The Scientific Temper (2024) Vol. 15 (spl): 71-77

E-ISSN: 2231-6396, ISSN: 0976-8653

Doi: 10.58414/SCIENTIFICTEMPER.2024.15.spl.09

https://scientifictemper.com/

RESEARCH ARTICLE

Fuzzy logic-driven scheduling for cloud computing operations: A dynamic and adaptive approach

A. Kalaiselvi*, A. Chandrabose

Abstract

Cloud computing is a decentralized approach of providing and accessing computer services through the internet. The phrase «cloud computing» is commonly used to describe this setup. The term «cloud computing» refers to a method of running computer programs, data, and services over the internet from a central location rather than on individual users' local machines. Cloud computing environments face the challenge of efficiently managing and scheduling diverse tasks to ensure optimal resource utilization and system performance. This paper introduces a fuzzy logic-based approach for scheduling cloud computing operations designed to handle the uncertainty and dynamic nature of task execution requirements. The proposed method incorporates fuzzy rules and membership functions to evaluate key parameters such as task priority, resource availability, and execution time. By modeling these uncertainties, the fuzzy logic system dynamically adjusts scheduling decisions to optimize load balancing and minimize delays. The approach offers flexibility in allocating resources and prioritizing tasks in real-time, adapting to fluctuating workloads and system conditions. Experimental simulations demonstrate the effectiveness of the fuzzy logic approach in enhancing system throughput and reducing task completion time, offering a robust solution for scheduling in heterogeneous and complex cloud environments. This method shows promise for improving the scalability and responsiveness of cloud-based operations. Comparisons with three separate scheduling algorithms the first come, first serve (FCFS) algorithm, the round robin (RR) strategy, and the Honeybee foraging (HF) algorithm, show that our method is quite effective. The experimental findings validate the efficacy of our algorithm.

Keywords: Cloud computing, Fuzzy logic, Task scheduling, Adaptive scheduling.

Introduction

Cloud computing is a model that enables on-demand access to a shared pool of configurable computing resources (such as servers, storage, networks, applications, and services) over the internet. These resources can be rapidly provisioned and released with minimal management effort, allowing organizations and individuals to store and process data, run applications, and manage systems remotely, Islam, R., Patamsetti, V., Gadhi, A., Gondu, R. M., Bandaru, C. M., Kesani, S. C., & Abiona, O. (2023), Golightly, L., Chang, V., Xu, Q. A., Gao, X., & Liu, B. S. (2022), Parast, F. K., Sindhav, C., Nikam, S., Yekta, H. I., Kent, K. B., & Hakak, S. (2022).

Key Characteristics

The following are the key characteristics of cloud computing, Pallathadka, H., Sajja, G. S., Phasinam, K., Ritonga, M., Naved, M., Bansal, R., & Quiñonez-Choquecota, J. (2022), Gang, L., & Badarch, T. (2023).

On-demand self-service

Users can automatically access and manage computing resources without requiring human interaction with the service provider.

Broad network access

Resources are available over the internet from various devices like laptops, smartphones, and tablets.

Resource pooling

Providers use multi-tenant models to serve multiple customers, dynamically allocating resources based on demand.

Edayathangudy G.S Pillay Arts and Science College (Autonomous) (Affiliated to Bharathidasan University, Tiruchirappalli) Nagapattinam, Tamilnadu, India.

***Corresponding Author:** A. Kalaiselvi, Edayathangudy G.S Pillay Arts and Science College (Autonomous) (Affiliated to Bharathidasan University, Tiruchirappalli) Nagapattinam, Tamilnadu, India., E-Mail: lakshithsiva1386@gmail.com

How to cite this article: Kalaiselvi, A., Chandrabose, A. (2024). Fuzzy logic-driven scheduling for cloud computing operations: A dynamic and adaptive approach. The Scientific Temper, **15**(spl):71-77.

Doi: 10.58414/SCIENTIFICTEMPER.2024.15.spl.09

Source of support: Nil

Conflict of interest: None.

Received: 05/09/2024 **Accepted:** 17/09/2024 **Published**: 16/10/2024

Rapid elasticity

Resources can be quickly scaled up or down based on usage needs.

Measured service

Cloud systems automatically control and optimize resource use by leveraging metering capabilities, offering a pay-as-you-go model.

Service Models

The following are the cloud service models: Marinescu, D. C. (2022), Maaz, M., Ahmed, M. A., Maqsood, M., & Soma, S. (2023), Volkov, A. O., Korobkina, A. V., & Stepanov, S. N. (2022, March).

