Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.051634

ARTICLE

Fitness Sharing Chaotic Particle Swarm Optimization (FSCPSO): A Metaheuristic Approach for Allocating Dynamic Virtual Machine (VM) in Fog Computing Architecture

Prasanna Kumar Kannughatta Ranganna[1], **Siddesh Gaddadevara Matt**[2], **Chin-Ling Chen**[3,4,*], **Ananda Babu Jayachandra**[5] and **Yong-Yuan Deng**[4,*]

[1]Department of Computer Science & Engineering, Ramaiah Institute of Technology, Bangalore, 560054, India

[2]Department of Artificial Intelligence & Data Science, Ramaiah Institute of Technology, Bangalore, 560054, India

[3]School of Information Engineering, Changchun Sci-Tech University, Changchun, 130600, China

[4]Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taichung, 41349, Taiwan

[5]Department of Information Science and Engineering, Malnad College of Engineering, Hassan, 573202, India

*Corresponding Authors: Chin-Ling Chen. Email: clc@cyut.edu.tw; Yong-Yuan Deng. Email: allendeng@cyut.edu.tw

Received: 11 March 2024 Accepted: 08 July 2024 Published: 15 August 2024

ABSTRACT

In recent decades, fog computing has played a vital role in executing parallel computational tasks, specifically, scientific workflow tasks. In cloud data centers, fog computing takes more time to run workflow applications. Therefore, it is essential to develop effective models for Virtual Machine (VM) allocation and task scheduling in fog computing environments. Effective task scheduling, VM migration, and allocation, altogether optimize the use of computational resources across different fog nodes. This process ensures that the tasks are executed with minimal energy consumption, which reduces the chances of resource bottlenecks. In this manuscript, the proposed framework comprises two phases: (i) effective task scheduling using a fractional selectivity approach and (ii) VM allocation by proposing an algorithm by the name of Fitness Sharing Chaotic Particle Swarm Optimization (FSCPSO). The proposed FSCPSO algorithm integrates the concepts of chaos theory and fitness sharing that effectively balance both global exploration and local exploitation. This balance enables the use of a wide range of solutions that leads to minimal total cost and makespan, in comparison to other traditional optimization algorithms. The FSCPSO algorithm's performance is analyzed using six evaluation measures namely, Load Balancing Level (LBL), Average Resource Utilization (ARU), total cost, makespan, energy consumption, and response time. In relation to the conventional optimization algorithms, the FSCPSO algorithm achieves a higher LBL of 39.12%, ARU of 58.15%, a minimal total cost of 1175, and a makespan of 85.87 ms, particularly when evaluated for 50 tasks.

KEYWORDS

Fog computing; fractional selectivity approach; particle swarm optimization algorithm; task scheduling; virtual machine allocation

1 Introduction

Fog computing is a distributed computing environment that brings computing resources closer to the location and the user where the data is generated [1]. This enables real-time processing and reduces latency in applications like autonomous systems, augmented reality, and Internet of Things (IoT) [2,3]. Fog computing distributes computing resources across different fog nodes, addresses privacy and security problems, supports scalability, and optimizes resource efficiency [4]. In a fog computing environment, fog nodes are placed between the cloud and end devices. Consequently, it reduces delay and network congestion in the computing environment by managing data streams and user requests in real time [5,6]. The fog computing environment has three important features, namely, good bandwidth usage, the capability of serving real-time applications, and lower latency; however, the fog nodes are dispersed and heterogeneous [7,8]. In order to overcome this problem, efficient task scheduling and Virtual Machine (VM) allocation algorithms are required in fog computing paradigms [9,10].

In this computing environment, task scheduling plays a vital role which ensures that the tasks are adaptively prioritized and executed expeditiously [11]. Additionally, task scheduling is crucial for managing Quality of Service (QoS) parameters like response time, energy consumption, etc., [12,13]. In distributed and multi-processing environments such as the fog computing environment, task scheduling is a multi-constraint and multi-objective problem [14]. On the other hand, fog computing environments often experience variable and dynamic workloads. VM migration and allocation allow for the reallocation and dynamic allocation of computing resources on the basis of the changing demands, ensuring optimal performance [15]. The meta-heuristic methods or algorithms handle a variety of restrictions and produce the best possible outcomes. In a majority of the existing research studies, several metaheuristic-based optimization algorithms such as Genetic Algorithm, Ant Colony Optimization, Firefly Optimization, Grey Wolf Optimization, and Particle Swarm Optimization have been utilized for task scheduling and VM allocation in fog computing environments. The conventional metaheuristic-based optimization algorithms are effective, but get trapped into local optima problems with poor convergence rates [16,17]. When compared to cloud computing, fog nodes offer more diverse and capacity-constrained computation as well as increased storage capabilities. One of the most difficult tasks for researchers is to coordinate these resources. Also, many researchers have developed and employed a number of unique metaheuristic strategies so far, and it is noted that emergent metaheuristic techniques are still in their development and have not received much attention in addressing the challenge of fog computing resource allocation [18,19]. In order to handle this problem and achieve better task scheduling and VM allocation performance, a novel framework, which is a combination of fractional selectivity approach and Fitness Sharing Chaotic Particle Swarm Optimization (FSCPSO) algorithm, is proposed in this manuscript.

