RESEARCH PAPER

SSA: An Improved Sparrow Search Algorithm for Cloud Cost Optimization in Healthcare Domain

Mayur Patel[1], Tejas Thakkar[2]
[1]Computer Science, The Charutar Vidya Mandal (CVM) University
Email: mayur.patel@cvmu.edu.in
[2]Computer Science, The Charutar Vidya Mandal (CVM) University
Email: tejas.thakkar@cvmu.edu.in

ABSTRACT

Cloud computing enables enterprises to efficiently transfer, compute, and host applications with access to a diverse set of services. The dynamic and heterogeneous nature of cloud resources requires an intelligent and adaptable task scheduling mechanism to manage concurrent user demands. Inefficient scheduling can lead to resource underutilization or overutilization, degrading system performance and increasing operational costs. To tackle these challenges, this study explores the use of swarm intelligence-based metaheuristic optimization techniques, which have shown strong potential in solving complex scheduling problems due to their adaptability and efficiency. This paper introduces novel improved swarm intelligence algorithms applied to task scheduling in cloud computing environments. It highlights the rationale for selecting swarm intelligence over traditional optimization methods, emphasizing their ability to handle the high dimensionality and dynamic nature of cloud systems. The study compares various swarm-based algorithms—such as Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Sparrow Search Algorithm (SSA)—based on key performance metrics including execution time, load balancing, resource utilization, and energy efficiency. It evaluates different simulation tools used for testing these algorithms and discusses current challenges, proposing future directions for research in this domain.

Keywords: Cloud computing; whale optimization; health care; resource optimization.
How to cite this article: Patel M, Thakkar T. ISSA: An Improved Sparrow Search Algorithm for Cloud Cost Optimization in Healthcare Domain. Int J Drug Deliv Technol. 2026;16(53s): 368.
DOI: 10.25258/ijddt.16.53s.84
Source of support: Nil.
Conflict of interest: None.

1. **Introduction:**

Technological developments in area of cloud computing enable a gateway to improved healthcare services that enhance the well-being of medical personnel and patients. There are several advantages to combining an e-health system with the cloud [1]. Different kinds of electronic systems are operational in the current situation. The electronic health record (EHR) is more expensive in traditional health systems, and maintenance tasks take longer as a result [2]. The study of big data for health service solutions has significantly altered the medical profession and enhanced healthcare services with the adoption of cloud computing and the Internet of Things (IoT) healthcare paradigm [3]. The necessary source presents a significant obstacle for bringing data into a cloud-IoT system [4]. The figure 1 illustrates the IoT cloud-based architecture designed for the proposed ISSA (Improved Sparrow Search Algorithm) technique. The architecture integrates multiple layers to effectively manage and process healthcare tasks in a cloud environment.

Layers of the IoT Cloud-Based Architecture
Things Layer

- **Components:** This layer includes various IoT devices such as environmental sensors (temperature, location, camera) and biomedical sensors (blood pressure, EEG, heart rate, body temperature, respiration rate).
- **Function:** It monitors the patient's health status, generates healthcare data, and transmits this data to the upper layer through Bluetooth or WiFi.

Data Collector Layer

- **Components:** This layer comprises mobile phones or smartwatches that act as intermediaries.
- **Function:** It collects healthcare data from the sensors and devices monitoring the patient's health status. The collected data is then transmitted to the cloud through the Internet.

Cloud Layer

The cloud layer is divided into four key modules:

Task Manager

- ✧ **Function:** Receives all tasks from the data collector layer. It handles a set of independent tasks submitted by users, classifies them based on criticality, and sorts them in descending order based on their length.

Task Scheduler

- ✧ **Function:** Responsible for scheduling and mapping the sorted tasks to the available Virtual Machines (VMs) based on the Task Balancing technique.

VM Manager

- ✧ **Function:** Monitors the resources of the VMs to ensure optimal allocation and usage.

Data Storage

- ✧ **Function:** Provides permanent storage for patient data and results. This data is accessible to stakeholders, including patients and doctors, ensuring that essential information is readily available for decision-making and medical interventions.

Process Flow

- ✧ **Task Submission:**

Users of medical services submit their tasks (represented as T1, T2, T3, T4, ... Tn) to the cloud.

- ✧ **Task Validation:**

The Task Manager checks the validity of the tasks. Valid tasks are handed over to the Task Scheduler.

- ✧ **Task Scheduling:**

The Task Scheduler maps the valid tasks to VMs (represented as VM1, VM2, VM3, VM4, ... VMm) on physical hosts housed in data centers.

- ✧ **Data Collection:**

IoT devices in the Things Layer continuously monitor the patient's health status and generate healthcare data.

- ✧ **Data Transmission:**
- ✧ The Data Collector Layer collects the healthcare data and transmits it to the cloud.
- ✧ **Data Management:**
- ✧ The Cloud Layer manages the tasks, schedules them efficiently, monitors VM resources, and stores the data permanently for stakeholder access.

This architecture ensures efficient and reliable healthcare data management, leveraging the power of IoT and cloud computing to enhance patient care and streamline medical service delivery. The Things Layer constitutes the base of the IoT cloud-based healthcare infrastructure, which includes biomedical and environmental sensors like temperature sensors, cameras, GPS modules, blood pressure monitors, EEG sensors, and heart rate monitors. These sensors are always on the lookout for collecting and sending healthcare data over Bluetooth or WiFi. The layer plays a key role in facilitating real-time health monitoring by capturing vital physiological information directly from patients [5].

The Data Collector Layer is a go-between between the physical IoT devices and the cloud. It consists of mobile phones, smartwatches, or other smart devices that serve as go-betweens to gather data from sensors. Once aggregated, the data is sent over the internet to the cloud for processing. This provides smooth data exchange from the patient's surroundings to centralized cloud services, facilitating continuous, location-independent health supervision [6].

