INTELLIGENT QOS-AWARE FLOOD PREDICTION USING A NOVEL HYBRID OPTIMIZATION METHOD ON CLOUD PLATFORMS

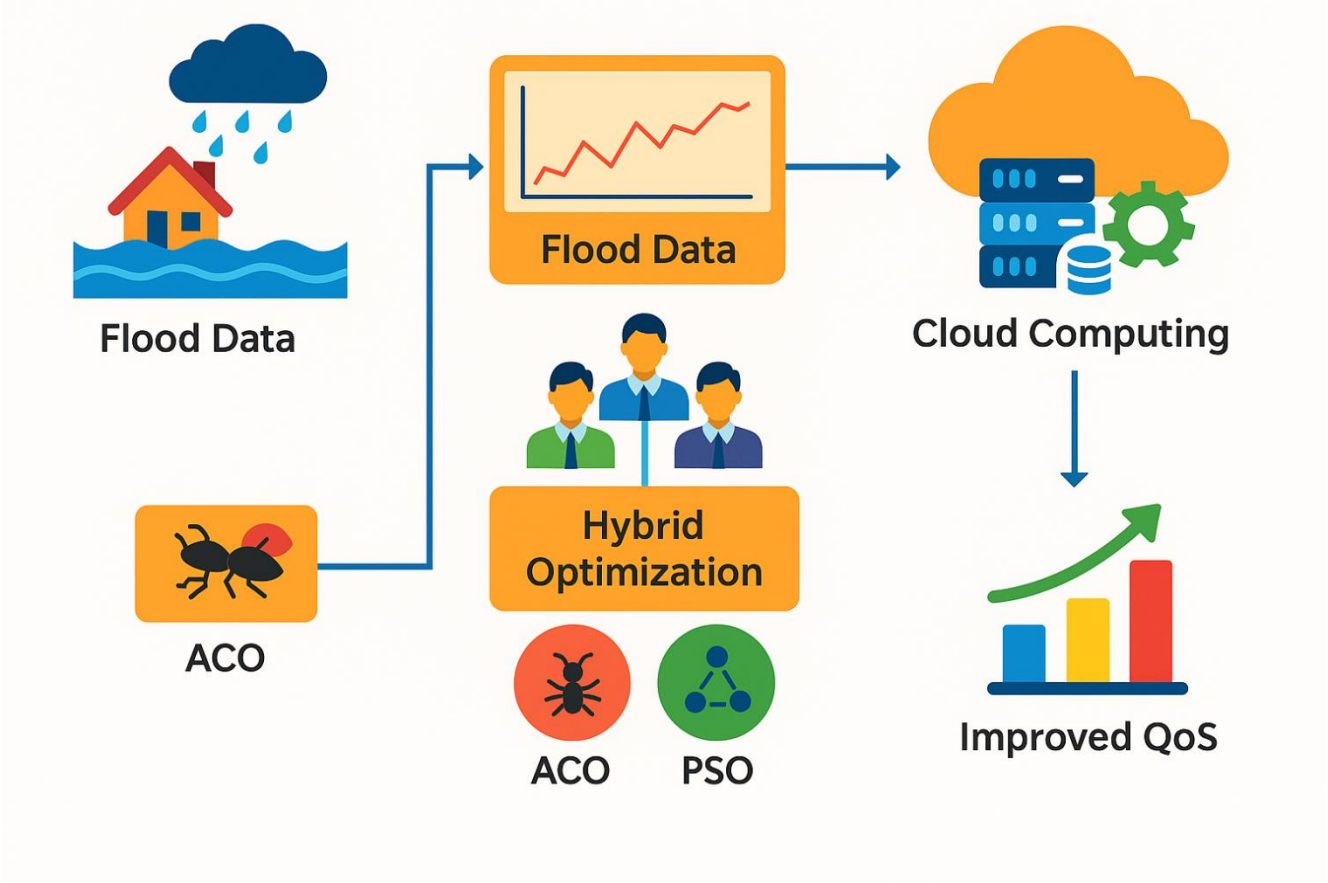
Sindhuja. K[1*], Baghya Shree. S[2]

[*1]Department of Computer Science and Engineering, RMK Engineering College, Chennai, Tamil Nadu, India.

[2]Department of Electrical and Electronics Engineering, Anna University Regional Campus, Madurai, Tamil Nadu, India.

[*]Corresponding Author: sindhuja.msec@gmail.com

Graphical Abstract



Abstract

Floods caused by intense and unpredictable rainfall remain one of the most frequent and destructive natural disasters, particularly affecting the eastern regions of India. Accurate and real-time flood prediction is critical in minimizing the loss of life and property. While cloud computing offers a scalable and flexible infrastructure for environmental data processing, challenges such as service heterogeneity, latency issues, and Quality of Service (QoS) degradation often hinder its effectiveness. This research proposes a novel multi-agent-based hybrid optimization framework that leverages Ant Colony

Optimization (ACO) and Particle Swarm Optimization (PSO) to address QoS limitations in cloud-based flood prediction systems. The innovation lies in integrating intelligent agents within a cloud platform, allowing dynamic interaction and coordination to handle user requests and data-driven flood prediction tasks optimally. An ablation study using real-world flood datasets validates the model's performance. The proposed hybrid ACO-PSO algorithm significantly improves prediction accuracy, reduces system latency, and reduces computational cost. It establishes its potential as an efficient solution for real-time, cloud-enabled flood monitoring and early warning systems. The proposed model gives an accuracy of 94.7%, a latency of 1.1s, a precision of 94%, a recall of 95% and an F1-score of 94.5% respectively, compared to standard PSO and ACO.

Keywords: Cloud computing; computational cost; flood prediction; Multi agent-based hybrid algorithm; prediction accuracy; QoS; service delivery; system latency

1. Introduction

Cloud computing has become a powerful tool among researchers, offering cloud services for processing real-time technical applications. Within this context, ecological and environmental sciences receive considerable attention due to their dependence on highly accurate, real-time predictions. Among various natural disasters, floods are recognized as one of the most devastating events globally, primarily driven by rapid urbanization and intense rainfall. The consequences of flooding are often severe and irreversible, including loss of human life, destruction of public and private property, transportation disruptions, service interruptions, and contamination of water bodies. Therefore, it is imperative to develop reliable methods for early flood prediction. Integrating machine learning techniques with cloud computing platforms is increasingly addressing this challenge. In the past, most districts in Odisha have experienced recurrent flooding, resulting in significant loss of life and extensive property damage. The frequency and severity of such events are expected to escalate in the coming years due to the impacts of climatic variation. Numerous studies have evaluated Odisha's vulnerability to floods, consistently highlighting that many districts are frequently affected by floods of varying intensities. Given this alarming trend, there is an urgent need to minimize these losses through timely and accurate flood prediction. A hybrid approach has been proposed to address this challenge, offering improved precision and timeliness compared to earlier methods. The proposed technique approach leverages the combined strengths of feature selection and data classification techniques. The study presents a robust and automated methodology for early flood prediction, utilizing current environmental indicators and data-driven analysis. The proposed hybrid model significantly improves prediction performance by integrating various data mining classification algorithms with optimization-based feature selection techniques. Furthermore, the cloud computing platform ensures seamless handling of large and continually

expanding datasets. Such advanced predictive models can be crucial in mitigating the devastating effects of floods, including loss of life and damage to public infrastructure and private property.

Information technology, cloud computing, and software agents are separate but interdependent concepts. Agents play an essential role in cloud computing, contributing to automation, optimization, security, and overall efficiency by improving various aspects of the cloud environment. These agents respond to events (such as changes in consumer requirements) and have the ability to self-organize through means such as collaboration [10, 11], cooperation [12], coordination [13, 14], negotiation, and interaction. There may be insufficient knowledge regarding cloud service providers and inadequate capabilities (for instance, mapping consumers' needs to access cloud resources) [15]. This can be challenging during the service composition and delivery phases, affecting the QoS. Agent-based cloud computing techniques have been proposed to overcome these problems. Furthermore, agent-based techniques are efficient and effective in various usability domains. Still, they are not limited to assisting humans in supporting Grid and Cloud resource management [1, 2-6], processing natural language [15], and in day-to-day activities [7, 8, 9]. Agents are practical tools for automating Cloud service administration, coping with changing requests from users, and autonomous resource mapping, according to the claims made in [3-6]. Agent-based cloud computing is the name given to this type of computing model. Cloud vendors typically work to improve the Quality of Service (QoS) for their various customers to encourage more people to embrace cloud computing. In cloud services, these aspects, including cost, security, throughput, accessibility, availability, and installation time, are frequently considered. Cloud computing, on the other hand, presents significant issues in terms of data security, balancing workloads, the allocation and scheduling of resources, and the consumption of electricity. In contrast, meta-heuristic techniques have proved more effective in solving many real-world optimization problems like scheduling [10, 11]. The integration of Spider Monkey Optimization (SMO) and Ant Colony Optimization (ACO) algorithms is used to improve the QoS parameters, but not all of them are considered [12]. The proposed research is formulating and presenting a hybrid optimization algorithm-based scheduling model using agents to minimize the QoS parameters such as prediction accuracy, system latency, and computational cost. This study presents an algorithm that optimizes the process of flood prediction. The core novelty of this study lies in the synergistic integration of Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO)—two nature-inspired metaheuristic algorithms to optimize Quality of Service (QoS) in a cloud-based flood prediction environment. Using the proposed hybrid algorithms, an ablation study is done with flood data to measure multi-agent functionality in the cloud. As a summary of this paper:

1. An extensive literature review has been conducted to gain knowledge about the difficulties faced by alternative task scheduling algorithms and strategies, and how this influences cloud efficiency.

2. Floods are becoming more frequent and more severe, leading to the need for refined prediction systems that can handle vast amounts of dynamic environmental data. Cloud computing makes Real-time data processing possible, and intelligent agents enhance task coordination and optimization effectiveness. Using a hybrid PSO-ASO algorithm, this paper presents a Multi-Agent System (MAS) architecture in which agents cooperate to maximize system performance and model accuracy. This algorithm improves the task scheduling performance based on flood data by addressing the drawbacks and issues encountered by various algorithms. The implementation of this algorithm is based on the understanding gained from the different works. The functionality of the proposed hybrid model is tested with a real-time flood application to measure the multi-agent functionality over a cloud environment.
The research is organized as follows: Section 2 comprehensively evaluates diverse existing approaches. The methodology is drafted in section 3. The experimental findings are discussed in section 4, with the conclusion in section 5.

