round Robin, RR, provides better management of servers with different processing capacities. It assigns higher weights to servers with better performance in order to send a larger number of requests to different virtual machines, proportionally to the assigned weights and in a circular order. The WRR algorithm improves the critical limitation of the RR algorithm, but it still fails to fully balance the load among servers [8] [9]. Opportunistic Load Balancing (OLB), is one of the static load balancing algorithms. Its main objective is to keep every cloud node busy, but it does not consider the current load of each node. It also does not take into account the execution time on the node, which may lead to slower task processing [10]. Weighted Least Connection is an improvement of the Least Connection algorithm, where the weights assigned to the different services represent their processing performance. The default server weight is 1, and the system administrator can dynamically define the server weights. This algorithm assigns a weight value to each active server. New requests are automatically assigned to the servers whose ratio between active connections and assigned weight indicates the lowest load volume [10]. Throttled Load Balancing Algorithm is a dynamic algorithm that is fully deployed on virtual machines. Virtual machines are indexed in a table according to their availability status, Available or Busy. When the data center receives a request, called a cloudlet, the load balancer scans the table and returns the ID of the first available virtual machine. If all virtual machines are busy, the value "0" is returned. Once the task execution is completed, the resource is released. This algorithm does not perform efficiently when virtual machines have different hardware architectures or performance capacities [12]. Equally Spread Current Execution Load, ESCE, also known as the Active VM Load

Balancing algorithm, is based on the spread spectrum technique. The load balancer maintains an index table of virtual machines, along with the number of requests currently assigned to each virtual machine, VM. The Active VM Load Balancing algorithm distributes the data center workload by checking the load size and transferring requests to the lightly loaded virtual machine [13] [14]. This algorithm identifies the virtual machine with the lowest number of allocations, ensuring that active tasks are evenly distributed among the virtual machines. It first allocates virtual machines that are in a free state. If all virtual machines are already allocated, the algorithm selects the VM with the minimum number of allocations [15]. Adaptive algorithms are load balancing algorithms that consider different system parameters during resource allocation. This approach adapts load distribution to changes in the system state by adjusting its parameters dynamically. It provides better performance when the system changes frequently, which makes it suitable for widely distributed environments such as Cloud Computing[7]. Examples of adaptive load balancing algorithms include genetic algorithms [16], MapReduce-based approaches [17], resource scheduling algorithms [18], adaptive dynamic VM migration [19], adaptive task allocation schedulers [20], and ALBA, Adaptive Load Balancing Algorithm [21]. Although several studies exist in the literature, these methods still do not consider all the characteristics of the cloud computing environment. There is still a need to improve performance in several aspects. Therefore, existing systems offer opportunities to improve resource utilization, response time, and cost

3. Methodology

The objective of load balancing is to distribute the workload efficiently among the available resources in order to maximize resource performance and minimize the response and processing time of client requests. It also aims to avoid situations where some nodes are overloaded while others remain idle. Existing load balancing techniques consider different parameters, such as performance, response time, which refers to the time between sending a request and receiving a response, throughput, resource utilization, fault tolerance, migration time, and the overhead associated with this technology. These parameters are essential for selecting the appropriate server to process a new task, which is known as the server selection policy. The overall system architecture consists of three main components, as shown in Figure 1 .

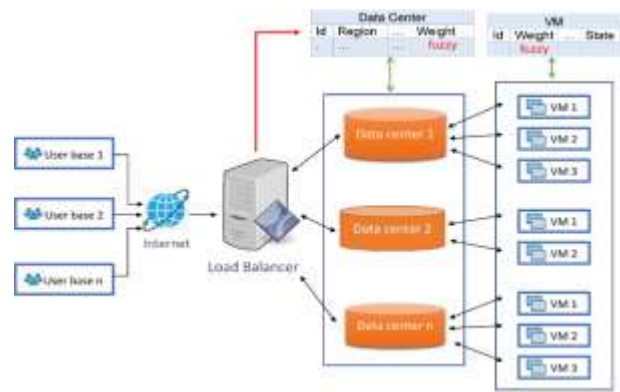


Figure 1. Architecture of the Proposed Framework [4]

The proposed approach operates in two main phases:

- Phase I: Virtual machine weighting. In this phase, fuzzy logic is applied to assign a weight to each virtual machine based on its characteristics.
- Phase II: Virtual machine classification. In this phase, SVM is used to classify the virtual machines and identify the most suitable one for task allocation.