The Cloud Layer is the computational core of the system, divided into four main modules: Task Manager, Task Scheduler, VM Manager, and Data Storage. The Task Manager accepts and checks for tasks, prioritizes them, and places them in queues accordingly [7]. The Task Scheduler then allocates these tasks among available virtual machines (VMs) based on the Priority and Shortest Task Based Allocation (PSTBA) algorithm. The VM Manager provides maximum use of resources across VMs, while the Data Storage module maintains health records and task results on a permanent basis, making it available to patients, caregivers, and healthcare professionals [8].

The Process Flow is initiated by the submission of tasks from users, and the validation and scheduling are done through the cloud modules. Information from the Things Layer is transmitted via the Data Collector Layer to the cloud for storage [9]. This coordinated process ensures patient information is both processed and saved effectively. Tasks are processed based on their criticality, and data is made available to stakeholders at any point in time, allowing timely decision-making and medical intervention [10].

This cloud-based IoT architecture combines real-time sensing and cloud intelligence to improve healthcare delivery. It supports constant monitoring, effective task scheduling, and secure data storage, facilitating proactive care for patients [11]. Through the combination of IoT and cloud computing, the system delivers a scalable and responsive platform for the management of healthcare

services in today's digital ecosystems [12].

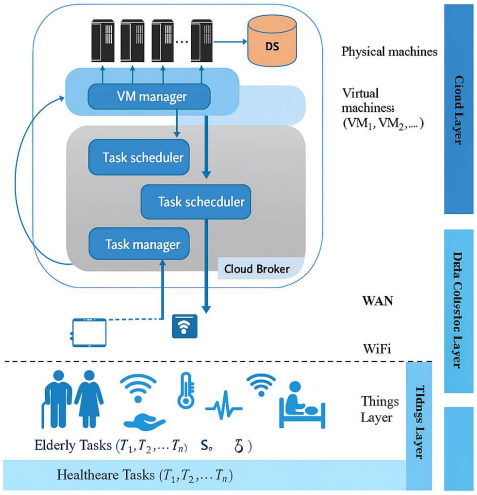

Figure 1: Healthcare Cloud-Based Architecture

The major contribution of the proposed ISSA model and reliability model
summarizes as follows.

1) The task execution on the virtual machine is enhanced using the proposed ISSA, Our focus includes to introduce a neural-inspired Sparrow Search Algorithm for task scheduling and apply the novel model to solve the virtual machine-level scheduling problem.

2) The proposed ISSA focuses on efficient task scheduling by incorporating advanced optimization strategies to improve both the convergence speed and accuracy of the Improve Sparrow Search Algorithm.

3) The detailed design and implementation of the neural-inspired Sparrow Search Algorithm described and compared against existing meta-heuristics technique whale optimization technique, and BAT. Our results and discussions section demonstrates that the hybrid technique ISSA outperforms using resource utilization in a scalable cloud environment.

4) The proposed model focuses on the identification of the sensitive component of cloud infrastructure by estimating the reliability of the cloud and measuring the effect of variation in failure and repair rates. The proposed model improves the performance of the task and cloud infrastructure both at the same time.

The rest of this manuscript is organized as follows. Section 2 covers the related works. Section3 includes the motivations of task scheduling optimization. Section 4 presents the proposed ISSA approach and its implementation details infrastructure. The performance evaluation and analysis are included in Section 6 concludes the work.

2. **Literature Review:** The integration of cloud and edge computing in the context of healthcare infrastructure has picked up pace in the past few years with increasing demand for high-performance, cost-effective, and real-time processing of data. Several studies have indicated new models and algorithms for improving performance, minimizing expenditure, and guaranteeing quality service delivery in the context of healthcare-specific cloud infrastructures. [13] Rahimi et al. (2022) sought to improve the efficiency of healthcare diagnostic cloud computing systems. Their paper presented an optimized Particle Swarm Optimization (PSO) method coupled with discrete wavelet transform (DWT) and artificial neural network (ANN) optimization for diagnosing neurological disorders through the use of EEG data. The hybrid model greatly enhanced diagnosis accuracy and execution time compared to the common Genetic Algorithm (GA) methods. Though, the model requires very accurate EEG signals and has intricate integration involved, this makes the model less scalable and flexible when used in resource-limited healthcare environments.

[14] Kanani et al. (2024) emphasized real-time processing of healthcare data using Fog computing. The authors suggested the application of OptiFog algorithm coupled with dynamic resource handling on Raspberry Pi clusters. The configuration assisted in latency minimization and response time for IoT-based healthcare systems. The major benefit was the efficient handling of healthcare information near the point of origin, thereby reducing network congestion. However, its real-world implementation is restricted because it relies on a particular hardware platform and SMS response tracking mechanism.

[15] Dhruva et al. (2024) designed a decision support framework to help in the right selection of cloud providers for healthcare applications. Their hybrid approach used Fermatean fuzzy sets, weighting using LOPCOW, and ranking using the CoCoSo algorithm. This coupling managed uncertainty and multiple conflicting criteria, minimizing the need for user intervention. Although the intelligent system enhanced decision-making effectiveness, its generality is doubtful owing to differences in regions and infrastructure, as well as the complexity of the multi-criteria decision-making process.

[16] Yuan et al. (2024) tackled the issue of excessive operating expenses in hybrid edge-cloud systems through a hybrid meta-heuristic algorithm that integrated simulated annealing

and genetic algorithms. The strategy effectively handled task allocation and resource provisioning while reporting reduced total system expense and improved convergence speeds. The complexity of implementation and scalability challenges of adapting the solution to larger healthcare systems, however, rendered it with significant limitations to extensive applicability.