2. Related Works

Numerous research efforts have been dedicated to predicting floods, utilizing diverse machine-learning algorithms that have achieved commendable classification accuracies. The following section presents a comprehensive review of existing literature related to flood prediction using data mining techniques. Ramli et al. [7] applied a Neural Network Autoregressive Model to predict floods in Kuala Lumpur, achieving a maximum accuracy of 73.54%. The study also noted that exploring alternative modeling techniques could enhance prediction efficiency. Chau et al. [8] adopted two hybrid models: ANN based on Genetic Algorithm (ANN-GA) and Adaptive Network-based Fuzzy Inference System (ANFIS) for flood prediction in China's Yangtze River. However, the ANFIS model was limited by its reliance on numerous parameters, while the ANN-GA approach was computationally intensive. This highlights the necessity for a hybrid model that minimizes parameter requirements and reduces computational load. Mojaddadi et al. [9] integrated the frequency proportion method with an SVM to predict flooding in Malaysia's Damansara River, achieving an accuracy of 78.9%. Similarly, the author in [10] employed ANNs for flood forecasting, while the author in [12] utilized SVM for their flood prediction studies. These investigations underscore the growing significance of ML techniques in enhancing flood forecasting accuracy and efficiency. There has been an increasing requirement for cloud computing, which has led to the development of several approaches that aim to improve computing performance in the cloud. The task scheduling algorithm continues to be one of the most significant challenges the cloud computing environment must face. Most task scheduler approaches used in the past use virtual machine (VM) instances, which require considerable time to start up and use all available resources to carry out their work. The process that has been introduced makes use of the ANFIS and the BWO (Black Widow

Optimization) algorithm to assign the appropriate virtual machine (VM) to every responsibility. The scheduling of resources is an essential objective to achieve the most efficient consumption of services in the cloud environment. The BWO method is advantageous in achieving the greatest possible outcome from the ANFIS approach. The approach that has been introduced uses the VMs operating on the servers that are part of the cloud platform to finish the task that the client has requested. The optimal scheduling method also allocates a virtual machine to each user request. A novel technique developed to enhance task assignment was MOWOS (Multi-Objective Workflow Optimization Strategy). It reduces the time required to complete the task and the cost of carrying it out. The method that has been presented is constructed in such a manner that it uses the task-segmentation approach to break down complex jobs into smaller, manageable subtasks, which ultimately results in a reduction in the amount of time that is required for scheduling. The results of the simulation that were obtained indicate that the method that was introduced can be utilized effectively in the process of allocating virtual machines and their deployment. It can also manage streaming jobs that arrive at a random pace. All jobs could fulfill their deadlines thanks to the proposed algorithm, which cut the time it took to do things by ten percent, cut costs by eight percent, and increased resource utilization by fifty-three percent [26].

3. Methodology

The proposed model's methodology is illustrated in Fig. 1. The framework is structured into two primary modules. The initial module focuses on predicting flood occurrences by applying various learning and optimization algorithms on a stand-alone machine. The second module involves deploying these trained models within a cloud environment, significantly reducing execution time and CPU usage while enhancing overall prediction accuracy.

3.1. Dataset

Water depth estimation was performed on a dataset of 1,177 images depicting flooded roadways featuring vehicles or pedestrians. The image resolutions varied significantly, ranging from 4613 × 2595 pixels to as low as 142 × 107 pixels (https://www.hydroshare.org/resource/24866122a6ee456c8f7c80aa87a9abcb/). These images were sourced using search queries such as "urban waterlogging" and "urban floods" on Google, with additional contributions from traffic surveillance cameras. The image selection process deliberately included challenging and complex scenarios, such as low-light conditions at night, murky water, and splashes caused by moving vehicles, to ensure the robustness of the analysis. There are three main parts to the dataset: 1. Water level (m), river discharge (m^3/s), and rainfall intensity (mm/hr) are among the

real-time features extracted from IoT sensors; 2. Historical records covering hourly and daily observations from 2010 to 2023 were acquired from NOAA and the Indian Meteorological Department (IMD) 3. Rainfall-runoff simulation models produce synthetic data to support scenarios that aren't well-represented. A flood risk label (Low, Moderate, or High) is appended to each data instance to facilitate supervised learning. To make model development and evaluation easier, the dataset is normalized and segmented into three sub15% validation and 15% testing data.

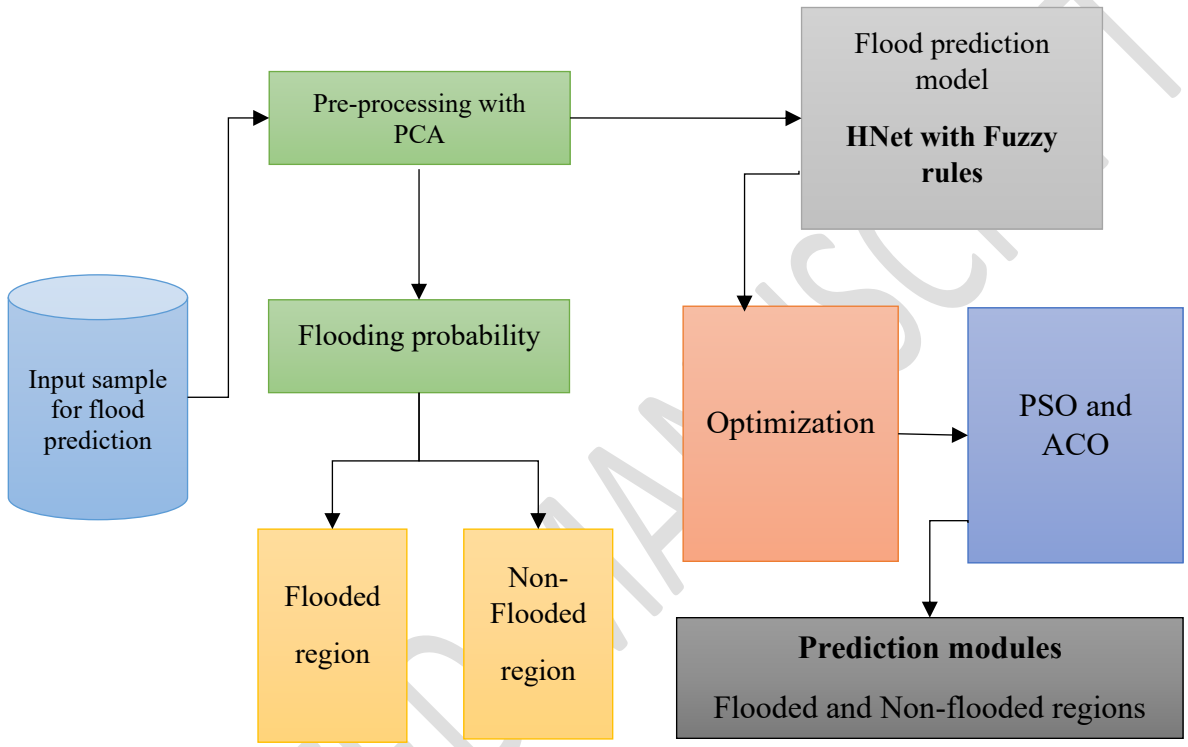


Fig 1a. Prediction pipeline

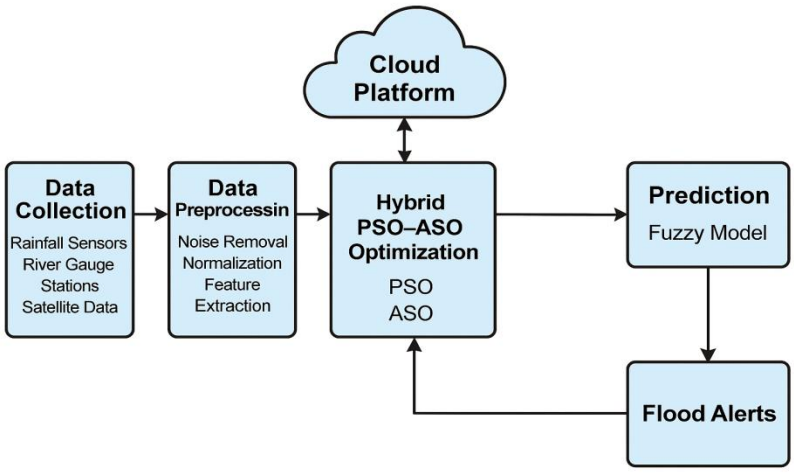


Fig 1b. Overall architecture

The proposed architecture consists of three agent layers:
Data Agents: This agent collects and processes real or historical data.
Model Agents: These agents train the fuzzy model using the optimized hybrid algorithm.
Quality of service Monitoring Agents: This evaluates accuracy, latency, and resource usage. The communication between the agents is done using a shared message queue on cloud platforms (e.g., AWS, Azure).

3.2. Flood Prediction with Hierarchical Network Layer (HNet) with Fuzzy Rules

The suggested approach uses the flood vulnerability analysis component based on HNet to evaluate environmental features and accurately calculate the flood susceptibility of any given geographic location. A neural network and a fuzzy inference system were combined to create HNet, which combines the exceptional knowledge interpretation and inference capabilities of fuzzy logic with the learning capabilities of neural networks (Sim et al. 2011). Figure 2 shows the 5-layer HNet design for the suggested method. Below is a discussion of each layer's significance.

Layer 1: In the layer's adaptive nodes, the membership values are assessed using the Gaussian MF for flood-related input parameters ($FCP_1;\ FCP_2;\ldots\ldots FCP_n$).

$$GMF = \mu_{P_i}(FCP_n) = e^{-\left(\frac{FCP_n-\beta_i}{2\alpha_i}\right)^2};\ \ i = 1,2,\ldots,k \tag{1}$$

Where k represents the total amount of inputs, μ_{P_i} represents the fuzzy set P_i's membership degree, and α_i and β_i represent the MF coefficients or parameters. Every input parameter linked to flooding, such as humidity, temperature, season, moisture, and water level, is called FCP. FCP_1: Temperature; FCP_2: Relative Humidity; and FCP_3: Rainfall. Five fuzzy sets labeled Very Low, Low, Moderate, High, and Extreme are created from the input variables such as temperature, relative humidity, and rainfall.

$$\mu_{P_i}\left(FCP_{Temp}\right);\ \ for\ \ i = 1,2,3,4,5\ \{P_1 = VL; P_2 = L; P_3 = M, P_4 = H; P_5 = E\} \tag{2}$$

$$\mu_{Q_i}\left(FCP_{RH}\right);\ \ for\ \ i = 1,2,3,4,5\ \{Q_1 = VL; Q_2 = L; Q_3 = M, Q_4 = H; Q_5 = E\} \tag{3}$$

$$\mu_{R_i}(FCP_{RH});\ \ for\ \ i = 1,2,3,4,5\ \{R_1 = VL; R_2 = L; R_3 = M, R_4 = H; R_5 = E\} \quad (4)$$

FCP_4: Season: Five fuzzy values—winter, spring, autumn, summer, and monsoon—comprise the input variable season.

$$\mu_{S_i}(FCP_{SE});\ \ for\ \ i = 1,2,3,4,5\ \{S_1 = W; S_2 = S; S_3 = A, S_4 = Sum; S_5 = Mon\} \quad (5)$$

FCP_5: Water Level: Five fuzzy sets are created from the water level input parameter: Very Low, Low, Medium, High, and extremely high.

$$\mu_{T_i}(FCP_{WL});\ \ for\ \ i = 1,2,3,4,5\ \{T_1 = VL; T_2 = L; T_3 = M, T_4 = H; T_5 = VH\} \quad (6)$$

Layer 2: The firing power of every principle is determined by multiplying the scores calculated in the nodes of the previous layer. The circular nodes that make up this layer are stable. The result is shown as:

$$w_i = \mu_{p_i}(FCP_{temp}) * \mu_{Q_i}(FCP_{RH}) * \mu_{R_i}(FCP_{RF}) * \mu_{S_i}(FCP_{SE}) * \mu_{T_i}(FCP_{WL}) \quad (7)$$

Layer 3: Circular nodes are also present, sometimes called stable nodes. By adding up the firing strength guidelines, this layer controls them.

$$\overline{w}_i = \frac{w_i}{\sum w_i}; i = 1,2,..,k \quad (8)$$

Where, w_i is the firing strength of the i^{th} rule.