This architecture assigns tasks to the most appropriate virtual machine according to hardware characteristics such as memory size, processor speed, number of processors, and other performance-related parameters.

A. Virtual Machine Prioritization

In this section, we detail the main steps involved in designing the fuzzy controller used to assign a weight to each virtual machine according to its characteristics: processor speed, number of processors, and memory size, RAM. The main question is how to determine the appropriate weight for each virtual machine. Assigning weights based only on server computing power ignores the fact that different requests may have different processing requirements. Although this approach is simple and intuitive, it does not guarantee optimal system performance. Fuzzy logic is one of the most effective methods for controlling processes, especially when they are too complex for conventional quantitative techniques, when the available information is partial or imprecise, or when it involves uncertainty. In our approach, the performance of virtual machines is treated as an imprecise concept. The relationship between memory size, number of processors, and processor speed is represented using linguistic variables in order to estimate the performance of each node. To model this relationship, we propose a fuzzy logic-based formalism to define the weights of the different virtual machines.

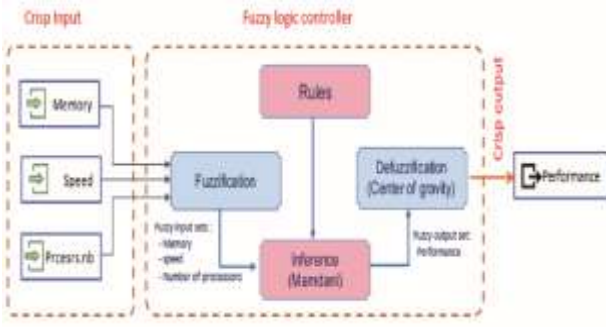


Figure 2. Fuzzy Controller

As shown in Figure 2, the proposed model has three inputs: memory size, processor speed, and the number of processors, and one output, which represents the weight assigned to each virtual machine. Fuzzification converts numerical values into linguistic variables using membership functions. In this study, we used trapezoidal membership functions for the input variables. To represent the different parameters, we defined five fuzzy sets for each linguistic variable: memory (verySmall, small, medium, large, veryLarge), processor speed (verySlow, slow, medium, fast, veryFast), number of processors (veryLow, low, medium, high, veryHigh) and vmPerformance (veryLow, low, medium, high, veryHigh).

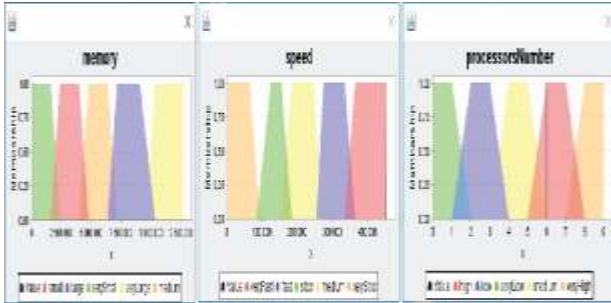


Figure 3. membership functions

Figure 3.5 shows the membership functions generated by our program.

Several rules are required to define the actions of the fuzzy controller. The rule base consists of fuzzy propositions, where each proposition is expressed in linguistic form. Using the rule-based structure of fuzzy logic, a set of IF-THEN rules is defined to generate the output response according to the input conditions. During rule evaluation, the min value was used as the activation method, while the max value was used as the accumulation method. The inference engine manages the way these rules are combined. In this approach, the Mamdani Fuzzy Inference System was used, and it is expressed as follows:

$$\mu_{R_M}(x, y) = \min[\mu_A(x), \mu_B(y)] \quad (1)$$

where $\mu(x)$ represents the membership value. Defuzzification is the process of converting the fuzzy output set into a single numerical value. The defuzzification method used for this transformation is the Center of Gravity, COG, method, defined by [22] :

$$\mu_{COG} = \frac{\int_U y \cdot \mu(y) dy}{\int_U \mu(y) dy} \quad (2)$$

where y is a point in the universe of discourse, $\mu(y)$ is the membership value of the resulting conclusion set, and U represents the universe of discourse.