The major contributions are represented as follows:

- In this research, a fractional selectivity approach is used for task scheduling in fog computing environments. This fractional selectivity approach adaptively assigns a portion of incoming data or tasks to each fog node based on the characteristics and requirements. The dynamic optimization of resources by a fractional selectivity approach ensures better processing power and storage capacity for fog nodes. Furthermore, the effective scheduling of tasks by a fractional selectivity approach reduces the response time and total cost in fog computing environments.
- An FSCPSO algorithm is proposed for the effective allocation of VMs in fog systems. The fitness sharing provides diverse solutions that overcome local optima problems, and its chaotic nature improves the exploration ability of the optimization algorithm, which speeds up the

convergence process. This particularly happens in a dynamic or complex fog computing environment. The proposed FSCPSO algorithm provides optimized resource usage and improves the convergence speed for VM migration in the fog system.

- The proposed framework's (combination of fractional selectivity approach and FSCPSO algorithm) performance is investigated using six evaluation measures that are Load Balancing Level (LBL), Average Resource Utilization (ARU), total cost, makespan, energy consumption, and response time.

The rest of the manuscript is prepared as follows. The recent articles on the topic of fog computing are surveyed in Section 2. The description of the proposed methodologies, numerical investigation, and the conclusion with future extension are stated in Sections 3–5, respectively.

2 Literature Survey

Task scheduling is one of the primary concerns in fog systems which directly influences latency time, task computation, and resource utilization. The objective of scheduling optimization is to ensure VM allocation and optimize operational efficiency. In recent times, numerous metaheuristic and heuristic optimization algorithms have been developed for resolving scheduling issues. A few of them are discussed below:

Yadav et al. [20] introduced a hybrid framework consisting of differential evolution and opposition-based learning methods to solve scheduling concerns. Specifically, an opposition-based learning method was employed to generate a diversified population set, and the differential evolution method was applied to avoid the local optima problems. This hybrid framework not only improved the resource utilization but also minimized the cost and makespan. Related to conventional methods, this hybrid method obtained impressive performance on different types of workloads.

Talaat et al. [21] developed an efficient load-balancing framework in fog systems by integrating a modified Particle Swarm Optimization (PSO) algorithm with the Convolutional Neural Network (CNN). This framework comprised three modules known as (i) Optimized Dynamic Scheduler (ODS), (ii) CNN-Based Classifier (CBC), and (iii) Fog Resource Monitor (FRM). In this framework, ODS was utilized to allocate the incoming processes to appropriate servers, CBC was responsible for classifying fog servers as appropriate or inappropriate, and FRM was used to monitor the resources of every server. As per the analysis of the empirical outcomes, this framework achieved superiorly higher resource utilization and reduced response time. Generally, the optimization algorithms completely rely on historical data or certain assumptions in order to make decisions. It is hard to make future workload predictions in fog computing environments, because of the dynamic nature.

Yakubu et al. [22] presented a modified version of the Harris Hawks Optimization (HHO) algorithm for assigning the best resources to a task, and the task was distributed between the cloud and fog systems. The aim of this literature was to improve resource utilization and to diminish power consumption, execution cost and makespan time in the cloud and fog layer. In comparison to the conventional algorithms, the presented algorithm effectively balanced the load across different tasks. Vispute et al. [23] as well as Baburao et al. [24] utilized the PSO algorithm for effective scheduling of tasks in fog computing systems. In addition to this, Subbaraj et al. [25] presented a Crow Search Algorithm (CSA) for resource scheduling and allocation in fog systems. In this literature, the security hit ratio and success ratio were considered objective functions, while a local search approach was utilized to improve the performance of the CSA. However, these implemented HHO, CSA and PSO algorithms proved to be ineffective for the fog systems when diverse fog nodes were considered, which in turn led to imbalance in resource usage.

Li et al. [26] employed the Beetle Antennae Search (BAS) algorithm and an enhanced Contract Net Protocol (CNP) for task offloading in fog systems. Initially, the CNP was applied to obtain the information in fog nodes, and the agents of the BAS algorithm were allocated based on the obtained information. The simulation outcomes and empirical analysis confirmed the superiority of this presented framework in task offloading. Furthermore, Kishor et al. [27] developed a smart Ant Colony Optimization (ACO) algorithm for task offloading in fog environments, especially for IoT-sensor applications. Initially, the fog environments comprised numerous edge devices, but the BAS and ACO optimization algorithms faced challenges in managing large-scale optimization problems because of increased computational complexity.