Layer 4: The outcomes of the preceding layer are multiplied by the Sugeno function. The nodes in this layer are adaptive.

$$\overline{w}_i f_i = \overline{w}_i(a_i x + b_i y + c_i); i = 1,2,..,k \quad (9)$$

Where $a_i, b_i, and\ c_i$ represent the Sugeno fuzzy inference system's variables, and w_i represents the result of the preceding layer.

Layer 5: A stable node in the current layer generates all values assessed from the preceding layer to calculate the throughput.

$$\sum \bar{w}_i f_i = \frac{\sum w_i f_i}{\sum w_i}; i = 1,2,..,k \tag{10}$$

To classify a geographic area into one of the three critical zones, safe, alert, or dangerous, the HNet generates results as an FVI. Figure 2 shows the structure of the HNet operating procedure.

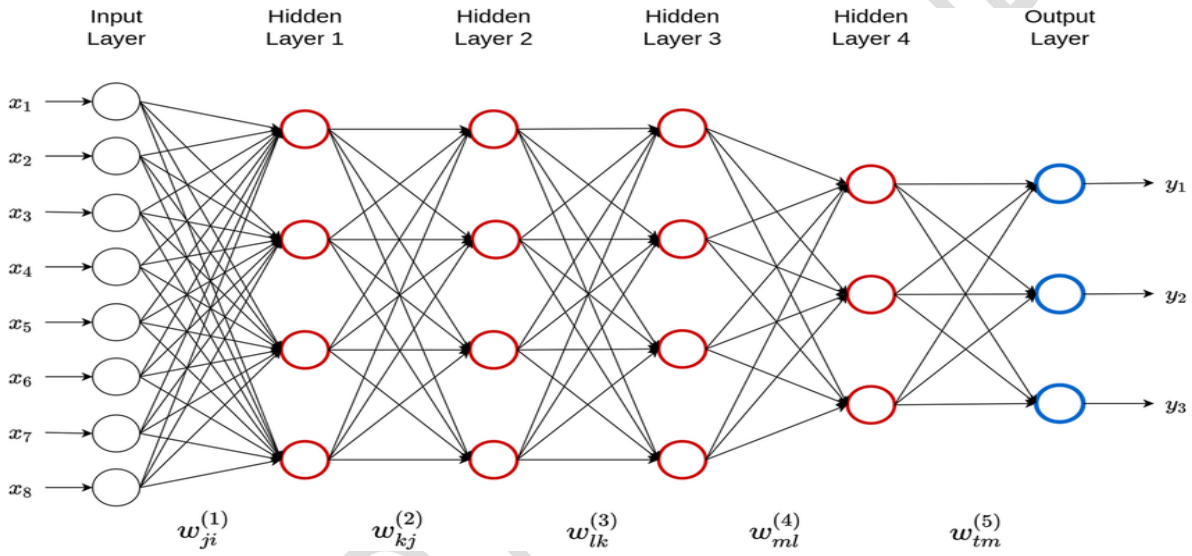


Figure 2. Hierarchical network model

Algorithm 1:

Input: dataset sample, survival time, and status

//stage_1 initialization

1. Analyze a non-overlapping image region to extract image patches from the flooded region;

2. **For all** flood region patches, **do**

3. **if** HNet ($p > 0.5$) as flooded patch

4. **else** non-flooded

---

5. **end for**

6. sampling flooded region patches from samples provided to HNet;

//stage_2 initialization

7. Train HNet;

8. model = train HNet ($\delta, patches$)

9. **For all** flood region patches, **do**

10. PCA = component analysis ($feature_{patches}$);

11. region clustering = $k-means, (no.of\ clusters, and\ PCA)$

12. **end for**

13. **For all** clusters, **do**

14. model = train HNet ($\delta, patches$)

15. **end for**

//stage_3 initialization

16. Final feature = $flood_{feature} \oplus \text{non} - \text{flood}_{feature}$;

17. Final_prediction = train cox ($final\ flood\ feature, \delta, t$)

Output: risk score

3.3.Hybrid optimization for analyzing flood features

Scheduling can be thought of as either mapping a set of tasks onto the available VMs or assigning VMs to execute the available features to satisfy the needs of the consumers. Both of these approaches are intended to fulfill the needs of the consumers. Utilizing scheduling strategies in cloud environments can accomplish several goals, the most important of which are to save energy, improve load balance and system throughput, maximize resource utilization, save expenses, and shorten the time needed for the whole execution. As a result, the scheduler needs to consider the virtualized resources and the required limitations of the consumers to encourage efficient matching between jobs and resources. One or more scheduling methods should be used to support each scheduling technique. Time, money, quality of service, amount of energy utilized, and error tolerance are the five most critical techniques. The proposed research optimizes the scheduling agent using PSO and ACO. The functionality of the hybrid optimization model is tested using the multi-agent process to measure the scheduling of tasks. The core novelty of this study lies in the synergistic integration of Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO), which are two nature-inspired meta-heuristic algorithms to optimize Quality of Service (QoS) in a cloud-based flood prediction environment. While both algorithms are powerful individually, combining their strengths helps overcome each other's weaknesses: 1) ACO excels at finding global optima through pheromone-based learning, but it may converge slowly; 2) PSO provides fast convergence using swarm intelligence, but it can get trapped in local optima. By hybridizing these algorithms, the model gains: 1) Exploration capability (from ACO) to search broader solution spaces; 2) Exploitation ability (from PSO) for faster fine-tuning of optimal solutions. This enables better optimization of QoS parameters such as: Latency, Prediction accuracy, Resource utilization, and Response time. Another key innovation is deploying the hybrid algorithm within a cloud-based multi-agent system. In this design, 1) Intelligent agents autonomously manage flood-related data and computational tasks. Agents dynamically interact with cloud resources and user requests, ensuring adaptive load balancing and efficient service provisioning. The hybrid ACO-PSO module guides the agents' decision-making in task scheduling, resource allocation, and model training. This architecture ensures: 1) Real-time responsiveness; 2) Scalable and distributed processing; and 3) Enhanced reliability and fault tolerance during peak environmental events.

3.3.1. Real-Time Flood Data Handling with Cloud Scalability

Traditional flood prediction models often fail in real-time due to limited computing resources or static infrastructure. This research addresses that gap by: 1) Using cloud computing for elastic and on-demand scalability; 2) allowing massive flood sensor data ingestion (rainfall, water levels, satellite input);

3) optimizing data handling and analysis pipelines for low-latency flood forecasting; 4) The novelty lies in making the cloud "intelligent" using the hybrid optimization engine, which: Predicts floods faster and more accurately and Adapts resource provisioning based on the urgency of the prediction request.

3.3.2. Ablation Study for Validating Optimization

Most flood prediction models lack explainability regarding why an optimization method works. This study introduces a comprehensive ablation study, systematically: 1) Testing performance of ACO alone, PSO alone, and hybrid ACO-PSO; 2) Validating improvements in prediction accuracy, latency, and computational cost; 3) Demonstrating the hybrid method's quantitative superiority across all QoS metrics and this scientific validation of hybridization impact is a novel contribution that sets your work apart from conventional optimization models.

3.3.3. Domain-Specific Application in Disaster Management

While hybrid optimization is not new in general AI, applying it to QoS-driven flood forecasting in a cloud-based multi-agent system is a novel domain-specific contribution. It addresses: 1) Real-world constraints in disaster-prone areas (e.g., bandwidth, computation bottlenecks) and 2) A use case with life-saving potential, bridging the gap between AI research and real-world environmental resilience. The process algorithm used in the proposed research and the pictorial representation are given in the flowchart (Figure 3), and the real-time adaptability is analyzed in this work.

1. Creating a cloud environment using CloudSim. CloudSim offers a simulation framework that is both generic and extensible. This framework enables the modeling and simulation of app performance effortlessly. Developers are freed from the burden of worrying about the specifics of cloud-related frameworks and services when they make use of CloudSim.
2. With the help of a cloud analyst, configure cloud data centers and their regions for CloudSim. The Cloud Analyst aims to assist developers in simulating large-scale Cloud applications to understand such systems' performance better. It can configure cloud data centers and their regions for CloudSim.
3. Assign a user base for cloud data centers. Cloud service providers construct their data centers and offer users various services.
4. Configure the scheduling protocol and system gateway using the cloud analyst.
5. Run cloud simulation by assigning the Virtual Machines and completing the execution.

6. Start the RMA agent manager using the JADE framework, which is a framework for developing multi-agent systems. An RMA is a Java object that can be initiated on the command line as an ordinary or remote management agent.

7. Create agents for particular tasks.

8. Create a container with address details for each agent.

9. Establish connection/communication between agents and VMs. The host hardware is the component responsible for facilitating virtual machine (VM) connectivity to other network agents.

10. Control transmission way and calculate communication gap using the JADE sniffer agent tool. The Sniffer is fully integrated into the Jade environment and is very helpful when debugging agents' actions.

11. Print the gateway/agent/communication log.

12. Calculate IaaS and SaaS parameters using the cloud analyzer of this simulation (QoS). The comprehensive quality of service model integrates all of CloudSim's quality of service-related metrics.

13. The scheduling agent is optimized using PSO and ACO for flood prediction.

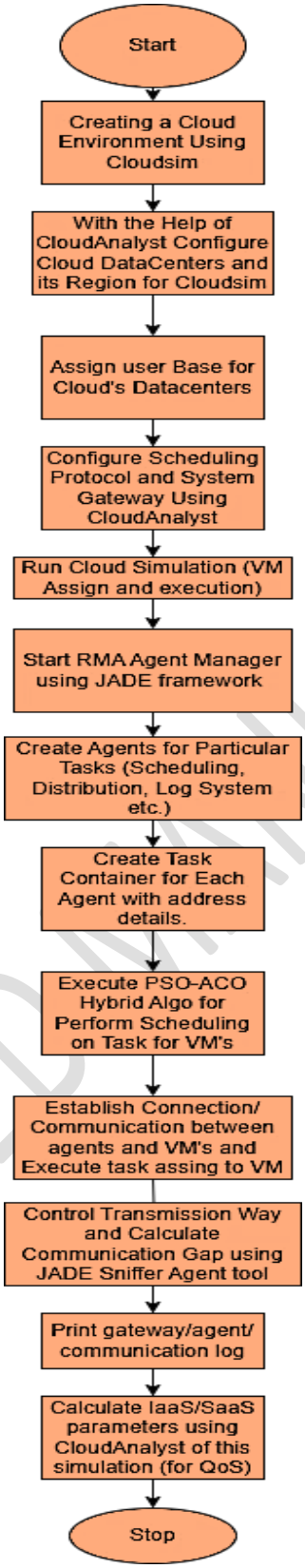


Fig3. Multi-agent-based scheduling algorithm

5. Results and discussion

5.1. Multi-Agent System using JADE in the cloud

A multi-agent-based technique for scheduling was executed using the JADE framework, and these findings were simulated using CLOUDSIM. Configuring hardware and software specifications of a cloud environment to ensure that those components can interoperate and communicate is called cloud configuration. Configuring the Cloud Environment is given in Fig. 4. The Cloud Environment, which is ready for simulation, is provided. The detailed results for the user base and data centers are represented. The experiment was conducted on a computer with the following specifications: Intel Core 2 Duo CPU, 2.10 GHz, 8 GB RAM, and a Windows 10 (64-bit) OS.