B. Classification of VMs Using SVM

After assigning a fuzzy weight to each virtual machine, the next step is to classify the virtual machines using the Support Vector Machine, SVM, algorithm. The fuzzy controller evaluates each VM according to its hardware characteristics, such as memory size, processor speed, and number of processors. These values are then used as input features for the SVM model. SVM is a supervised machine learning algorithm used for classification [23]. Its main objective is to find the best separation boundary between different classes. In the proposed approach, SVM classifies virtual machines according to their suitability for executing incoming tasks. Each VM is represented by a set of features, including memory, processor speed, number of processors, fuzzy weight, current status, and number of active allocations. The classification process helps the load balancer identify the most appropriate virtual machine for each new request. Virtual machines with higher performance, lower allocation count, and available status are given higher priority. This reduces the risk of assigning tasks to overloaded machines and improves the distribution of workload across the cloud environment. SVM learns from previous examples and classifies new cases based on learned patterns. This makes the proposed approach more suitable for intelligent task allocation, especially when the system needs to distinguish between suitable and unsuitable virtual machines. The final decision is made by selecting the virtual machine classified as suitable and having the

lowest number of current allocations. If several virtual machines belong to the same suitable class, the VM with the highest fuzzy weight is selected. This strategy improves response time, avoids overload, and supports better resource utilization. A simple formulation of the input vector is:

$X = (\text{Memory}, \text{ProcessorSpeed}, \text{NumberOfProcessors}, \text{VMWeight}, \text{AllocationCount}, \text{VMStatus})$

The SVM model returns the class of the virtual machine:

$$\text{Class}(\text{VM}) \in \{\text{Low}, \text{Medium}, \text{High}\}$$

In this way, the proposed Fuzzy-SVM approach combines fuzzy logic for performance evaluation and SVM for intelligent virtual machine classification. This improves the decision-making process of the load balancer and helps assign tasks to the most efficient virtual machines.

C. Comparative Performance Study

This section presents the experimental results obtained after executing the proposed load balancing algorithm and comparing it with three existing algorithms: Equally Spread Current Execution Load, ESCE, Round Robin, and Throttled. The comparison is based on response time, data center processing time, request servicing time, and total virtual machine cost.

We used the latest stable versions of the required tools to evaluate our algorithm. The implementation was carried out under Windows 11 using Eclipse IDE 2026-03 and Java JDK 25 LTS. The fuzzy logic-based weight calculation was implemented using jFuzzyLogic, while the SVM classification model was developed with Weka 3.8.7. Finally, JFreeChart 1.5.6 was used to display the graphs generated after running the different algorithms.

Table 1. Response Time and DC Processing Time

Algorithm	Average Response Time (ms)	Data Center Processing Time (ms)
Proposed Fuzzy-SVM	57.32	0.51
ESCE	57.47	0.61
Round Robin	57.51	0.65
Throttled	57.49	0.65

Table 1 shows that the proposed Fuzzy-SVM approach is expected to achieve the lowest average response time, with 57.32 ms, and the lowest data center processing time, with 0.51 ms. This expected improvement comes from selecting virtual machines according to both performance and suitability for task execution.

Table 2. Data Center Request Servicing Time

Data Center	Proposed Fuzzy-SVM	ESCE	Round Robin	Throttled
DC1	0.47	0.44	0.48	0.45
DC2	0.55	0.64	0.70	0.75
DC3	0.49	0.53	0.48	0.51
DC4	0.59	0.58	0.56	0.60
DC5	0.58	0.62	0.77	0.71
DC6	0.57	0.97	0.97	0.95
DC7	0.40	0.41	0.74	0.72
DC8	0.42	0.41	0.38	0.37
Total	4.07	4.60	5.08	5.06

Table 2 compares the data center request servicing time. The proposed Fuzzy-SVM approach is expected to obtain the lowest total value, with 4.07 ms, compared with 4.60 ms for ESCE, 5.08 ms for Round Robin, and 5.06 ms for Throttled.