[17] Aldailamy et al. (2024) sought to minimize the cost of data object replication and placement in multi-cloud settings, specifically in the domain of online social networks. The authors proposed Dynamic Fixed Time (DFT) and Dynamic Exponential Time (DET) algorithms adaptively replicating data on the basis of object popularity. The algorithms saved substantial cost and enhanced data distribution efficacy. However, the problem is to precisely forecast data object popularity, which is a major factor in making dynamic replication techniques efficient.

[18] Zhang et al. (2024) investigated applying machine learning to improve cloud resource scheduling and management. Their method combined deep learning and genetic algorithms for dynamically optimizing the allocation of resources. The combination resulted in enhanced system performance and utilization of resources while handling computational workloads. The use of robust and accurate machine learning models, however, opens up overfitting and low-quality performance risks in noisy or skewed data.

[19] Chen et al. (2024) introduced a new heuristic algorithm to optimize placement of virtual machines (VMs) in cloud data centers with a view to reducing communication and image retrieval costs. The method entailed PM clustering, VM partitioning, and semidefinite programming to compute effective arrangements of VMs. Experiments revealed enhanced performance compared to current methods. The primary disadvantage of this approach is the complexity of heuristic installation, particularly in big-scale or highly dynamic cloud environments where real-time consistent performance is paramount.

[20] Bi et al. (2024) emphasized minimizing computation offloading cost and optimizing user association in hybrid cloud-edge networks. They presented a two-stage optimization model LSAG using Lévy flight with a simulated annealing-based grey wolf optimizer. The approach ensured effective cost minimization and rapid convergence in resource allocation functions. However, implementing this approach under varying user behavior and network dynamics is challenging based on its algorithmic nature.

[21] Rac and Brorsson (2024) addressed the challenge of application price and service quality (QoS) upkeep along the edge-to-cloud continuum. Their designed framework employs adaptive placement of service, continuous monitoring of KPI, and dynamic relocation. The system showed strong adaptability and cost-effectiveness in a 5G-based use case for healthcare. Italic font to indicate reference numbers. Yet, the dependence of the system on accurate and ongoing KPI feedback adds dependency on the quality and detail of monitoring mechanisms, which would slow down performance if not correctly calibrated.

[22] Guindani et al. (2024) optimized cloud settings customized to fit the needs of a particular application through Bayesian Optimization (BO) with machine learning approaches. Their approach easily traversed intricate black-box constraints to provide cost-effective and high-performing cloud configurations. The most significant benefit was gaining improved results with fewer failed configurations. Nevertheless, combining Bayesian Optimization with other machine learning models under different conditions is a technical challenge, particularly when scaling to other application domains with special configuration requirements.

Table 1: Related work summary for cloud computing in healthcare

Reference	**Objective**	**Methodology**	**Advantage**	**Limitations**
[13] Rahimi et al. (2022)	Improve efficiency of cloud computing in healthcare services.	Improved PSO approach for sensor-data fusion, discrete wavelet transform, and ANN optimization for diagnosing neurological	Increases diagnostic accuracy and reduces execution times compared to GA.	High complexity and requirement of precise EEG signals.

		disabilities.		
[14] Kanani et al. (2024)	Optimize real-time healthcare data processing using Fog computing.	Use of Fog computing, OptiFog algorithm, and dynamic resource management with Raspberry Pi clusters.	Better response time and optimized resource usage for IoT-based healthcare applications.	Limited to the specific setup of Raspberry Pi clusters and SMS-based response measurement.
[15] Dhruva et al. (2024)	Develop a decision framework for selecting suitable cloud vendors in healthcare.	Combination of Fermatean fuzzy set, LOPCOW method, and CoCoSo ranking algorithm for personalized cloud vendor selection.	Handles uncertainty and conflicting criteria effectively, reducing human intervention in decision-making.	Framework complexity and potential difficulty in generalization across different regions or healthcare setups.
[16] Yuan et al. (2024)	Minimize total system cost in hybrid edge and cloud systems.	Hybrid meta-heuristic algorithm combining simulated annealing and genetic operations for task allocation and resource management.	Lower system cost and faster convergence speed compared to benchmarks.	Complexity in algorithm implementation and potential scalability issues in larger systems.
[17] Aldailamy et al. (2024)	Optimize cost of online social networks using multi-cloud replication and placement algorithms.	Dynamic Fixed Time (DFT) and Dynamic Exponential Time (DET) algorithms for adaptive replication and placement based on data object popularity.	Significant cost savings in data object replication and placement in multi-cloud environments.	Potential challenges in accurately predicting data object popularity and adjusting replication times.
[18] Zhang et al. (2024)	Improve resource scheduling and management in cloud computing using machine learning.	Machine learning optimization techniques, including deep learning and genetic algorithms, for cloud resource allocation.	Enhanced system performance and efficiency in cloud resource management.	Dependence on accurate ML models and potential overfitting issues.
[19] Chen et al. (2024)	Minimize image retrieval and communication cost in VM placement in cloud data centers.	Heuristic-based algorithm with PM clustering, VM partitioning, and semidefinite programming for VM placement.	More effective and efficient VM placement compared to state-of-the-art methods.	Complexity in heuristic implementation and potential limitations in dynamic and large-scale environments.

[20] Bi et al. (2024)	Minimize computation offloading cost and user association in hybrid cloud and edge computing.	Two-stage optimization algorithm (LSAG) combining Lévy flight and simulated annealing-based grey wolf optimizer.	Cost-efficient and faster convergence in computation offloading and resource allocation.	Algorithm complexity and potential difficulty in adapting to varying user demands and network conditions.
[21] Rac & Brorsson (2024)	Lower costs of running applications in edge-to-cloud continuum while maintaining QoS.	Adaptive service placement and scheduling methodology with continuous KPI monitoring and dynamic location adjustments.	Robust cost savings and adaptability to changing environments, demonstrated in a 5G application use case.	Performance dependency on accurate KPI monitoring and adaptability mechanisms.
[22] Guindani et al. (2024)	Find optimal cloud configurations for different applications using Bayesian Optimization and Machine Learning.	Integration of Bayesian Optimization and ML techniques for cloud system configuration and performance optimization.	Outperforms other black-box techniques with fewer unfeasible executions and reduced costs.	Potential challenges in handling black-box constraints and integrating BO with different ML techniques.