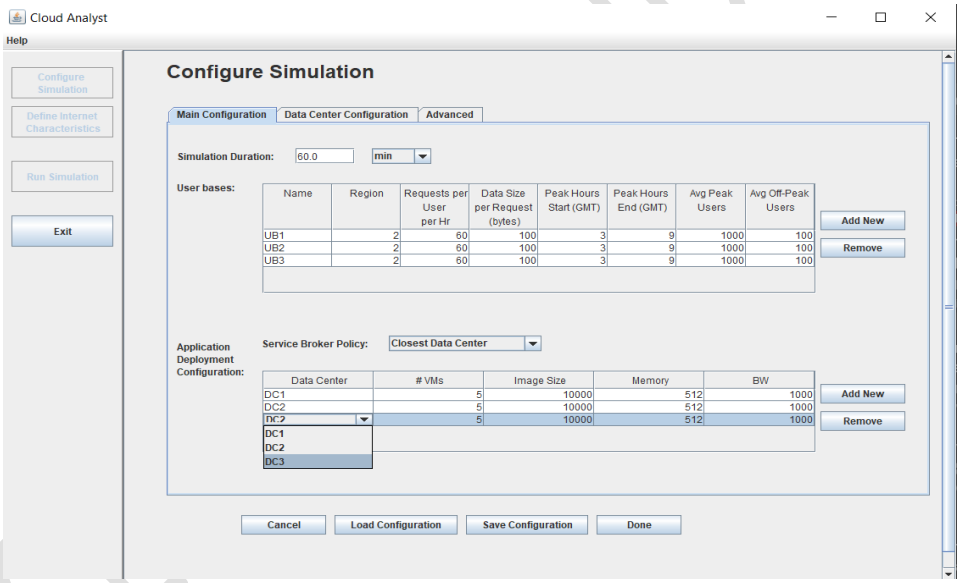


Fig 4. Configuring Cloud Environment

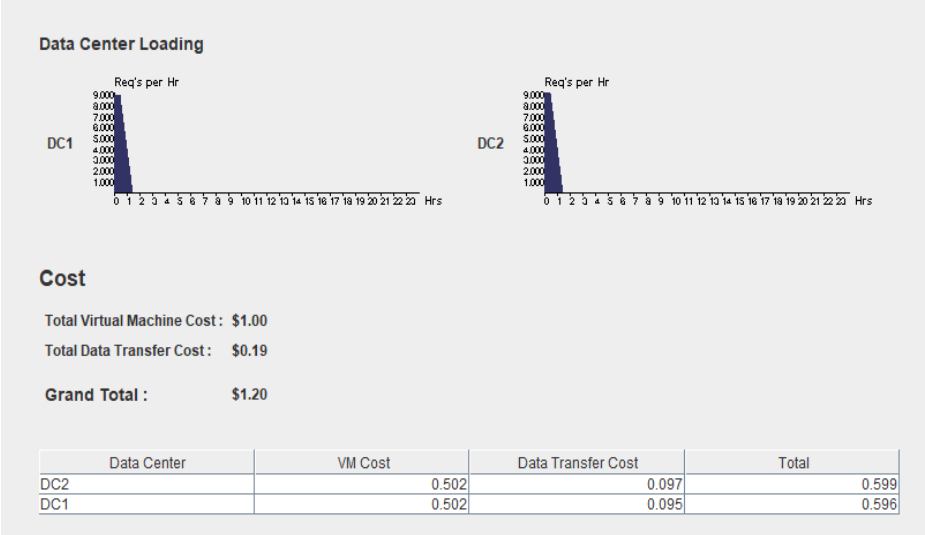


Fig 5. Detailed Results for User Base and Datacenters

5.2. Configuring for the data center network

The Cloud Virtual Machine Configurations for Datacenters, Network, and Individual Bases are given in Figs. 6. This shows how many VMs are created and how the cloudlets/requests are assigned to the VMs.

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0          SUCCESS        3             0       320        0.1          320.1
    5          SUCCESS        3             0       320        0.1          320.1
    1          SUCCESS        3             1       320        0.1          320.1
    6          SUCCESS        3             1       320        0.1          320.1
    2          SUCCESS        3             2       320        0.1          320.1
    7          SUCCESS        3             2       320        0.1          320.1
    4          SUCCESS        3             4       320        0.1          320.1
    9          SUCCESS        3             4       320        0.1          320.1
    3          SUCCESS        3             3       320        0.1          320.1
    8          SUCCESS        3             3       320        0.1          320.1
    101          SUCCESS          3             101          320          200.1          520.1
    106          SUCCESS          3             101          320          200.1          520.1
    103          SUCCESS          3             103          320          200.1          520.1
    108          SUCCESS          3             103          320          200.1          520.1
    100          SUCCESS          3             100          320          200.1          520.1
    105          SUCCESS          3             100          320          200.1          520.1
    102          SUCCESS          3             102          320          200.1          520.1
    107          SUCCESS          3             102          320          200.1          520.1
    104          SUCCESS          3             104          320          200.1          520.1
```

Fig. 6. Allocation of Cloudlets/tasks to VMs

Once the connection is established between the agents and VMs, the transmission process is initiated. Configuring the cloudlets and the VMs required for the users is done after establishing the connection. The calculated parameters are the cloudlet parameters, such as the size, file length, and output dimension. The VM parameters calculated are the MIPS, Size, RAM, and Bandwidth. The experiment uses 50 virtual machines distributed across 5 data centers. The cloudlets refer to the user's

jobs/tasks. Here, we considered different bandwidths of user requests. The RAM size and other specifications are also given. Then, the tasks are assigned to the agents where it is being processed. Next, the transmission way is controlled, and the communication gap between the agents and VMs is calculated using the JADE sniffer agent tool. Logs are created once the communication gap is calculated and transmission occurs. Gateway Communication Logs are given.

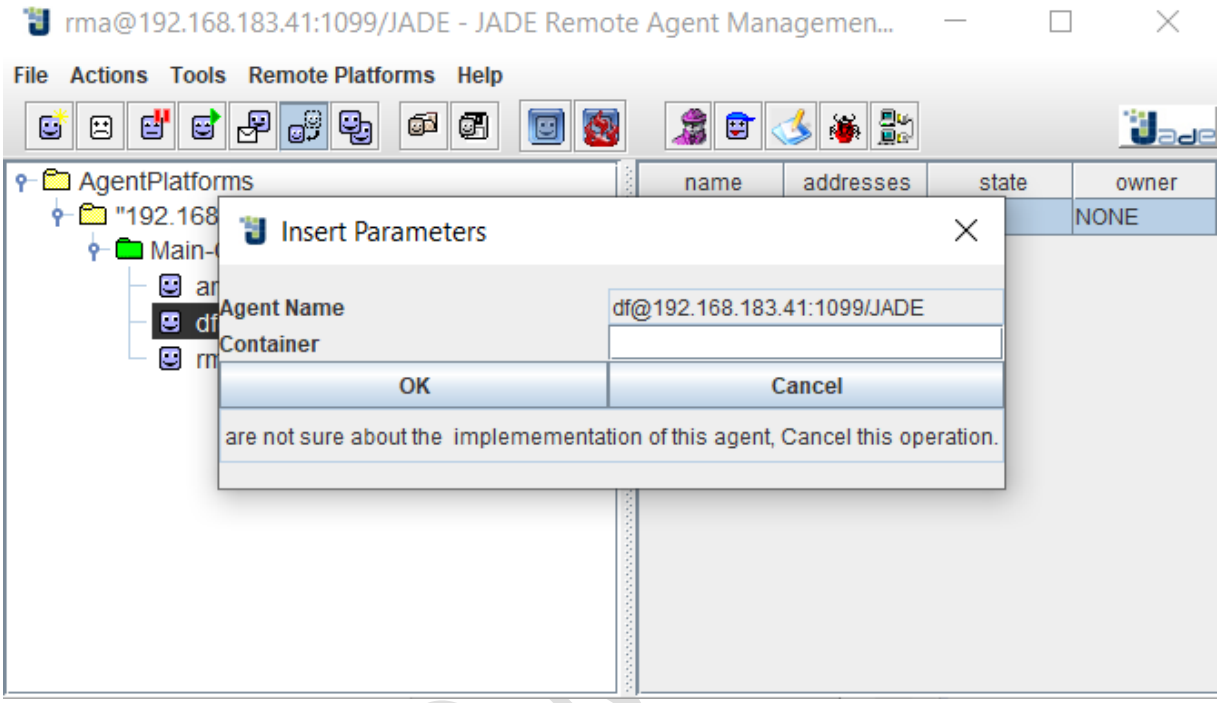


Fig 7: Transmission

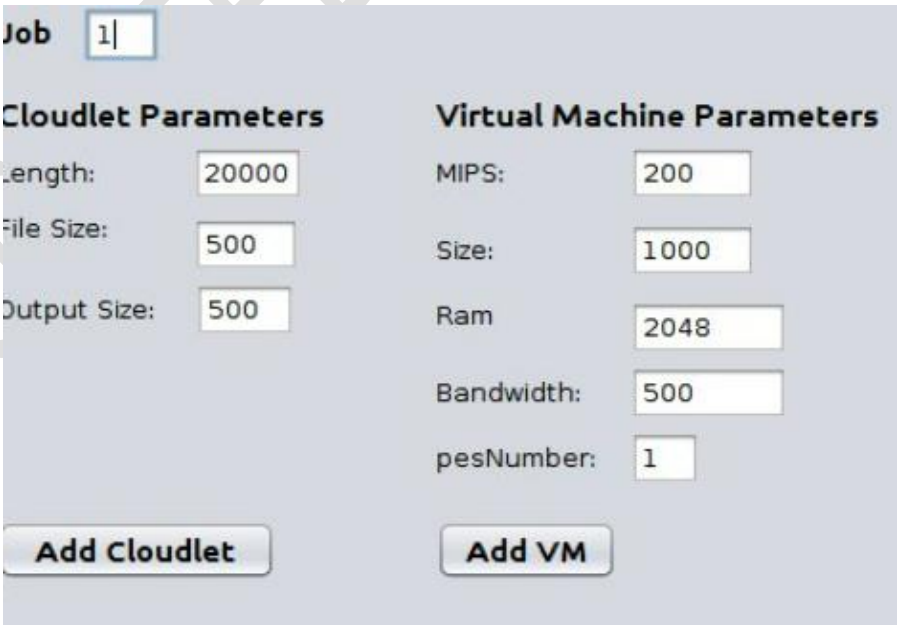


Fig 8. Configuration Parameters

5.3. Scheduling Agent using Particle Swarm Optimization and Ant Colony Optimization

The following process optimizes the scheduling agent using PSO and ACO.