Infrastructure as a Service (IaaS)

Provides virtualized computing resources like virtual machines, storage, and networking.

Platform as a Service (PaaS)

Offers platforms that allow developers to build, deploy, and manage applications without dealing with the underlying infrastructure.

Software as a Service (SaaS)

Delivers fully functional applications over the internet that users can access through a browser without managing any infrastructure.

Deployment Models

Public Cloud

Resources are owned and operated by a third-party provider and made available to the general public.

Private Cloud

Resources are used exclusively by a single organization, offering greater control and security.

Hybrid Cloud

Combines public and private cloud models, allowing data and applications to be shared between them for greater flexibility.

Background Study On Task Scheduling

Task scheduling is a critical process in cloud computing, as it determines how computing tasks are assigned to resources like virtual machines (VMs), ensuring efficient resource utilization and maintaining performance goals such as minimizing execution time, energy consumption, and operational cost, Abdel-Basset, M., Mohamed, R., Abd Elkhalik, W., Sharawi, M., & Sallam, K. M. (2022), Liu, H. (2022).

In cloud computing environments, task scheduling involves distributing and managing tasks across different servers or data centers, balancing the load while meeting the diverse requirements of users and applications.

Importance of Task Scheduling

Task scheduling in cloud computing significantly impacts, Khan, M. S. A., & Santhosh, R. (2022), Murad, S. A., Muzahid, A. J. M., Azmi, Z. R. M., Hoque, M. I., & Kowsher, M. (2022):

Performance

Effective task scheduling ensures timely execution of tasks, reducing delays and improving system throughput.

Resource Utilization

By properly allocating tasks to virtual machines, resources can be used more efficiently, avoiding situations where some machines are idle while others are overburdened.

Energy Efficiency

Scheduling algorithms can minimize energy consumption by reducing the number of active servers and optimizing resource usage.

Cost Optimization

Proper scheduling lowers operational costs by optimizing resource usage, enabling pay-per-use cost models common in cloud environments.

Types of Task Scheduling in Cloud Computing

Task scheduling algorithms can be broadly categorized into static and dynamic scheduling, Alakbarov, R. (2022); Sharma, M., Kumar, M., & Samriya, J. K. (2022):

Static Scheduling

In this approach, tasks are pre-assigned to resources before execution based on known parameters such as task execution time, resource availability, and dependency. Once assigned, tasks cannot be rescheduled. This method is suitable for environments where task characteristics are predictable.

Dynamic Scheduling

In dynamic scheduling, tasks are assigned to resources during runtime based on current system conditions such as available CPU, memory, or bandwidth. This method is more flexible and adaptive, making it suitable for dynamic cloud environments where tasks and resources vary.

Common Task Scheduling Algorithms

There are various scheduling algorithms used in cloud computing, each focusing on different performance metrics such as time, cost, energy, or a combination of these, Aktan, M. N., & Bulut, H. (2022); Hamid, L., Jadoon, A., & Asghar, H. (2022):

First Come, First Served (FCFS)

Tasks are executed in the order they arrive, without considering resource availability or task priorities.

Round Robin (RR)

Tasks are assigned to resources in a cyclic manner, distributing the workload evenly but without considering task length or priority.

Min-Min and Max-Min

These are heuristic approaches that schedule tasks based on their execution times. Min-Min assigns the shortest tasks

to the fastest available resources, while Max-Min assigns the longest tasks to the fastest resources.

Genetic Algorithms (GA)

These are evolutionary algorithms that search for an optimal or near-optimal scheduling solution by mimicking the process of natural selection.

Ant Colony Optimization (ACO)

Inspired by the behavior of ants, this algorithm uses cooperative agents to find optimal paths for task scheduling.

Particle Swarm Optimization (PSO)

PSO uses a population-based approach to explore the scheduling space and find an optimal or near-optimal solution.

Priority-based Scheduling

This method assigns tasks based on their priority levels, ensuring that high-priority tasks are executed before lower-priority ones.

Hybrid Scheduling Algorithms

These algorithms combine multiple approaches (e.g., GA + PSO) to achieve better performance in multi-objective optimization.

Background Study On Fuzzy Logic

Fuzzy logic is a form of logic that allows reasoning with imprecise or uncertain information. It extends classical (binary) logic by introducing degrees of truth, enabling the modeling of human reasoning more effectively. While classical logic operates with discrete values (true/false or 1/0), fuzzy logic works with continuous values ranging between 0 and 1, which can represent the degree to which a statement is true or false, Zadeh, L. A. (2023), Van Krieken, E., Acar, E., & van Harmelen, F. (2022).