3. Motivations and Scenario:

Motivations:

Efficiency in Healthcare Delivery: The increasing demand for healthcare services requires methods that optimize performance, reduce costs, and improve patient outcomes. Efficient execution of tasks, reduced CPU utilization, and streamlined processes can directly benefit healthcare providers and patients.

Reducing Stakeholder Requirements: In healthcare, stakeholders include patients, doctors, nurses, administrators, and insurers. Each stakeholder has different requirements, such as faster diagnosis, accurate records, and efficient resource utilization. Reducing these requirements through innovative methods can make healthcare more accessible and responsive.

Minimizing Execution Time: Time is a critical factor in healthcare. Whether it's diagnosing a patient, processing medical records, or managing healthcare resources, faster execution can lead to better outcomes. The motivation is to minimize delays and ensure timely delivery of services.

Optimizing CPU Utilization: With the rise of digital healthcare, including electronic health records (EHRs) and telemedicine, healthcare systems must efficiently manage computational resources. Optimizing CPU utilization can lower costs, enhance performance, and ensure system reliability, especially during peak usage.

Efficient Management of Patient Digital Records: Storing and managing patient data is a significant challenge in healthcare. Ensuring that digital records are stored securely, accessed efficiently, and managed effectively is essential for compliance, continuity of care, and data-driven decision-making.

Scenario:

Imagine a large healthcare facility that serves thousands of patients daily. The facility has implemented a new digital healthcare management system that automates various processes, such as patient registration, diagnostics, and billing. The sensors monitor vital signs such as heart rate, blood pressure, and other health metrics. Sensors that track environmental factors like humidity and temperature, which are crucial for patient care, especially in remote monitoring scenarios. Devices in smart homes and hospitals that provide real-time data on patient conditions and emergency situations.

The data from the sensors and connected devices is collected and stored in a centralized dataset. This data is crucial for processing and analysis in the cloud. The system faces challenges such as high CPU usage, long execution times for processing medical records, and meeting the diverse requirements of stakeholders like patients, medical staff, and administrators. The research aims to address these challenges by developing methods that optimize the system's performance. The proposed solution helps to process tasks faster,

reducing the overall execution time, and can minimize CPU utilization, ensuring that the system runs efficiently without overloading the hardware.

Figure 2: Scenario healthcare in Cloud-computing domain

4. Proposed Approach:

Dataset Collection:

The dataset utilized in this research is composed of two key segments: Cloud Task Data and Healthcare Cloud Optimization Data. The data-set is collected from Kaggle website named healthcare cloud optimization dataset and cloud-task-data on https://www.kaggle.com/datasets/rakeshjalota/healthcare-cloud-optimization-dataset . The sample data records are shown in figure 3

These datasets are designed to capture various attributes necessary for optimizing healthcare cloud services.

a) Cloud Task Data:

This dataset focuses on the tasks that are scheduled in a cloud computing environment. It includes the following attributes:

Task ID: A unique identifier for each task.

Task Length: The length of the task, typically measured in time or computation units.

Priority: A value indicating the priority level of the task, with higher numbers often indicating higher importance or urgency.

This dataset is instrumental in simulating real-world cloud workloads, where tasks with varying lengths and priorities must be managed efficiently.

b) Healthcare Cloud Optimization Data:

This dataset is specific to the healthcare domain, where tasks related to patient data processing are optimized. The attributes include:

Task ID: A unique identifier for each healthcare-related task.

Task Length: The duration or complexity of the task, typically measured in computation time.

Input Size (I): The size of the input data that needs to be processed, measured in bytes.

Output Size (O): The size of the data generated after processing, also measured in bytes.

PE (Processing Element): The number of processing elements required to execute the task.

Start Time (Patient ID): The start time of the task or a reference to the specific patient ID linked to the task.

Power Consumption: The power consumed by the task during execution, reflecting the energy efficiency of the process.

SLA Compliance: A boolean value indicating whether the task complies with the Service Level Agreement (SLA), which often includes constraints on time, cost, and resource usage.

Task ID	Task Len
1	37
2	46
3	45
4	45
5	26

a) Cloud Task Data

Task ID	Task Length	Input Size (I	Output Size	PE
1	37487	446	420	
2	18490	488	414	
3	9797	390	294	
4	22258	324	409	
5	13985	355	286	

b) healthcare cloud optimization data

Figure 3: Sample data a) Cloud Task Data and b) healthcare cloud optimization data

The proposed model aims to enhance the efficiency of healthcare services by leveraging cloud computing and optimization techniques. This model integrates various data sources, cloud infrastructure, and optimization algorithms to deliver cost-effective and high-quality healthcare services.

Data Processing and Analysis:

Once the data is collected, it undergoes processing and analysis to extract meaningful insights. This step involves filtering, organizing, and preparing the data for further actions. The processed data is then analyzed to detect anomalies, trends, and other significant patterns that can inform healthcare decisions.

Cloud Infrastructure:

The cloud acts as the central hub where all data is processed, stored, and managed. It provides the computational resources needed to handle the large volume of data generated by the sensors and connected devices. The cloud infrastructure ensures scalability, reliability, and availability of healthcare services, even in remote locations.

Cost Optimization Technique (Improved SSA - I-SSA):

The Improved Sparrow Search Algorithm (I-SSA) is employed to optimize the cost of cloud

resources. This algorithm helps in the efficient allocation of resources while minimizing costs and ensuring Service Level Agreement (SLA) compliance.