```
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID        Time    Start Time    Finish Time    Waiting Time
     07        SUCCESS         02           02        2110.91       00.1          2111.01           00
     00        SUCCESS         03           03        2818.05       00.1          2818.15           00
     03        SUCCESS         05           05        2897.27       00.1          2897.37           00
     06        SUCCESS         04           04        3449.86       00.1          3449.96           00
     05        SUCCESS         05           05        1497.18       2897.37          4394.56           2897.27
     01        SUCCESS         06           06        4401.24       00.1          4401.34           00
     09        SUCCESS         04           04        1952.9        3449.96          5402.86           3449.86
     14        SUCCESS         02           02        3562.27       2111.01          5673.28           2110.91
     08        SUCCESS         05           05        1372.76       4394.56          5767.32           4394.46
     02        SUCCESS         03           03        3336.04       2818.15          6154.19           2818.05
     04        SUCCESS         06           06        2043.83       4401.34          6445.17           4401.24
     16        SUCCESS         02           02        1412.68       5673.28          7085.96           5673.18
     12        SUCCESS         03           03        1358.39       6154.19          7512.58           6154.09
     10        SUCCESS         04           04        2209.55       5402.86          7612.42           5402.76
     28        SUCCESS         02           02        1624.34       7085.96          8710.3            7085.86
     17        SUCCESS         06           06        3169.1        6445.17          9614.27           6445.07
     11        SUCCESS         05           05        4080.21       5767.32          9847.52           5767.22
     26        SUCCESS         03           03        2362.04       7512.58          9874.62           7512.48
     15        SUCCESS         04           04        2588.44       7612.42          10200.85          7612.32
     18        SUCCESS         06           06        1656.31       9614.27          11270.58          9614.17
     13        SUCCESS         05           05        2015.81       9847.52          11863.34          9847.42
     20        SUCCESS         04           04        2799.72       10200.85         13000.57          10200.75
     19        SUCCESS         06           06        2512.85       11270.58         13783.42          11270.48
     23        SUCCESS         05           05        2892.5        11863.34         14755.84          11863.24
     21        SUCCESS         04           04        2397.97       13000.57         15398.54          13000.47
     22        SUCCESS         06           06        2135.95       13783.42         15919.37          13783.32
     27        SUCCESS         05           05        2835          14755.84         17590.84          14755.74
     24        SUCCESS         04           04        2569.5        15398.54         17968.04          15398.44
     25        SUCCESS         06           06        3065.36       15919.37         18984.73          15919.27
     29        SUCCESS         04           04        1440.67       17968.04         19408.71          17967.94