Fuzzy logic was introduced by Lotfi Zadeh in 1965 as part of his research on the mathematical representation of human reasoning. Zadeh proposed that many real-world situations are not strictly binary and that traditional logic struggles to handle this complexity. For example, terms like "warm," "tall," or "expensive" are inherently vague, and fuzzy logic provides a framework to deal with this vagueness by representing these concepts with fuzzy sets and degrees of membership.

Fuzzy Sets and Membership Functions

At the core of fuzzy logic is the concept of fuzzy sets. In classical set theory, an element either belongs to a set or it does not. However, in fuzzy set theory, an element can have a degree of membership in a set, represented by a value between 0 and 1, Varshney, A., & Goyal, V. (2023).

Fuzzy Sets

A fuzzy set is a collection of elements where each element has a degree of membership. For example, a fuzzy set could represent the concept of "tall people," and each person's height would have a corresponding membership value between 0 and 1, indicating how "tall" they are relative to others.

Membership Functions

These functions define how the degree of membership is determined for any given input. Common types of membership functions include:

- *Triangular*

Defined by a triangle shape with a peak value and linear transitions.

- *Trapezoidal*

Similar to the triangular function but with a flat top, representing a range of values with full membership.

- *Gaussian*

Bell-shaped, representing smooth transitions.

Fuzzy Logic Systems (FLS)

A fuzzy logic system (FLS) is a decision-making system based on fuzzy logic, consisting of four key components, Zhao, T., Cao, H., & Dian, S. (2022):

Fuzzification

Converts crisp inputs into fuzzy sets by determining the degree of membership for each input variable using membership functions.

Inference Engine

Applies a set of fuzzy rules to the fuzzified inputs to generate fuzzy output sets. These rules are typically in the form of IF-THEN statements (e.g., «If temperature is high, then fan speed should be high»).

Rule Base

A collection of fuzzy rules that represent the knowledge of the system. These rules define how to combine fuzzy inputs to determine fuzzy outputs.

Defuzzification

Converts the fuzzy outputs back into crisp values, which can be used as actionable decisions or control inputs. Common defuzzification techniques include the Centroid method and Mean of Maximum.

Proposed Fuzzy Logic Driven Scheduling Approach

In cloud computing environments, efficient task scheduling is crucial to improve resource utilization, minimize execution time, and optimize energy consumption. A fuzzy logic-based task scheduling (FLTS) approach leverages fuzzy logic to make intelligent and adaptive decisions regarding task allocation, especially when dealing with uncertainty in resource availability, workload, and user requirements.

The main objective of the fuzzy logic-based scheduling approach is to allocate tasks to the most suitable resources

(virtual machines) by considering multiple factors such as task priority, execution time, resource load, and energy consumption. Fuzzy logic is employed to handle the inherent uncertainties in these factors and derive an optimal decision for task assignment.

Step by step procedure for FLTS

Step 1: Define Input Variables

To implement fuzzy logic-based task scheduling, identify the key parameters that influence task scheduling. These input parameters are fuzzified to reflect the imprecise nature of resource and task characteristics.

- *Task Priority (TP)*

Indicates the importance of the task. Higher priority tasks should be allocated to resources first.

Fuzzy terms: {Low, Medium, High}

- *Estimated Execution Time (EET)*

The expected time a task will take to execute on a resource.

Fuzzy terms: {Short, Moderate, Long}

Resource Load (RL)

The current load or utilization of the resource (VM).

Fuzzy terms: {Low, Medium, High}

- *Energy Consumption (EC)*

The energy cost of executing the task on a specific resource.

Fuzzy terms: {Low, Moderate, High}\

Step 2: Define Membership Functions

For each input variable, define membership functions to represent the fuzziness in the data. The membership functions translate crisp input values into fuzzy degrees of membership between 0 and 1. A typical approach uses triangular or trapezoidal membership functions for each fuzzy set. Example of a triangular membership function for Task Priority:

$$\mu_{Low}(TP) = \begin{cases} 1, TP \leq 1 \\ \frac{3 - TP}{2}, 1 < TP \leq 3 \\ 0, TP > 3 \end{cases}$$

Step 3: Define Fuzzy Rule Base

Develop a rule base that captures the scheduling decisions based on the input parameters. These IF-THEN rules help to decide the best allocation of tasks to resources. Examples of fuzzy rules:

- *Rule 1*

IF Task Priority is High AND Estimated Execution Time is Short AND Resource Load is Low THEN Task Allocation is Immediate.