The I-SSA dynamically adjusts the allocation of resources based on the real-time demands of the healthcare system, ensuring that the cloud infrastructure operates at optimal cost levels. It is a swarm intelligence optimization technique inspired by the foraging and anti-predation behaviors of sparrows. The population in SSA is typically divided into three roles: producers, scroungers, and vigilant sparrows.

1. Role Classification in SSA

Producers: These individuals lead the search for food, representing solutions with higher fitness. They explore a broader space and guide the swarm toward optimal regions.

Scroungers: These sparrows follow producers to obtain resources or independently explore new areas. Their role is to refine the search and maintain diversity.

Vigilant Sparrows (Reconnaissance Agents): They monitor the environment for threats (e.g., predators in the metaphor), adjusting the swarm's strategy to avoid poor solutions.

2. Position Update for Producers

Producers update their position based on environmental safety levels. If no predator threat is detected (R2 < ST), they perform extensive exploration. However, in unsafe conditions (R2 ≥ ST), they switch to a more conservative strategy using Gaussian-distributed random steps. The formula is:

$$X_{i,j}^{t+1} = \begin{cases} X_{i,j}^{t} \cdot e^{-i/\text{iter}_{\max}}, & \text{if } R_2 < ST \\ X_{i,j}^{t} + Q \cdot L, & \text{if } R_2 \geq ST \end{cases} \quad (1)$$

Where X_{i,j}^t is the position of the ith sparrow in the jth dimension at time t, and Q, L, ST, and R2 are parameters controlling the search behavior.

3. Position Update for Scroungers

Scroungers update their positions based on their index. If their index i > n/2, it means they're more hungry and independently search for food. If i ≤ n/2, they stay near the best producer's location:

$$X_{i,j}^{t+1} = \begin{cases} Q \cdot e^{(X_{\text{worst}} - X_{i,j}^{t})/i^2}, & \text{if } i > \frac{n}{2} \\ X_{i,j}^{t+1} = X_p^{t+1} + |X_{i,j}^{t} - X_p^{t+1}| \cdot A \cdot L, & \text{if } i \leq \frac{n}{2} \end{cases} \quad (2)$$

4. Vigilance Mechanism

When a sparrow becomes aware of danger (e.g., a predator threat), it reacts by moving toward the globally best solution or away from the worst one depending on its own fitness:

$$X_{i,j}^{t+1} = \begin{cases} X_{\text{best}}^{t} + \beta \cdot |X_{i,j}^{t} - X_{\text{best}}^{t}|, & \text{if } f_i > f_g \\ X_{i,j}^{t} + K \cdot (|X_{i,j}^{t} - X_{\text{worst}}^{t}|/(f_i - f_w + \varepsilon)), & \text{if } f_i = f_g \end{cases} \quad (3)$$

Where f_i is the fitness of the ith sparrow, f_g and f_w are global best and worst fitness values, and ε is a small constant to prevent division by zero.

Improvements to the Sparrow Search Algorithm

Although the original SSA performs well, it has limitations in population diversity, convergence speed, and getting trapped in local optima. Therefore, two main improvements are proposed:

1. Optimized Initial Population via Improved Sine Chaotic Map

update equation, making it responsive to iteration progress.

3. Updated Producer Position Formula

With the adaptive weight integrated, the producer's position is updated as follows:

$$X_{i,j}^{t+1} = \begin{cases} X_{i,j}^{t} \cdot f_{ib}^{t} \cdot w(t), & \text{if } R_2 < ST \\ X_{i,j}^{t} + Q \cdot L, & \text{if } R_2 \geq ST \end{cases} \quad (6)$$

Here, f_{ib}^t is the optimal fitness in the current iteration, and w(t) adjusts the exploration intensity.

These enhancements significantly improve SSA's convergence speed, global search capability, and solution accuracy, making it more robust for applications such as optimizing support vector machines (SVMs) in machine learning or other complex optimization problems.

$$\begin{aligned} a_{i+1} &= \sin(\pi \cdot a_i) \\ b_{i+1} &= \sin(\pi \cdot b_i) \\ y_{i+1} &= \text{mod}(a_{i+1} \cdot b_{i+1}, 1) \end{aligned} \quad (4)$$

This enhanced initialization increases diversity and avoids premature convergence.

2. Adaptive Dynamic Weight Strategy

To better balance exploration and exploitation, a dynamic adaptive weight is introduced into the producer's position update. This weight adjusts over iterations to encourage wide exploration early and focused exploitation later:

$$w(t) = \frac{\exp(2 \cdot (1 - \sin(t/\text{iter}_{\max})))}{\exp(2 \cdot (1 - \cos(t/\text{iter}_{\max}))} \quad (5)$$

This w(t) modifies the original producer

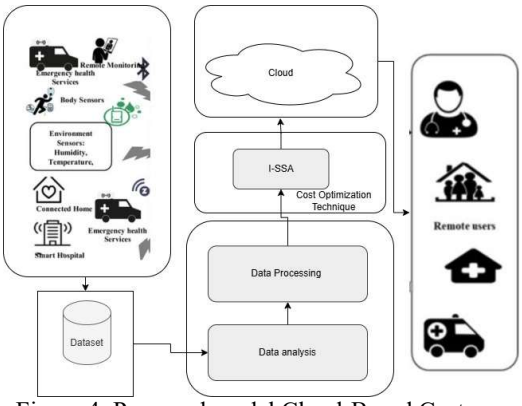

Figure 4: Proposed model Cloud-Based Cost Optimization in Health Care Domain

Pseudo Code: Improved Sparrow Search Algorithm:

Function Improved_Sparrow_Search_Algorithm(num_iterations, num_vms, num_hosts, task_length)
Step 1: Initialize VM capabilities and host assignments Initialize cpu_capability, memory_capability, host_assignment for num_vms and num_hosts
Step 2: Initialize positions randomly for each VM positions ← Randomly assign each VM to a host best_position ← positions[0]
Step 3: Calculate initial fitness for the best position best_fitness ← Calculate_Fitness(cpu_capability[best_position], memory_capability[best_position], task_length)
Step 4: Start optimization loop For iteration in range(0 to num_iterations):
Step 4.1: Evaluate fitness for all VMs For i in range(0 to num_vms): current_fitness ← Calculate_Fitness(cpu_capability[positions[i]], memory_capability[positions[i]], task_length)
Step 4.2: Update the best position if better fitness is found If current_fitness > best_fitness: best_fitness ← current_fitness best_position ← positions[i]
Step 4.3: Calculate mean and standard deviation of current positions mean_position ← Calculate_Mean(positions) std_position ← Calculate_Standard_Deviation(positions)
Step 4.4: Update positions (exploration and exploitation) For i in range(0 to num_vms): If Random_Number > 0.5: # Exploitation: Move based on current position with some random adjustment positions[i] ← positions[i] + Random(-1, 1) * std_position Else: # Exploration: Move based on the mean position with random adjustment positions[i] ← mean_position + Random(-1, 1) * std_position
Step 4.5: Ensure positions are within the valid range positions[i] ← Clip(positions[i], 0, num_hosts - 1)
Step 5: Return the best position and the total time taken Return best_position, time_taken End Function

Improved Sparrow search:

The improvements made in the Improved Sparrow Search Algorithm (Improved SSA) over the Original Sparrow Search Algorithm (SSA) can be summarized as follows:

Neighborhood Search Strategy:

Original SSA: In the original SSA, the positions of the sparrows are updated based on a global random strategy influenced by the standard deviation and mean of the current positions. This approach can sometimes lead to suboptimal exploration and exploitation.

Improved SSA: The improved SSA introduces a focused neighborhood search. For each iteration, the current position of the sparrow has its neighbors evaluated. The algorithm then moves to the best neighbor, enhancing the local search capability and potentially leading to faster convergence to optimal or near-optimal solutions.

Fitness Calculation and Selection:

Original SSA: In the original SSA, the fitness of each position is calculated, and the best position is updated if a better fitness is found. This is done for each sparrow in each iteration.

Improved SSA: The improved SSA still calculates fitness in each iteration but focuses on the current position and its immediate neighbors. This allows the algorithm to maintain a balance between exploration (searching new areas) and exploitation (refining the current solution).

Initial Position:

Original SSA: The positions of the sparrows are initialized randomly across the search space.

Improved SSA: The initial position of the sparrow is set to the middle of the virtual machines (num_vms // 2). This initialization can potentially give a better starting point, reducing the randomness in initial searches.

Simplified Position Update:

Original SSA: The position updates in the original SSA involve a combination of random walks and adjustments based on the overall position statistics (mean and standard deviation).

Improved SSA: The improved SSA simplifies the position update by focusing on immediate neighbors. This change makes the search more directed and can reduce the number of iterations needed to find a good solution.

Service Delivery to End Users:

The processed and analyzed data, after undergoing cost optimization, is used to deliver healthcare services to remote users, including patients at home, emergency services, and healthcare professionals. The optimized system ensures that healthcare services are delivered in a timely, efficient, and cost-effective manner.

5. Result Analysis: The figure 5 illustrates the fitness percentages of Virtual Machines (VMs) in a cloud-based architecture. The architecture aims to optimize the allocation of computational resources in the cloud to process the vast amount of data generated by health care devices. The plot shows the fitness percentages of 100 VMs, which indicates how efficiently each VM is performing its assigned tasks. The fitness percentage is a measure of the VM's ability to handle tasks based on factors such as CPU utilization, memory usage, and power consumption. The graph shows a significant variation in fitness percentages across the VMs. Some VMs perform exceptionally well, reaching fitness levels of over 95%, while others drop to around 55%.

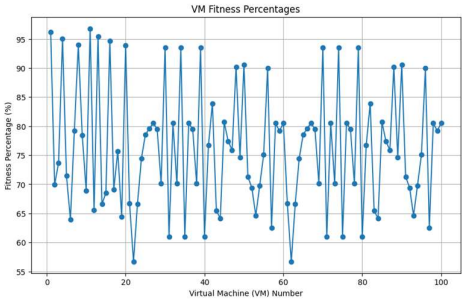

Figure 5: Fitness percentages of Virtual Machines (VMs)

The provided figure 6 compares the movement and fitness values of several optimization algorithms Sparrow Search Algorithm (SSA), Whale Optimization Algorithm (WOA), BAT Algorithm for choosing virtual machines (VMs) in a cloud-based architecture. The figure shows 15 VMs (VM0–VM14) with varying fitness values ranging from 63.9 to 96.8. The majority of the VMs have fitness values above 70, indicating an efficient utilization of resources, with some outliers having lower fitness values. Original SSA Chosen VMs are Highlighted with green dots. SSA Starting Point with Purple dots show the original SSA starting points, and the blue lines track their path toward selecting better VMs. Improved SSA (orange dots) shows more optimized movement as it tries to improve the fitness values of selected VMs, resulting in higher fitness convergence than the original SSA. Represented by the red lines and cyan dots, WOA also selects VMs based on their fitness. The algorithm starts from a lower fitness value and converges toward a higher fitness over time. BAT's optimization (yellow dots and black lines) shows fewer movements compared to SSA and WOA but focuses on a different selection of VMs with slightly higher fitness. ISSA shows a more aggressive path movement, where the algorithm explores the VMs more extensively and quickly converges toward the highest fitness VMs (VM0, VM10, VM3).ISSA seems to have the fastest convergence, especially after applying improvements, as shown by the movement of the orange dots. BAT focuses on fewer VM transitions but maintains comparable performance, suggesting it might be more suited for scenarios where fewer iterations are preferred.