Table 3. Overall Processing Cost

Data Center	Proposed Fuzzy-SVM	ESCE	Round Robin	Throttled
DC1	2.04	2.26	2.31	2.33
DC2	1.86	2.45	2.50	2.34
DC3	1.70	2.50	2.39	2.54
DC4	1.45	2.10	2.09	2.02
DC5	1.97	2.26	2.19	2.22
DC6	1.32	1.56	1.63	1.68
DC7	0.37	0.59	0.59	0.59
DC8	0.19	0.59	0.59	0.59
Total	10.90	14.31	14.29	14.31

Table 3 presents the overall processing cost. The proposed Fuzzy-SVM algorithm is expected to reduce the total processing cost to 10.90, while

ESCE, Round Robin, and Throttled record 14.31, 14.29, and 14.31 respectively. Overall, the expected results indicate that the proposed Fuzzy-SVM approach can improve task allocation by combining fuzzy logic-based VM weighting with SVM-based virtual machine classification. This allows the load balancer to select more suitable virtual machines, reduce overload, improve response time, and optimize resource utilization.

3. Conclusion

Load balancing remains an important research area in Cloud Computing because it directly affects resource utilization, response time, throughput, and user satisfaction. Its main role is to detect overloaded and underloaded nodes, then distribute workloads between them in a balanced way. Although many load balancing algorithms have been proposed in the literature, the problem remains difficult because task allocation in cloud environments is highly complex and belongs to NP-complete problems. In the proposed work, we introduced an intelligent load balancing approach based on fuzzy logic and Support Vector Machine, SVM. Fuzzy logic was used to evaluate the performance of virtual machines according to their hardware characteristics, such as memory size, processor speed, and number of processors. Then, SVM was used to classify the most suitable virtual machine for each incoming task. The aim was to assign requests to efficient machines while avoiding overload and improving resource usage. The experimental comparison with Round Robin, Throttled, and Equally Spread Current Execution Load showed the effectiveness of the proposed approach in terms of response time, data center processing time, and processing cost. The results indicate better task distribution and improved cloud system performance. Future work should extend the proposed model by including additional parameters such as bandwidth, energy consumption, task priority, and network delay. Testing the approach in larger and more dynamic cloud environments would also strengthen its practical value.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper

- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

- [1] M. Hamdani, Y. Aklouf, and H. Chaalal, "A Comparative Study on Load Balancing Algorithms in Cloud Environment," in *Proceedings of the 10th International Conference on Information Systems and Technologies*, in ICIST '20. New York, NY, USA: Association for Computing Machinery, 2021. doi: 10.1145/3447568.3448466.
- [2] J. Li *et al.*, "An End-to-End Load Balancer Based on Deep Learning for Vehicular Network Traffic Control," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 953–966, 2019, doi: 10.1109/JIOT.2018.2866435.
- [3] L. A. Zadeh, "Knowledge representation in fuzzy logic," in *An introduction to fuzzy logic applications in intelligent systems*, Springer, 1992, pp. 1–25.
- [4] M. Hamdani, Y. Aklouf, and H. A. Bouarara, "A Parallel Fuzzy Load Balancing Algorithm for Distributed Nodes Over a Cloud System," *Int. J. Grid High Perform. Comput. IJGHPC*, vol. 14, no. 1, pp. 1–22, 2022.
- [5] H. Chaalal, M. Hamdani, and H. Belbachir, "Finding the best between the column store and row store Databases," in *Proceedings of the 10th International Conference on Information Systems and Technologies*, in ICIST '20. New York, NY, USA: Association for Computing Machinery, 2021. doi: 10.1145/3447568.3448548.
- [6] A. M. Alakeel and S. Arabia, "A Guide to Dynamic Load Balancing in Distributed Computer Systems," 2010. [Online]. Available: <https://api.semanticscholar.org/CorpusID:14313112>
- [7] A. A. AlKhatib, T. Sawalha, and S. AlZu'bi, "Load Balancing Techniques in Software-Defined Cloud Computing: an overview," in *2020 Seventh International Conference on Software Defined Systems (SDS)*, 2020, pp. 240–244. doi: 10.1109/SDS49854.2020.9143874.
- [8] A. Katangur, S. Akkaladevi, and S. Vivekanandhan, "Priority Weighted Round Robin Algorithm for Load Balancing in the Cloud," in *2022 IEEE 7th International Conference on Smart Cloud (SmartCloud)*, 2022, pp. 230–235. doi: 10.1109/SmartCloud55982.2022.00044.
- [9] S.-L. Chen, Y.-Y. Chen, and S.-H. Kuo, "CLB: A novel load balancing architecture and algorithm for cloud services," *Comput. Electr. Eng.*, vol. 58, pp.