- *Rule 2*

IF Task Priority is Medium AND Resource Load is High, THEN Task Allocation is Delayed.

- *Rule 3*

IF Task Priority is Low AND Energy Consumption is High THEN Task Allocation is Rejected.

Step 4: Fuzzification

During runtime, the crisp input values (e.g., the actual task priority, estimated execution time, resource load, and energy consumption) are converted into fuzzy values using the predefined membership functions.

For example, suppose a task has a priority of 4, an estimated execution time of 10 seconds, and is assigned to a resource with a current load of 60%. In that case, the corresponding fuzzy values for these inputs are calculated using the membership functions.

Step 5: Inference Engine

The inference engine processes the fuzzified inputs using the Mamdani inference model. It evaluates the fuzzy rules from the rule base and produces fuzzy outputs (e.g., Task Allocation Decisions). The inference process involves:

- *Rule Evaluation*

Checking which fuzzy rules are triggered based on the fuzzified inputs.

- *Aggregation of Outputs*

Combining the results of all triggered rules to form a single fuzzy output.

The defuzzified output is the task allocation decision, which can have values like:

Immediate allocation (high priority, low resource load, short execution time).

Delayed allocation (low priority or high resource load).

Rejection (high energy consumption or low-priority task when resources are scarce).

The centroid defuzzification method is given by:

$$y = \frac{\int \mu(y) y dy}{\int \mu(y) dy}$$

Where μ(y) is the aggregated fuzzy output, and y is the task allocation decision.

Step 7: Task Assignment

Based on the defuzzified task allocation decision, the task is either:

- Assigned to a virtual machine (VM) for immediate execution.
- Delayed until more resources are available.
- Rejected if resources are not sufficient or the task is not critical.

Result And Discussion

The performance of the proposed fuzzy logic-based task scheduling (FLTS), first come, first serve (FCFS), round robin (RR), and honeybee foraging (HF) approaches are compared based on metrics like makespan, average waiting time (AWT), resource utilization (RU), throughput for the varying number of tasks.

Table 1: Makespan (in seconds) by the proposed FLST, FCFS, RR and HF with varying number of tasks

Number of Tasks	*Makespan (in seconds*			
	FLTS	*FCFS*	*RR*	*HF*
100	180	230	210	200
200	360	460	420	400
300	550	690	640	610
400	740	920	850	820
500	930	1150	1060	1030

Table 2: Average Waiting Time (AWT) (in seconds) by the Proposed FLTS, FCFS, RR and HF with varying number of tasks

Number of Tasks	*Average Waiting Time (AWT) (in seconds)*			
	FLTS	*FCFS*	*RR*	*HF*
100	20	60	45	30
200	35	120	95	60
300	50	180	140	90
400	70	240	185	120
500	90	300	230	150

Table 3: Throughput (in task/seconds) by the proposed FLTS, FCFS, RR and HF with varying number of tasks

Number of Tasks	*Throughput (in task/seconds)*			
	FLTS	*FCFS*	*RR*	*HF*
100	50	40	45	42
200	52	38	44	46
300	55	36	43	48
400	58	34	41	49
500	60	32	39	50

Table 4: Resource utilization (in %) by the proposed FLTS, FCFS, RR and HF with varying number of tasks

Number of tasks	*Resource utilization (in %)*			
	FLTS	*FCFS*	*RR*	*HF*
100	85	60	70	75
200	88	63	73	77
300	90	65	75	79
400	92	68	78	81
500	94	70	80	83

Table 1 depicts the makespan (in seconds) by the Proposed FLST, FCFS, RR and HF with varying number of tasks.

The FLTS approach consistently achieves the lowest makespan for all numbers of tasks. This reduction is due to the intelligent decision-making of the fuzzy logic system, which considers multiple factors like task priority, resource load, and estimated execution time to assign tasks more efficiently. As the number of tasks increases, FLTS adapts well to the rising load, keeping the makespan increase at a lower rate than the other methods. The makespan increases significantly as the number of tasks rises, indicating that FCFS struggles to handle larger task loads effectively. The RR strategy shows better performance than FCFS but lags behind both FLTS and HF. The HF approach performs better than FCFS and RR, as it balances tasks dynamically across resources based on task demand and resource availability.