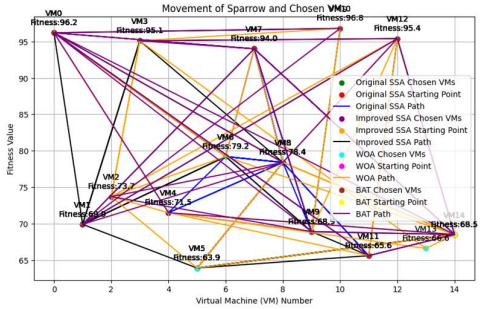

Figure 6: The movement and fitness values of several optimization algorithms

The figure 7 compares the fitness percentages of VMs chosen for each chunk across four different algorithms: Improved SSA (blue), Original SSA (orange), WOA (green), and BAT (red). The fitness percentage is a measure of how well the algorithm performs in selecting the best VM for each chunk, with higher percentages indicating better performance. The Improved SSA generally has better performance across most chunks compared to the Original SSA.

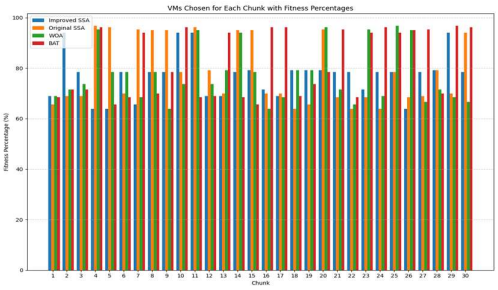

Figure 7: fitness percentages of VMs chosen for each chunk

The figure 8 compares the time taken (in seconds) for each chunk iteration by four different algorithms: Improved SSA (blue), Original SSA (orange), WOA (green), and BAT (red). The x-axis represents the chunk, while the y-axis shows the time taken to process each chunk. Across most chunks, all algorithms maintain relatively high fitness percentages. This indicates that all algorithms are effective at VM selection but show differences in optimization power. Its time increases moderately from approximately 0.5 seconds to just under 3 seconds, indicating better optimization and reduced computational complexity. The Original SSA, WOA, and BAT exhibit similar performance trends, with execution time increasing almost linearly with the number of chunks. These algorithms reach up to 16 seconds by the final chunk, with the BAT algorithm consistently being the slowest among them.

.

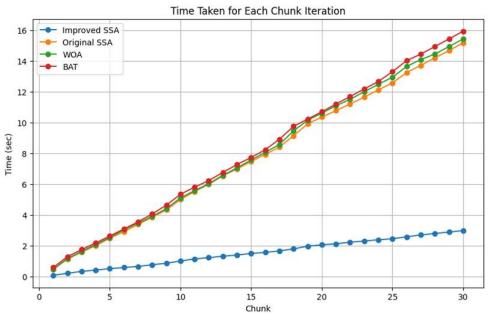

Figure 8: Compares the time taken (in seconds) for each chunk iteration

$$\text{Percentage Improvement} = \left(\frac{\text{Time}_{\text{other}} - \text{Time}_{\text{Improved SSA}}}{\text{Time}_{\text{other}}} \right) X$$

(7)

These results clearly demonstrate the enhanced capability of the Improved SSA in reducing execution time, thereby enabling more efficient task scheduling in cloud computing environments. The significant improvements confirm the effectiveness of the proposed modifications, including the dynamic adaptive weight and improved initialization strategy.

Over Original SSA:

$$\left(\frac{15.5 - 3}{15.5} \right) \times 100 \approx 80.65\%$$

Over WOA:

$$\left(\frac{15.7 - 3}{15.7} \right) \times 100 \approx 80.89\%$$

Over BAT:

$$\left(\frac{16 - 3}{16} \right) \times 100 = 81.25\%$$

The figure 9 shows how the average finish time (in milliseconds) varies as the number of tasks increases from 2000 to 10,000. Improved SSA (blue) consistently shows the lowest average finish time across all task sizes. The finish time starts at around 615 ms for 2000 tasks and drops below 610 ms as the number of tasks increases. Improved SSA is the clear leader in both average finish time and execution time. So, ISSA offers superior performance across all task sizes.

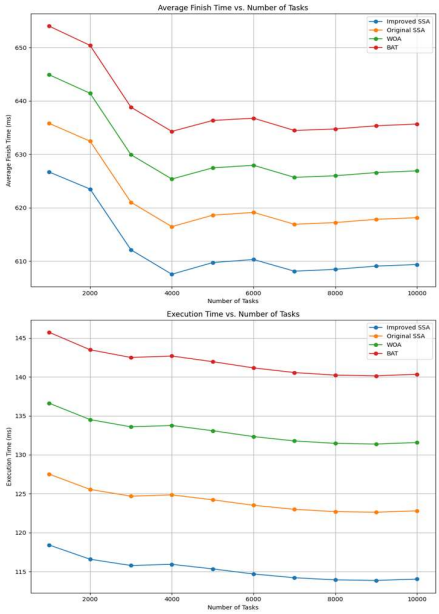

Figure 9: IoT Cloud-Based Architecture

6. Conclusion: The proposed ISSA leads in optimization with quicker convergence toward high-fitness VMs. WOA provides a balanced approach for convergence. it achieves competitive fitness values. BAT offers a more conservative strategy focus on high-efficiency moves with fewer iterations. We an conclude that BAT is suitable for scenarios where fewer resources or iterations are preferred. The Improved SSA algorithm appears to have the most consistent and high fitness percentage across the chunks. This indicate that it is most effective method for VM selection. The Improved SSA algorithm outperforms all other algorithms significantly in terms of time efficiency. Its runtime is substantially lower

and scales much better as the number of chunks increases. The Original SSA, WOA, and BAT algorithms take significantly more time per chunk iteration. The BAT algorithm shows slightly slower performance than the other two toward the higher chunk numbers. Improved SSA shows strong fitness performance but also achieves far superior time efficiency. This makes it an attractive choice for scenarios where both performance and processing time are critical. The improved SSA algorithm outperforms all other algorithms significantly in terms of time efficiency. Its runtime is substantially lower and scales much better as the number of chunks increases. The Original SSA, WOA, and BAT algorithms take significantly more time per chunk iteration. Improved SSA delivers the best performance in terms of average finish time. BAT struggles to match the efficiency of the other algorithms. Improved SSA once again stands out for its efficiency in execution time. Its performance improves as the number of tasks increases. which makes it ideal for high workloads. In future, the proposed approach can be implemented in other cloud computing application.