The best fitness value: 4285.91501827442
Best makespan using MPSO: 4918.600190779811
```

Fig 9. Scheduling of Tasks using PSO and ACO optimizer by agents

The time taken for the proposed algorithm to complete the scheduling process is shown in Fig. 9. Fig. 10 shows the linear increase in the execution time as the number of tasks/user requests increases. When the number of cloudlets/tasks is 6, the execution time is 15ms. When the number of tasks/cloudlets is 1000, the execution time is 804ms. Using a hybrid algorithm and agents, the time to process the user's request has been considerably reduced.

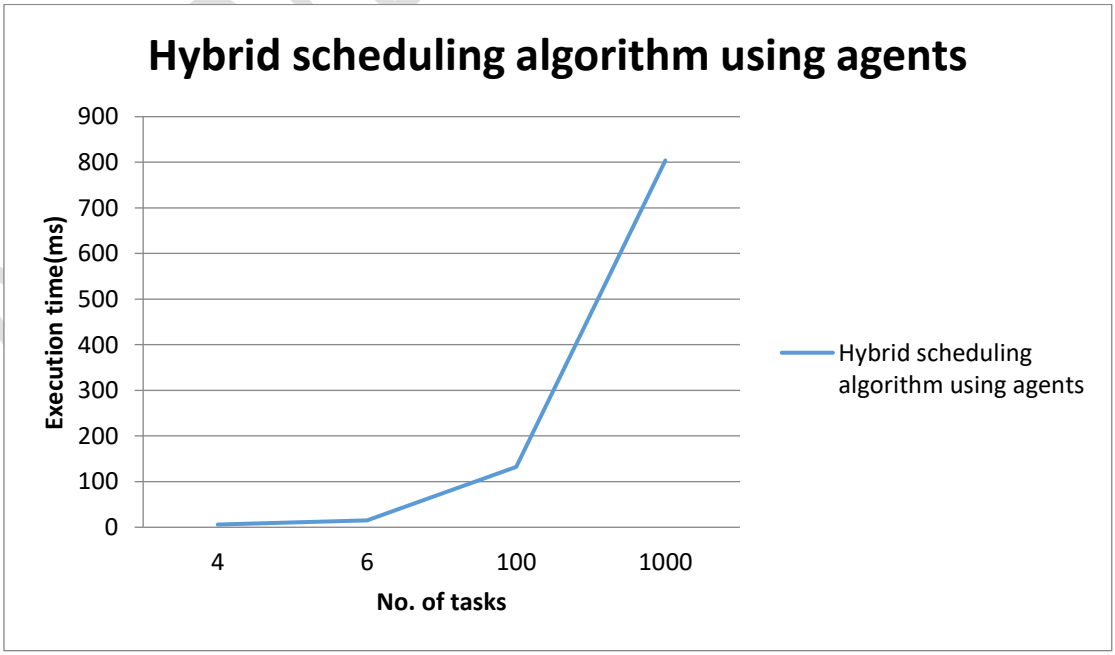


Fig 10. Execution time variation with varying the number of user tasks

Table 1 shows the makespan value and the optimum fitness point by varying the number of user requests and VMs. When the number of user requests and VMs is 5, the optimum fitness point obtained is 0.2093. Several optimum points have been obtained by changing the input parameters many times. The agents pick the best optimum point on behalf of the user and display it to the user. The makespan value is the same for almost all the input parameters. This is because of the parallel execution of multiple agents that perform the scheduling task successfully.

Table 1. Makespan and fitness point value for varying input parameters

No. of user requests	No. of VMs	Makespan value	Fitness Point
5	5	0.032	0.2093
10	5	0.032	0.3774
10	10	0.032	0.1189
15	10	0.032	0.2791
15	15	0.032	0.5013
20	15	0.032	1.1455

The execution time for processing the user request has been compared with two other scheduling techniques in the cloud that use GA and ACO algorithms. The time taken to process user requests using GA and ACO is very high compared to our proposed hybrid algorithm, as shown in Table 2. During the scheduling process, the mapping of user requests to the VMs is optimized using the hybrid algorithm and the agents. Thus, the success rate of service scheduling in the cloud is higher using our proposed approach due to the parallel execution of the user request. This hybrid algorithm combines agent-based coordination and optimization methodologies to adjust to changing cloud resource needs. Because of this, it can dynamically scale resources, manage peak traffic, and maintain system stability in extremely dynamic cloud settings. Figure 11 gives the performance analysis of our proposed work. The graph has been plotted; the results are shown below in Figs. 11 to 17.

Table 2. Performance Analysis

No. of Tasks/Request	Execution time using GA (ms)	Execution time using ACO (ms)	Proposed Hybrid Algorithm (ACO+PSO using agents) (ms)
4	12	12	6
6	30	30	15
100	210	210	132
1000	1500	1506	804

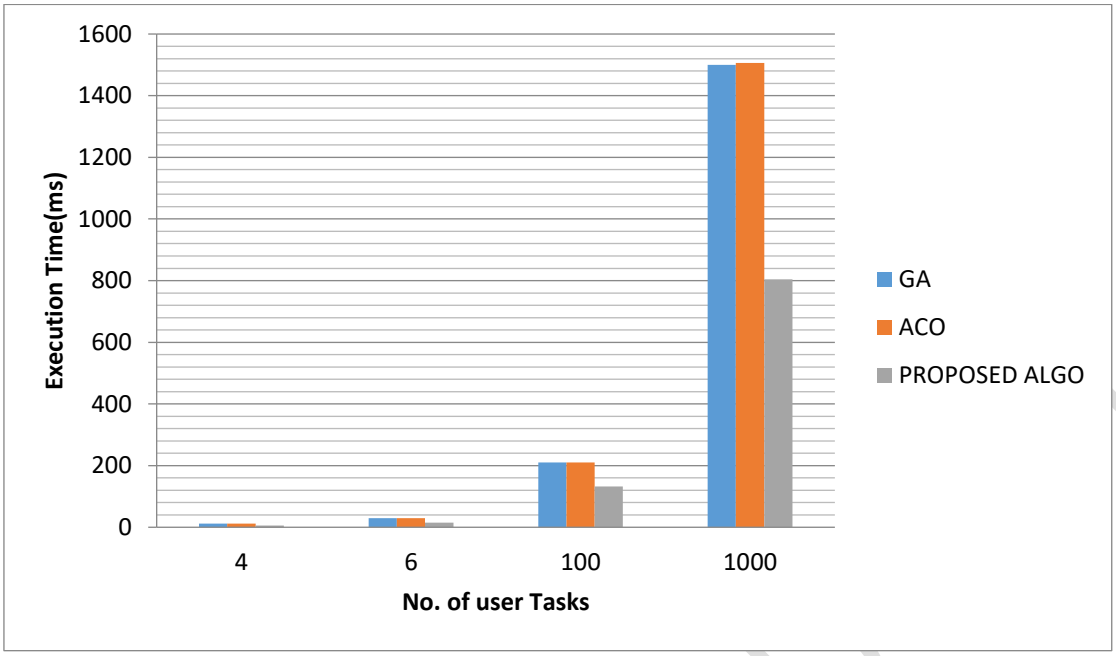


Fig 11. Performance analysis of the processing time

5.4. Prediction outcomes

Monitoring the changes in mean average precision (mAP) and loss during training using different learning strategies enabled a comparative analysis of frozen learning versus regular learning for flood depth recognition. Table 3 presents the mAP and loss values recorded at 50 training epochs. Results indicate that regular learning achieved a higher mAP at 200 epochs than frozen learning, exhibiting lower loss values. This demonstrates that regular learning outperformed frozen learning regarding accuracy and convergence. Moreover, adopting a mixed-learning strategy further improved the model's overall performance. This approach initially employed frozen learning for the first 90 epochs, then transitioned to regular learning for the remaining epochs, resulting in more effective training and enhanced predictive accuracy.

Table 3. Flood depth recognition based on cloud storage

Data attributes	Precision (%)	Recall (%)	AP (%)
1	96	95	98.1
2	87	97	98.2
3	92	96	99.4
4	95	97.5	97.4

5	95.7	96.5	98.6
6	95	98	99.2
7	92	81	93
Map			98.2

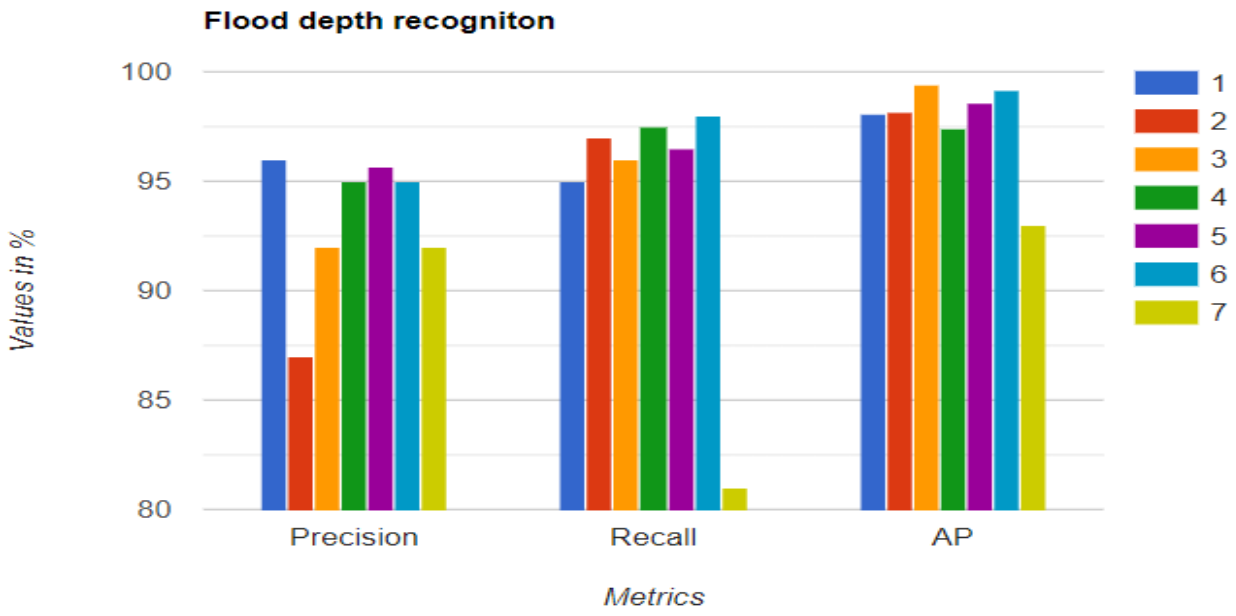


Fig 12. Flood depth recognition

5.5. Performance metrics

Hydrological characteristics (such as rainfall, water level, and discharge) are mapped to three flood risk categories by the fuzzy inference system: low, moderate, and high. A multi-agent system coordinates the optimization. In particular, 1) Data agents are in charge of gathering and standardizing both historical and real-time data; 2) Optimization agents implement the hybrid PSO-ACO logic. While ACO agents locally refine fuzzy rules, PSO agents search the global parameter space; 3) Fuzzy rule repositories are maintained, inference agents carry out classification; and 4) QoS Monitoring Agents continuously monitor accuracy, latency, and resource usage. In the cloud environment, agents communicate via a shared message bus in real-time. In this investigation, mean average precision (MAP) was used to measure the reliability of flood image identification. This type of measurement is frequently employed in classifying and identifying objects. The MAP was computed using the modeling results' precision and recall metrics. The proportion of accurately predicted positive findings to all true positive observations is known as recall. Where FP (false positive) is the number of samples that are incorrectly

recognized, FN (false negative) is the number of unacknowledged results, and TP (true positive) is the number of samples that are correctly identified or classified, precision is the fraction of exactly computed positive findings to all positive ones. The average precision, or AP, is the area under the precision-recall curve of each category's prediction results. Using the observations above, the MAP can be evaluated as follows. There are n overall categories. The $p(r)$ represents the curve of Precision-Recall, where r indicates the recall score from Eq. (7). The $p(r)$ represents the most significant amount of the corresponding accuracy at a given interval, say 0.1, when the recall score increases from zero to one. Latency: Mean inference time measured by inference agents across prediction tasks. Cost: Summation of cloud resource usage per hour, logged by QoS agents.

$$Precision = \frac{TP}{FP + TP} \tag{6}$$

$$Recall = \frac{TP}{FN + TP} \tag{7}$$

$$mAP = \frac{\sum_1^n AP}{n} \tag{8}$$

$$AP = \int_0^1 p(r)dr \tag{9}$$

Table 4. MAP and loss analysis based on cloud data

Epochs	Deep		Regular		Frozen	
	Map	Loss	Map	Loss	Map	Loss
50	95	33	90	31	88	33
100	95	33	83	28	89	23
150	97	33	85	26	93	18
200	93	32	85	25	93	17