Table 2 depicts the average waiting time (AWT) (in seconds) by the proposed FLST, FCFS, RR and HF with varying number of tasks.

From Table 2, The proposed FLTS approach consistently achieves the lowest average waiting time (AWT) across all numbers of tasks. FCFS shows the highest AWT among all the approaches. The RR strategy performs better than FCFS but still has a higher AWT compared to both FLTS and HF. HF shows better performance than RR and FCFS, as it balances tasks dynamically and uses an adaptive scheduling approach.

Table 3 depicts the throughput (in task/seconds) by the Proposed FLST, FCFS, RR and HF with varying number of tasks.

From Table 3, The FLTS approach achieves the highest throughput across all task sizes. By dynamically allocating tasks based on resource availability and task priority, FLTS optimizes the completion of tasks, leading to higher throughput. As the number of tasks increases, the throughput shows a consistent increase, demonstrating the scalability and efficiency of this method. The FCFS approach has the lowest throughput across all task numbers. Tasks are processed in the order they arrive without consideration for resource optimization, resulting in lower efficiency, particularly with longer tasks at the front of the queue. The throughput decreases steadily as the number of tasks increases, indicating a struggle to handle larger workloads effectively. The RR strategy shows a moderate throughput, better than FCFS but lower than FLTS and HF. While RR attempts to balance task processing by allocating equal time slices, it may not efficiently utilize resources, particularly if tasks have varying execution times. Throughput remains stable but lower than FLTS, with a gradual decrease as the number of tasks increases. The HF approach performs better than FCFS and RR but does not reach the efficiency of FLTS. By mimicking natural foraging behavior, HF can adaptively allocate tasks, improving throughput compared to FCFS and RR. The throughput for HF increases as task numbers rise, reflecting its adaptability to workload demands.

Table 4 depicts the resource utilization (RU) (in %) by the proposed FLST, FCFS, RR and HF with varying numbers of tasks.

From Table 4, The Proposed FLTS approach shows the highest resource utilization across all task numbers. FLTS

dynamically allocates resources based on task priority and resource availability using fuzzy logic, leading to optimized usage of computing resources. As the number of tasks increases, the resource utilization improves significantly, showing that the approach efficiently scales with workload increases. FCFS has the lowest resource utilization among all approaches. Since FCFS does not prioritize tasks or consider the current state of resources, it often leads to resource underutilization, especially when handling a mix of tasks with varying resource requirements. Resource utilization increases gradually with the task load but remains lower than the other methods. The RR approach shows moderate resource utilization but is still outperformed by both FLTS and HF. RR assigns time slices equally to tasks, which can lead to suboptimal use of resources when some tasks could be completed faster or require fewer resources. Resource utilization increases steadily as the task count grows, but it lacks the adaptability needed for maximum efficiency. The HF algorithm performs better than RR and FCFS in terms of resource utilization but is still behind the proposed FLTS approach. By mimicking the foraging behavior of bees to balance workload distribution, HF achieves better resource utilization compared to RR and FCFS. However, it cannot match the fine-grained resource allocation and task prioritization provided by the FLTS approach, resulting in slightly lower resource utilization.

Conclusion

The results of evaluating the proposed fuzzy logic-based task scheduling (FLTS), first come first serve (FCFS), round Robin (RR), and honeybee foraging (HF) approaches in a cloud computing environment demonstrate the superior performance of the FLTS method across multiple metrics, including throughput, average waiting time (AWT), and resource utilization (RU).

Throughput

The FLTS approach consistently achieved the highest throughput across all task sizes, efficiently completing more tasks per unit time compared to FCFS, RR, and HF. This improvement in throughput is due to FLTS's ability to dynamically allocate resources and prioritize tasks based on system conditions and task urgency.

Average Waiting Time (AWT)

FLTS significantly reduced the waiting time for tasks in the queue, outperforming all other approaches. This reduction is attributed to its intelligent scheduling, which balances the load while minimizing delays.

Resource Utilization (RU)

FLTS maximized resource utilization, effectively using the available computing resources more efficiently than the other methods. The dynamic nature of fuzzy logic ensured optimal allocation of CPU, memory, and other resources, reducing wastage and improving system performance.

In comparison, FCFS showed the poorest performance across all metrics, mainly due to its rigid scheduling mechanism, which does not optimize task processing or resource allocation. Round Robin (RR) demonstrated moderate performance but failed to adapt dynamically to varying task demands, leading to lower throughput and resource utilization than FLTS. Honeybee foraging (HF) performed better than RR and FCFS due to its adaptive nature, though it still lagged behind FLTS in overall performance.