Reference :

[1] Acharya, B., Panda, S., Das, S., Majhi, S. K., Gerogiannis, V. C., & Kanavos, A. (2025). Optimizing task scheduling in cloud environments: a hybrid golden search whale optimization algorithm approach. Neural Computing and Applications, 1-23.

[2] Yanamala, A. K. Y. (2024). Emerging Challenges in Cloud Computing Security: A Comprehensive Review. International Journal of Advanced Engineering Technologies and Innovations, 1(4), 448-479.

[3] Mistry, H. K., Mavani, C., Goswami, A., & Patel, R. (2024). The Impact Of Cloud Computing And Ai On Industry Dynamics And Competition. Educational Administration: Theory and Practice, 30(7), 797-804.

[4] Ogundipe, D. O. (2024). Conceptualizing cloud computing in financial services: opportunities and challenges in Africa-US contexts. Computer Science & IT Research Journal, 5(4), 757-767.

[5] Yanamala, A. K. Y. (2024). Optimizing Data Storage in Cloud Computing: Techniques and Best Practices. International Journal of Advanced Engineering Technologies and Innovations, 1(3), 476-513.

[6] Kanungo, S. (2024). AI-driven resource management strategies for cloud computing systems, services, and applications. World Journal of Advanced Engineering Technology and Sciences, 11(2), 559-566.

[7] Putzier, M., Khakzad, T., Dreischarf, M., Thun, S., Trautwein, F., & Taheri, N. (2024). Implementation of cloud computing in the German healthcare system. NPJ Digital Medicine, 7(1), 12.

[8] Bell, C. (2024). Cloud computing. In MicroPython for the Internet of Things: A Beginner's Guide to Programming with Python on Microcontrollers (pp. 413-424). Berkeley, CA: Apress.

[9] Yalamati, S. (2024). Data Privacy, Compliance, and Security in Cloud Computing for Finance. In Practical Applications of Data Processing, Algorithms, and Modeling (pp. 127-144). IGI Global.

[10] Gammelgaard, B., & Nowicka, K. (2024). Next generation supply chain management: the impact of cloud computing. Journal of Enterprise Information Management, 37(4), 1140-1160.

[11] Khan, H. U., Ali, F., & Nazir, S. (2024). Systematic analysis of software development in cloud computing perceptions. Journal of Software: Evolution and Process, 36(2), e2485.

[12] Wang, Y., Bao, Q., Wang, J., Su, G., & Xu, X. (2024). Cloud computing for large-scale resource computation and storage in machine learning. Journal of Theory and Practice of Engineering Science, 4(03), 163-171.

[13] Rahimi, M., Navimipour, N. J., Hosseinzadeh, M., Moattar, M. H., & Darwesh, A. (2022). Cloud healthcare services: A comprehensive and systematic literature review. Transactions on Emerging Telecommunications Technologies, 33(7), e4473.

[14] Kanani, P., Vasoya, A., Shah, K., Kothari, N., Patil, N., Pandya, G., & Padole, M. (2024). Optimization of Health-as-a-Service Using OptiFog Algorithm. International Journal of Intelligent Engineering & Systems, 17(1).

[15] Dhruva, S., Krishankumar, R., Zavadskas, E. K., Ravichandran, K. S., & Gandomi, A. H. (2024). Selection of suitable cloud vendors for health centre: a personalized decision framework with fermatean fuzzy set, LOPCOW, and CoCoSo. Informatica, 35(1), 65-98.

[16] Yuan, H., Bi, J., Wang, Z., Yang, J., & Zhang, J. (2024). Partial and cost-minimized computation offloading in hybrid edge and cloud systems. Expert Systems with Applications, 250, 123896.

[17] Aldailamy, A. Y., Muhammed, A., Latip, R., Hamid, N. A. W. A., & Ismail, W. (2024). Online dynamic replication and placement algorithms for cost optimization of online

social networks in two-tier multi-cloud. Journal of Network and Computer Applications, 224, 103827.
[18] Zhang, Y., Liu, B., Gong, Y., Huang, J., Xu, J., & Wan, W. (2024). Application of Machine Learning Optimization in Cloud Computing Resource Scheduling and Management. arXiv preprint arXiv:2402.17216.
[19] Chen, X., Gu, C., Gao, X., Shen, Y., Sun, Z., & Huang, H. (2024). Virtual Machine Placement for Minimizing Image Retrieval Cost and Communication Cost in Cloud Data Center. IEEE Transactions on Network and Service Management.
[20] Bi, J., Wang, Z., Yuan, H., Zhang, J., & Zhou, M. (2024). Cost-Minimized Computation Offloading and User Association in Hybrid Cloud and Edge Computing. IEEE Internet of Things Journal.
[21] Rac, S., & Brorsson, M. (2024). Cost-aware service placement and scheduling in the Edge-Cloud Continuum. ACM Transactions on Architecture and Code Optimization.
[22] Guindani, B., Ardagna, D., Guglielmi, A., Rocco, R., & Palermo, G. (2024). Integrating Bayesian Optimization and Machine Learning for the Optimal Configuration of Cloud Systems. IEEE Transactions on Cloud Computing.