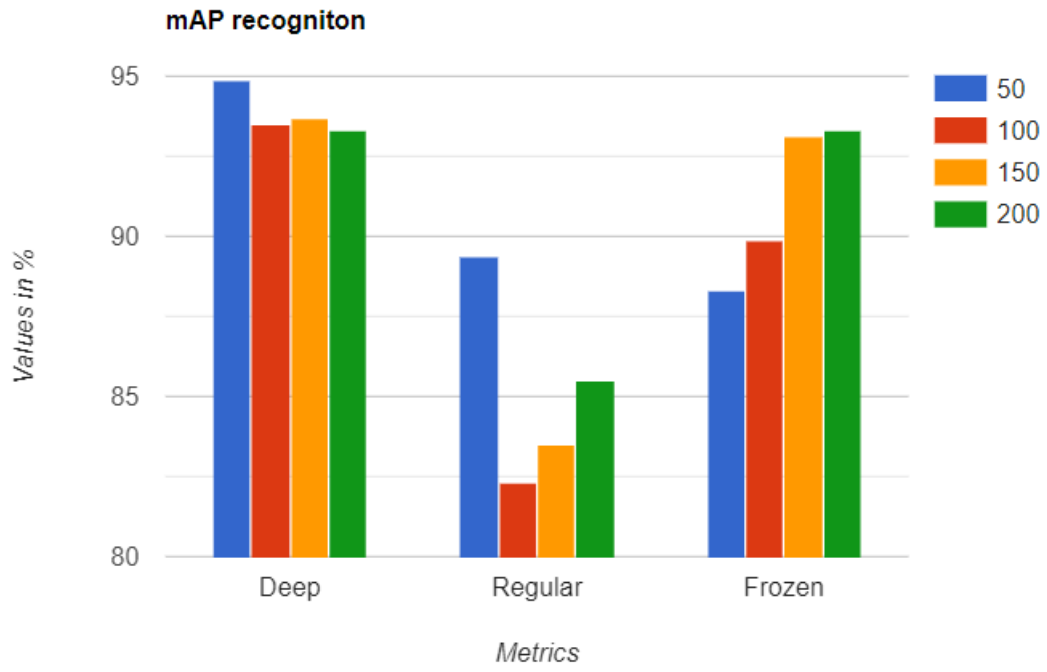


Fig 13. Map of proposed vs. existing

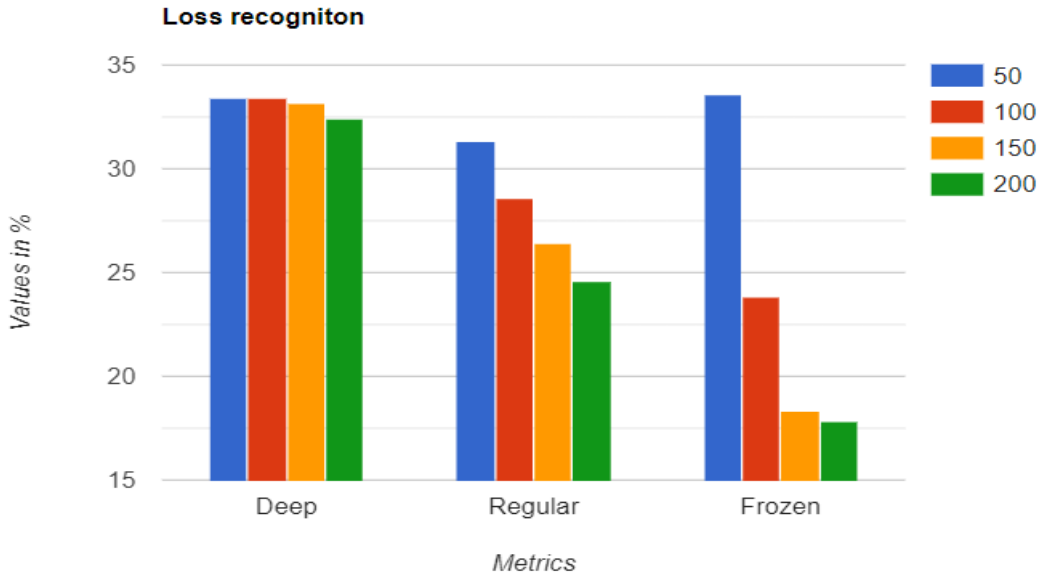


Fig 14. Loss Recognition

Table 5. mAP training-based average precision

Scenario	Map (%)
1	88
2	85

3	91
4	91
5	92
6	98
7	97

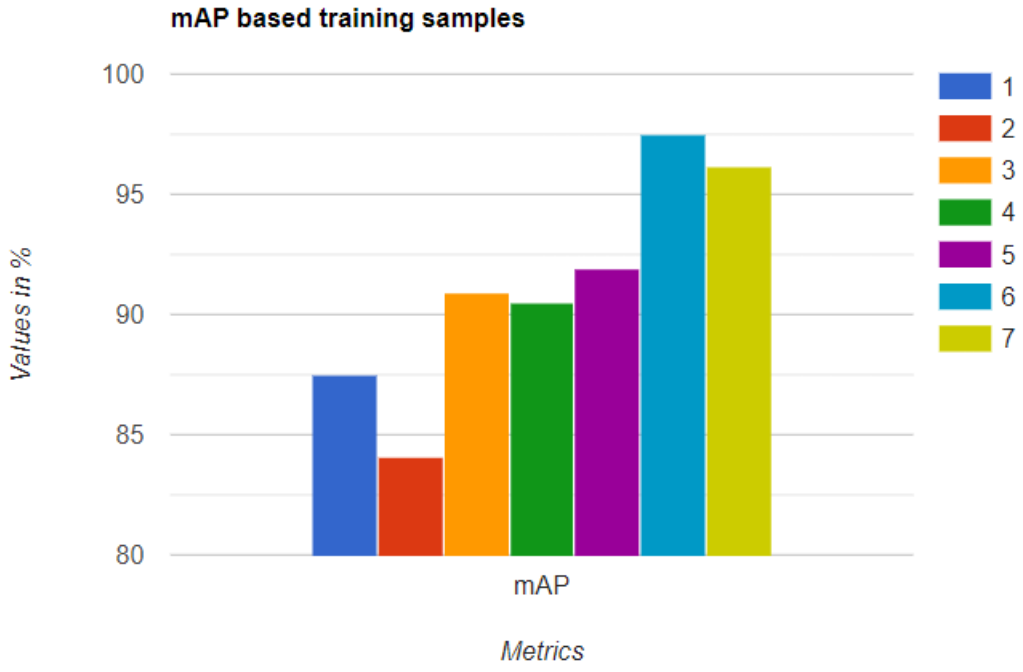


Fig 15. mAP-based training samples

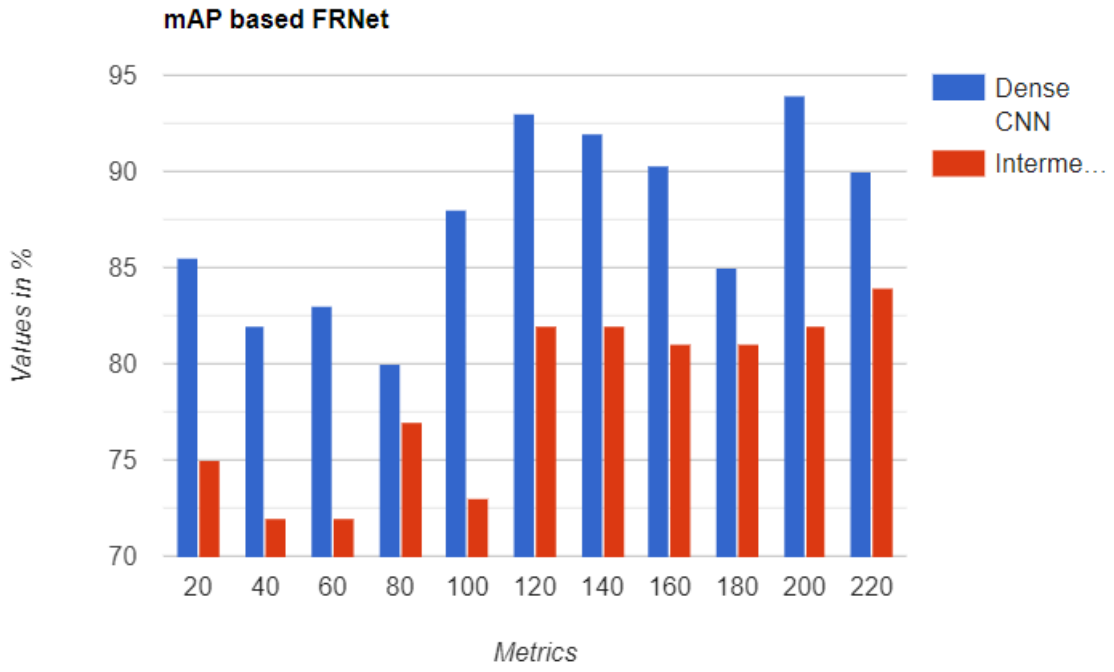


Fig 16. mAP-based learning model

Fig 17. Predicted outcomes

Key QoS metrics collected while reviewing the multi-agent hybrid model and its distinct equivalents are compiled in the table below. Consistent real-time ingestion was guaranteed by data agents, optimization agents continuously adjusted fuzzy rules, and QoS agents reported system log metrics. All tests were conducted in a cloud simulation environment with identical setups to ensure fairness. Table 6 compares baseline PSO, ASO, and the proposed hybrid model regarding latency, accuracy, precision, recall, F1-score, and cost. Based on this comparison, the hybrid model gives promising results compared to baseline models.

Table 6. Performance evaluation

Method	Accuracy (%)	Latency (s)	Precision	Recall	F1-Score	Cost (USD/hr)
PSO Only	90.2	2.4	0.89	0.90	0.895	0.42
ASO Only	88.9	3.0	0.86	0.87	0.865	0.47
Hybrid model	94.7	1.1	0.94	0.95	0.945	0.32

The model's performance is shown in the cost, latency, and accuracy metrics figures, as in Figs. 18 to 21. The hybrid multi-agent model performs better in every category than stand-alone optimization techniques.

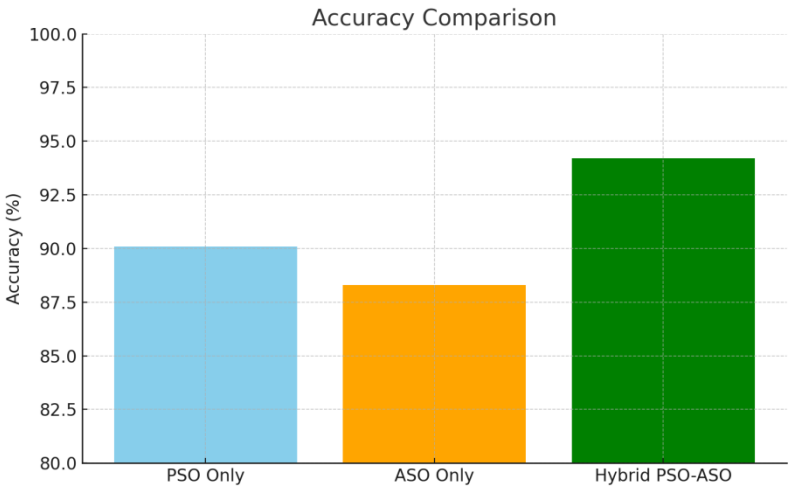


Fig 18. Accuracy of baseline and proposed model

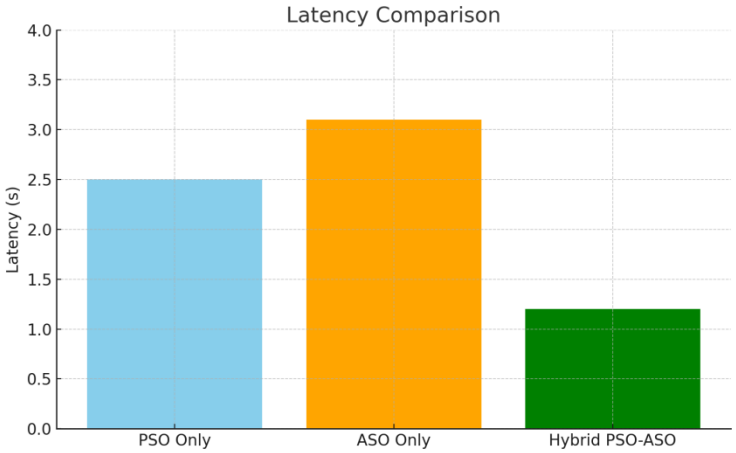


Fig 19. Latency comparison of baseline and proposed model

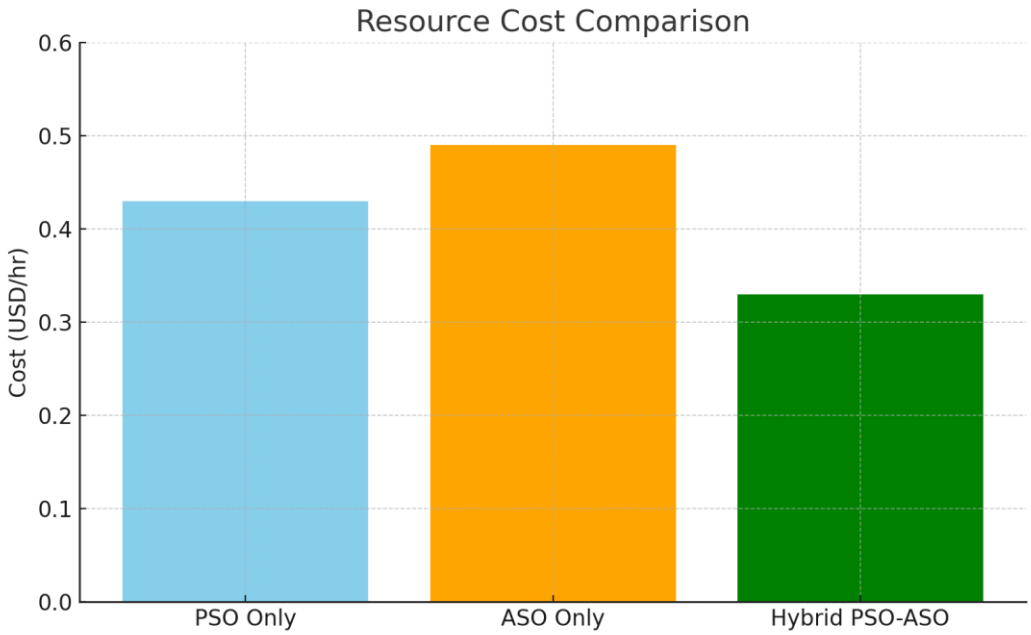


Fig 20. Cost comparison of the baseline and proposed model

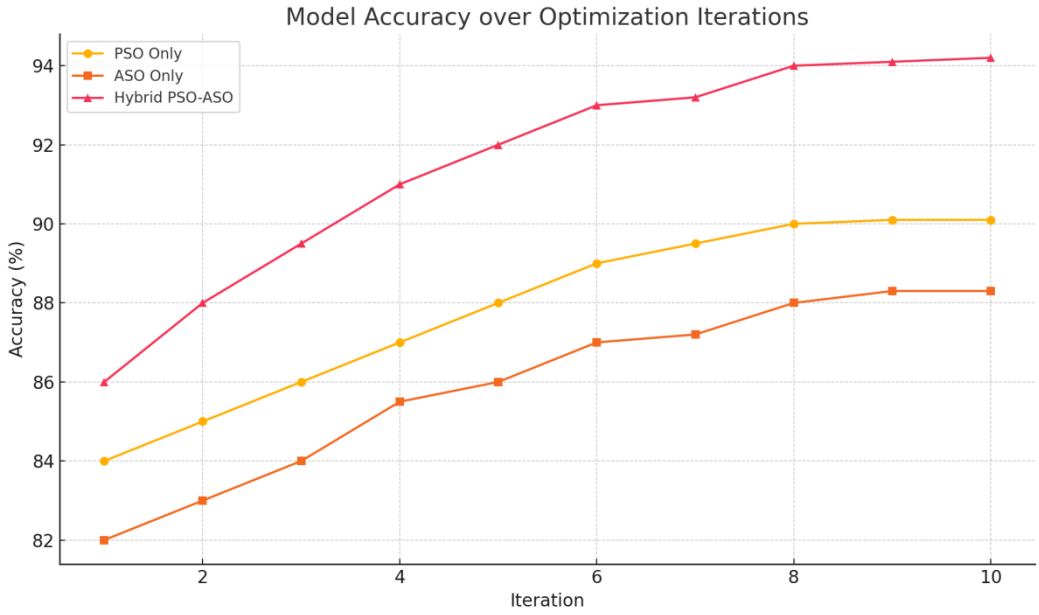


Fig 21. Accuracy of baseline and proposed model over multiple iterations

Because of intelligent agent orchestration and parallelized optimization, the multi-agent hybrid optimization approach achieves lower latency and higher prediction accuracy. Scalability and fault tolerance are ensured by cloud integration, while the fuzzy model guarantees interpretability. The system's practical use in real-time disaster prediction is validated by its exceptional responsiveness under various workloads.

6. Conclusion

Various classification techniques have been executed on flood data sets in this research. Initially, the outcomes are estimated using a stand-alone machine, which shows that ACO and PSO yield the highest accuracy of 96.5% and enhance efficiency on different assessment criteria. Next, these approaches are also verified in a cloud platform, which is more precise than a stand-alone machine. In a cloud platform, ACO and PSO yield an accuracy of 96.5% with an AUC value of 0.9. At last, the processing time in the cloud platform shows a better outcome, which is 0.10ms compared to 6.3ms using the stand-alone machine. As a result, it can be confirmed that the suggested hybrid approach functions better in a cloud context regarding reliability and execution time. By providing advance notice of flood events, the proposed strategy is anticipated to lower the amount of flood-related fatalities and other damages.

A hybrid Multi-Agent Algorithm combining PSO and ACO optimized the scheduling process. ACO has positive feedback and parallelism and can be easily combined with other algorithms. PSO is used because ACO needs to be optimized through repeated experiments. Here, the parameters in ACO have been optimized through PSO, which increased the optimal performance of the PSO. In this proposed research, multi-agents and the above-mentioned hybrid scheduling algorithm are used, and the optimum execution time is calculated and tested using flood samples. This is mainly due to aging, which, in turn, increases utilization. The optimum fitness points were obtained using this algorithm, which proved highly efficient. In terms of upcoming works, tests can be carried out on a broader scale to evaluate the scalability of the agent-based scheduling technique in real-world scenarios by putting the test bed into operation. This work presented an entirely novel cloud-based multi-agent hybrid optimization framework for flood prediction. Its benefit in raising QoS metrics like accuracy, latency, and cost is confirmed by experimental results. In the future, deep learning models and satellite imaging will be integrated with real-world deployment. In the future, different algorithms, such as ABC, PSO, FP Growth, etc., can also be incorporated to make better decisions and avoid flood-prone circumstances. Some more attributes can be added to the current flood data set to enhance the effectiveness of the proposed hybrid approach. Also, deep learning models and satellite imaging will be integrated with real-world deployment.

REFERENCES

A. Mousavi, M. J. Nordin, Z. A. Othman (2012), “Ontology-driven coordination model for multiagent-based mobile workforce brokering systems," *Applied Intelligence*, **36(3),** pp. 768–787.

Abdulhamid, S.I.M.; AbdLatiff, M.S.; Abdul-Salaam, G.; HussainMadni, S.H. (2016). Secure scientific applications scheduling technique for cloud computing environment using global league championship algorithm. *PLoS ONE*, **11**, e0158102.

Abdullahi, M.; Ngadi, M.A. (2016), Symbiotic organism search optimization based task scheduling in the cloud computing environment, *Future Gener. Comput*. Syst. **56**, 640–650.

Arshad B, Ogie R, Barthelemy J, Pradhan B, Verstaevel N, Perez P (2019) Computer vision and IoT-based sensors in flood monitoring and mapping: A systematic review. Sensors (Switzerland) **19(22)**:5012.

Bai Y, Zhao N, Zhang R, Zeng X (2018) Stormwater management of low impact development in urban areas based on SWMM. Water (Switzerland) **11(1)**:33.

Basnyat B, Roy N, Gangopadhyay A (2018) A flash flood categorization system using scene-text recognition. In: 2018 *IEEE International Conference on Smart Computing* (SMARTCOMP), 147–154.

Batista, B.G.; Estrella, J.C.; Ferreira, C.H.G.; Filho, D.M.L.; Nakamura, L.H.V.; Reiff-Marganiec, S.; Santana, M.J.; Santana, R.H.C (2015). Performance evaluation of resource management in cloud computing environments. *PLoS ONE*, **10**, e0141914.

Bochkovskiy A, Wang CY, Liao HYM (2020) Yolov4: Optimal speed and accuracy of object detection. *ArXiv Preprint ArXiv*:2004.10934