- 154–160, 2017, doi: <https://doi.org/10.1016/j.compeleceng.2016.01.029>.
- [10] S. Razzaghzadeh, A. H. Navin, A. M. Rahmani, and M. Hosseinzadeh, “Probabilistic modeling to achieve load balancing in Expert Clouds,” *Ad Hoc Netw.*, vol. 59, pp. 12–23, 2017, doi: <https://doi.org/10.1016/j.adhoc.2017.01.001>.
- [11] L. Kang and X. Ting, “Application of adaptive load balancing algorithm based on minimum traffic in cloud computing architecture,” in *2015 International Conference on Logistics, Informatics and Service Sciences (LISS)*, 2015, pp. 1–5. doi: 10.1109/LISS.2015.7369792.
- [12] P. A. Pattanaik, S. Roy, and P. K. Pattnaik, “Performance study of some dynamic load balancing algorithms in cloud computing environment,” in *2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN)*, IEEE, 2015, pp. 619–624.
- [13] P. V. Lahande, “Quantum Equally Spread Current Execution Load Algorithm for Edge-Cloud Environment,” presented at the 2026 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE), IEEE, 2026, pp. 1–6.
- [14] K. Patel, A. Patel, and B. Patel, “A Comparative Analytics for Dynamic Load Balancing Mechanisms Intended to Improve Task Scheduling in Cloud Computing With Security Challenges,” *Converg. Strateg. Green Comput. Sustain. Dev.*, pp. 182–201, 2024.
- [15] M. A. Khan, “Optimized hybrid service brokering for multi-cloud architectures,” *J. Supercomput.*, vol. 76, no. 1, pp. 666–687, Jan. 2020, doi: 10.1007/s11227-019-03048-5.
- [16] Z. Wang, F. Qin, Q. Miao, and X. Yuan, “Dynamic Load Balancing of Virtual Machine Servers Using Genetic Algorithm in a Cloud Computing Environmen,” presented at the 2025 4th International Conference on Artificial Intelligence, Internet of Things and Cloud Computing Technology (AIoTC), IEEE, 2025, pp. 776–780.
- [17] S.-J. Yang and Y.-R. Chen, “Design adaptive task allocation scheduler to improve MapReduce performance in heterogeneous clouds,” *J. Netw. Comput. Appl.*, vol. 57, pp. 61–70, 2015, doi: <https://doi.org/10.1016/j.jnca.2015.07.012>.
- [18] M. Kalra and S. Singh, “A review of metaheuristic scheduling techniques in cloud computing,” *Egypt. Inform. J.*, vol. 16, no. 3, pp. 275–295, 2015, doi: <https://doi.org/10.1016/j.eij.2015.07.001>.
- [19] I. A. T. Hashem *et al.*, “MapReduce scheduling algorithms: a review,” *J. Supercomput.*, vol. 76, no. 7, pp. 4915–4945, Jul. 2020, doi: 10.1007/s11227-018-2719-5.
- [20] F. Ebadifard and S. M. Babamir, “Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment,” *Clust. Comput.*, vol. 24, no. 2, pp. 1075–1101, Jun. 2021, doi: 10.1007/s10586-020-03177-0.
- [21] K. Madanmohan and T. Hirwarkar, “Analysis Of Alba Efficiency For Distributed Cloud Computing,” *Turk. J. Comput. Math. Educ.*, vol. 12, no. 2, pp. 1405–1410, 2021.
- [22] P. Wu, L. Zhou, H. Chen, and H. Zhou, “An improved fuzzy risk analysis by using a new similarity measure with center of gravity and area of trapezoidal fuzzy numbers,” *Soft Comput.*, vol. 24, no. 6, pp. 3923–3936, Mar. 2020, doi: 10.1007/s00500-019-04160-7.
- [23] P. Ram and R. Sibal, “A Novel Hybrid Approach to Enhance Software Quality by Using the Honey Badger Algorithm (HBA) and Support Vector Machine (SVM),” in *2025 International Conference on Electronics, Computing, Communication and Control Technology (ICECCC)*, 2025, pp. 1–6. doi: 10.1109/ICECCC65144.2025.11064181.