In conclusion, the proposed FLTS approach offers a highly efficient solution for task scheduling in cloud computing environments. It provides superior performance in terms of throughput, task responsiveness, and resource efficiency, making it an ideal choice for managing diverse and dynamic workloads in modern cloud infrastructures.

References

Abdel-Basset, M., Mohamed, R., Abd Elkhalik, W., Sharawi, M., & Sallam, K. M. (2022). Task scheduling approach in cloud computing environment using hybrid differential evolution. *Mathematics*, 10(21), 4049.

Aktan, M. N., & Bulut, H. (2022). Metaheuristic task scheduling algorithms for cloud computing environments. *Concurrency and Computation: Practice and Experience*, 34(9), e6513.

Alakbarov, R. (2022). An optimization model for task scheduling in mobile cloud computing. *International Journal of Cloud Applications and Computing (IJCAC)*, 12(1), 1-17.

Gang, L., & Badarch, T. (2023). Research on Characteristics and Technologies of Cloud Computing. *American Journal of Computer Science and Technology*, 6(1), 33-41.

Golightly, L., Chang, V., Xu, Q. A., Gao, X., & Liu, B. S. (2022). Adoption of cloud computing as innovation in the organization. *International Journal of Engineering Business Management*, 14, 18479790221093992.

Hamid, L., Jadoon, A., & Asghar, H. (2022). Comparative analysis of task level heuristic scheduling algorithms in cloud computing. *The Journal of Supercomputing*, 78(11), 12931-12949.

Islam, R., Patamsetti, V., Gadhi, A., Gondu, R. M., Bandaru, C. M., Kesani, S. C., & Abiona, O. (2023). The future of cloud computing: benefits and challenges. *International Journal of Communications, Network and System Sciences*, 16(4), 53-65.

Khan, M. S. A., & Santhosh, R. (2022). Task scheduling in cloud computing using hybrid optimization algorithm. *Soft computing*, 26(23), 13069-13079.

Liu, H. (2022). Research on cloud computing adaptive task scheduling based on ant colony algorithm. *Optik*, 258, 168677.

Maaz, M., Ahmed, M. A., Maqsood, M., & Soma, S. (2023). Development Of Service Deployment Models In Private Cloud. *Journal of Scientific Research and Technology*, 1-12.

Marinescu, D. C. (2022). Cloud computing: theory and practice. Morgan Kaufmann.

Murad, S. A., Muzahid, A. J. M., Azmi, Z. R. M., Hoque, M. I., &

Kowsher, M. (2022). A review on job scheduling technique in cloud computing and priority rule based intelligent framework. *Journal of King Saud University-Computer and Information Sciences*, 34(6), 2309-2331.

Pallathadka, H., Sajja, G. S., Phasinam, K., Ritonga, M., Naved, M., Bansal, R., & Quiñonez-Choquecota, J. (2022). An investigation of various applications and related challenges in cloud computing. *Materials Today: Proceedings*, 51, 2245-2248.

Parast, F. K., Sindhav, C., Nikam, S., Yekta, H. I., Kent, K. B., & Hakak, S. (2022). Cloud computing security: A survey of service-based models. Computers & Security, 114, 102580.

Sharma, M., Kumar, M., & Samriya, J. K. (2022). An optimistic approach for task scheduling in cloud computing. *International Journal of Information Technology*, 14(6), 2951-2961.

Van Krieken, E., Acar, E., & van Harmelen, F. (2022). Analyzing differentiable fuzzy logic operators. *Artificial Intelligence*, 302, 103602.

Varshney, A., & Goyal, V. (2023). Re-evaluation on fuzzy logic controlled system by optimizing the membership functions. *Materials Today: Proceedings*.

Volkov, A. O., Korobkina, A. V., & Stepanov, S. N. (2022, March). Development of a Model and Algorithms for Servicing Real-Time and Data Traffic in a Cloud Computing System. *In 2022 Systems of Signals Generating and Processing in the Field of on Board Communications* (pp. 1-5). IEEE.

Zadeh, L. A. (2023). Fuzzy logic. In Granular, Fuzzy, and Soft Computing (pp. 19-49). New York, NY: *Springer* US.

Zhao, T., Cao, H., & Dian, S. (2022). A self-organized method for a hierarchical fuzzy logic system based on a fuzzy autoencoder. *IEEE Transactions on Fuzzy Systems*, 30(12), 5104-5115.