Bulti DT, Abebe BG (2020) A review of flood modeling methods for urban pluvial flood application. In: Modeling Earth Systems and Environment. In: Modeling earth systems and environment, **6(3)**. *Springer Science and Business Media Deutschland*, 1293–1302.

Chia MY, Koo CH, Huang YF, Di Chan W, Pang JY (2023) Artificial intelligence generated synthetic datasets to remedy water quality index estimation data scarcity. *Water Resour Management* 1–6.

D. A.Amer, G.Attiya, & I. Ziedan (2023), "An efficient multi-objective scheduling algorithm based on spider monkey and ant colony optimization in cloud computing," *Cluster Computing*, **1(5),** pp. 1-21.

D. Isern, A. Moreno, D. Sánchez, Á. Hajnal, G. Pedone, L. Varga (2011), "Agent-based execution of personalized home care treatments," *Applied Intelligence*, **34(2),** pp. 155–180

Domanal, S.G.; Guddeti, R.M.R.; Buyya (2017), R., A Hybrid Bio-Inspired Algorithm for Scheduling and Resource Management in Cloud Environment. *IEEE Trans. Serv. Comput*. **13**, 3–15.

F. Both, M. Hoogendoorn, A. Mee, J. Treur, M. Vos (2012), "An intelligent agent model with awareness of workflow progress," *Applied Intelligence*, **36(2),** 498–510.

Faramarzzadeh M, Ehsani MR, Akbari M, Rahimi R, Moghaddam M, Behrangi A, Klöve B, Haghighi AT, Oussalah M (2023) Application of machine learning and remote sensing for gap-filling daily precipitation data of a sparsely gauged basin in East Africa. *Environ Process* **10 (1):**8.

Feng B, Zhang Y, Bourke R (2021) Urbanization impacts on flood risks based on urban growth data and coupled flood models. *Nat Hazards***106(1):**613–627

Gauen K, Dailey R, Laiman J, Zi Y, Asokan N, Lu YH, Thiruvathukal GK, Shyu ML, Chen SC (2017) Comparison of visual datasets for machine learning. In: 2017 IEEE *International Conference on Information Reuse and Integration* (IRI). 346–355.

Guo K, Guan M, Yu D (2021) Urban surface water flood modeling: a comprehensive review of current models and future challenges. *In: Hydrology and Earth System Sciences*, **25(5)**. Copernicus GmbH, pp 2843–2860.

J. Kang, K. M. Sim (2012), "A multi-agent brokering protocol for supporting Grid resource discovery," Applied Intelligence, **1(2).**

K. M. Sim, "Agent-based cloud commerce (2010)", In Proc *IEEE international conference on industrial engineering and engineering management*, Hong Kong, **10(1),**717–721.

K. M. Sim, "Agent-based cloud computing (2011)", IEEE *Trans ServComput*, **15(3),** 16-20.

K. M. Sim (2012), "Complex and concurrent negotiations for multiple interrelated e-markets," IEEE Trans Syst Man Cybern B Cybern, 16(3), 18-20.

K. M. Sim (2006), "Guest editorial: agent-based grid computing," *Applied Intelligence*, **25(2),** 127–129.

K. M. Sim (2010), "Towards complex negotiation for cloud economy," In Chang RS et al. (eds) GPC. LNCS, **6104**. *Springer, Heidelberg*, 395–406.

M. Kalra, S. Singh (2015), "A review of meta-heuristic scheduling techniques in cloud computing," *Egypt. Informatics J.***16(3)**, 275–295.

Nadimi-Shahraki, M.H.; Zamani (2022), H. DMDE: Diversity-maintained multi-trial vector differential evolution algorithm for nondecomposition large-scale global optimization. *Expert Syst. Appl*. **198**, 116895.

Nadimi-Shahraki, M.H.; Zamani, H.; Mirjalili (2022), Enhanced whale optimization algorithm for medical feature selection: A COVID-19 case study. *Comput. Biol. Med.***. 148,** 105858.

Nanjappan, M.; Natesan, G.; Krishna Doss, P. (2021), An Adaptive Neuro-Fuzzy Inference System and Black Widow Optimization Approach for Optimal Resource Utilization and Task Scheduling in a Cloud Environment. *Wireless. Pers. Commun*. **121,** 1891–1916.

Natesan, G., Manikandan, N., Pradeep, K., &SherlyPuspha Annabel, L. (2023). Task scheduling based on minimization of makespan and energy consumption using a binary GWO algorithm in a cloud environment. *Peer-to-Peer Networking and Applications*, **16(5),** 2560-2573.

Pirozmand, P., Jalalinejad, H., Hosseinabadi, A. A. R., Mirkamali, S., & Li, Y. (2023), An improved particle swarm optimization algorithm for task scheduling in cloud computing, Journal of Ambient Intelligence and Humanized Computing, 14(4), 4313-4327.

Ramezani, F.; Lu, J.; Hussain, F.K. (2014). Task-Based System Load Balancing in Cloud Computing Using Particle Swarm Optimization. *Int. J. Parallel Program*, **42**, 739–754.

S.Torabi, F.Safi-Esfahani (2018), "A dynamic task scheduling framework based on chicken swarm and improved raven roosting optimization methods in cloud computing," *J. Supercomputer*, **74(6)**, 2581–2626.

S.Wen, R.Han, C. H.Liu, & L. Y. Chen (2023), "Fast DRL-based scheduler configuration tuning for reducing tail latency in edge-cloud jobs," *Journal of Cloud Computing*, **12(1)**, pp. 1-32.

Sardaraz, M.; Tahir, M. (2016), A parallel multi-objective genetic algorithm for scheduling scientific workflows in cloud computing, *Int. J. Distrib. Sens. Netw*. **16**, 1550147720949142.

U.Jambulingam, & K. Balasubadra (2023), "An Energy-Aware Agent-Based Resource Allocation Using Targeted Load Balancer for Improving Quality of Service in Cloud Environment," *Cybernetics and Systems*, 1-21.

Zamani, H.; Nadimi-Shahraki, M.H.; Gandomi, A.H. (2022). Starling murmuration optimizer: A novel bio-inspired algorithm for global and engineering optimization. *Comput. Methods Appl. Mech. Eng.*, **392**, 114616.

Zamani, H.; Nadimi-Shahraki, M.H.; Gandomi, A.H., QANA (2012): Quantum-based avian navigation optimizer algorithm. *Eng. Appl. Artif. Intell* **. 104**, 104314.

Zuo, L.; Shu, L.; Dong, S.; Zhu, C.; Hara, T. (2015). A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. *IEEE Access*, **3**, 2687–2699.

APPENDIX

ACO	Ant Colony Optimization
ANFIS	Adaptive Network-based Fuzzy Inference System
ANN	Artificial Neural Networks
AUC	Area Under Curve
BWO	Black Widow Optimization
CPU	Central Processing Unit
TP	True Positive
TN	True Negative
PSO	Particle Swarm Optimization
CC	Cloud Computing
FN	False Negative
FP	False Positive
GA	Genetic Algorithm
HNet	Hierarchical Network Layer
IMD	Indian Meteorological Department
SVM	Support Vector Machine

mAP	Mean Average Precision
MAS	Multi-Agent System
MIPS	Million Instruction Per Second
MOWOS	Multi-Objective Workflow Optimization Strategy
ML	Machine Learning
ms	Milli second
OS	Operating System
PCA	Principle Component Analysis
RAM	Random Access Memory
SMO	Spider Monkey Optimization
QOS	Quality of Service
VM	virtual machine