Abbasi et al. [28] introduced a multi-objective Genetic Algorithm (GA) for improving the trade-off between delay and energy consumption in fog systems by effectively processing workloads. The extensive empirical investigation revealed that this optimization algorithm decreased almost 25% of delay and energy consumption in comparison to the existing algorithms, but the issue of lower convergence speed was a concern. Furthermore, Hajam et al. [29] presented the Spider Monkey Optimization (SMO) algorithm for task scheduling and resource allocation in fog systems. This SMO algorithm efficiently selected optimal fog nodes, reducing the total cost by means of monetary cost and service time. However, the conventional SMO algorithm had high communication overhead because task-offloading decisions often involved communication among cloud servers, fog nodes, and edge devices. Additionally, Ragmani et al. [30] developed a hybrid fuzzy-based ACO algorithm for VM scheduling in cloud environments. This developed hybrid fuzzy-based ACO algorithm selected an appropriate server with an optimum computational time. However, in the context of VM scheduling, defining a suitable objective function was more complex for the hybrid fuzzy-based ACO algorithm.

Kaur et al. [31] employed an Elastic Scheduling Algorithm (ESA) for an effective VM migration and allocation in cloud environments. This presented ESA utilized cosine similarity for enhancing the performance of VM migration and allocation by means of QoS, and this presented ESA was analyzed and analyzed using the parameter of power consumption. However, the ESA involved complex decision-making for allocation and migration of VMs based on varying resource availability and workloads. Sha et al. [32] applied an efficient optimization algorithm by name of the Discrete Bacteria Foraging Optimization (DBFO) algorithm, for cloud-computing technology-based applications to effectively migrate VMs in cloud computing environments. The optimal VM migration offered optimal load balancing, significant resource usage, and reduced cloud cost. The DBFO algorithm's performance was investigated utilizing the Cloud-Sin simulator. The numerical investigation demonstrated that this DBFO algorithm effectively improved the migration count and energy consumption related to other baseline algorithms; however, this DBFO algorithm had non-deterministic polynomial-time hard problems.

Sutar et al. [33] and Dai et al. [34] employed the ACO algorithm and deep reinforcement learning for live VM migration in a cloud-computing environment. The numerical investigation showed that these presented strategies were impressive by means of energy consumption and convergence rate. Still, the live VM migration comprised three problems of network overhead, service disruption, and downtime. Torre et al. [35] utilized a dynamic evolutionary algorithm, Ajmera et al. [36] utilized a clonal selection algorithm, and Gharehpasha et al. [37] utilized a multi-verse optimization algorithm for efficient VM placement in the cloud data center. By investigating this literature, it was understood that the existing algorithms faced compatibility issues while placing VMs in cloud data centers.

From the overall review, it is clear that the majority of the existing models do not consider the important and fundamental metric of QoS. The traditional optimization algorithms such as Genetic Algorithm, Ant Colony Optimization, Firefly Optimization, Grey Wolf Optimization, and Particle Swarm Optimization were unable to solve the challenge of efficient resource allocation in fog computing because of heterogeneity, mobility, uncertainty, and restricted resource availability. The task of choosing an effective resource allocation method for a broad range of applications continues to confound the researchers. In order to overcome these problems, a novel mathematical algorithm named the FSCPSO algorithm is proposed in this manuscript. The FSCPSO is chosen here because the fitness sharing process offers diverse solutions that avoid the local optima problems, and the chaotic nature enhances the optimization algorithm's exploration potential to accelerate the convergence process, as it is a dynamic fog computing environment. Therefore, the proposed FSCPSO technique increases convergence speed and optimizes resource utilization for VM migration in the fog system.

3 Methodology

Numerous low-latency and delay applications are being run closer to the end devices with the support of the fog layer. Nonetheless, the drawbacks of fog computing are the availability of limited resources as well as the heterogeneity and dynamic nature of fog nodes. Fog computing presents a difficult and demanding task of allocating resources in an optimal and efficient manner. The issue of resource allocation in fog computing has given rise to several algorithms that aim to maximize resource consumption while minimizing latency and delay, among other QoS restrictions. Therefore, this research chooses the FSCPSO algorithm to accomplish the goal of effective and efficient resource allocation. In fog computing environments, the methodology section is categorized into two phases: (i) fractional selectivity approach for task scheduling and (ii) FSCPSO algorithm for VM migration and allocation. The proposed FSCPSO algorithm with the fractional selectivity approach improves efficiency of resource utilization and fog data centers, thereby reducing the processing time of requests given by fog users. The descriptions of the fractional selectivity approach and FSCPSO algorithm are illustrated in Sections 3.1 and 3.2.

3.1 Fractional Selectivity Approach for Task Scheduling

The fractional selectivity approach is developed for task scheduling in fog computing environments as explained in this subsection. Fractional selectivity is a process of scheduling computing resources in the fog system. Here, the scheduling is based on the fraction of data or tasks that need processing at different fog nodes. For this research, the fractional selectivity approach is simulated utilizing iFogSim. Typically, the fractional selectivity approach distributes data across different fog nodes. These fog nodes are located very close to the edge device to reduce latency and improve the effectiveness of task scheduling. The workflow of the fractional selectivity approach is illustrated in Fig. 1. Initially, the fractional selectivity approach generates the workload for determining the selectivity of the features. After selecting the appropriate fog nodes, the remaining fractional workload is offloaded to the fog to process the selected fog nodes. This process will continue for task scheduling, as depicted in Fig. 1.

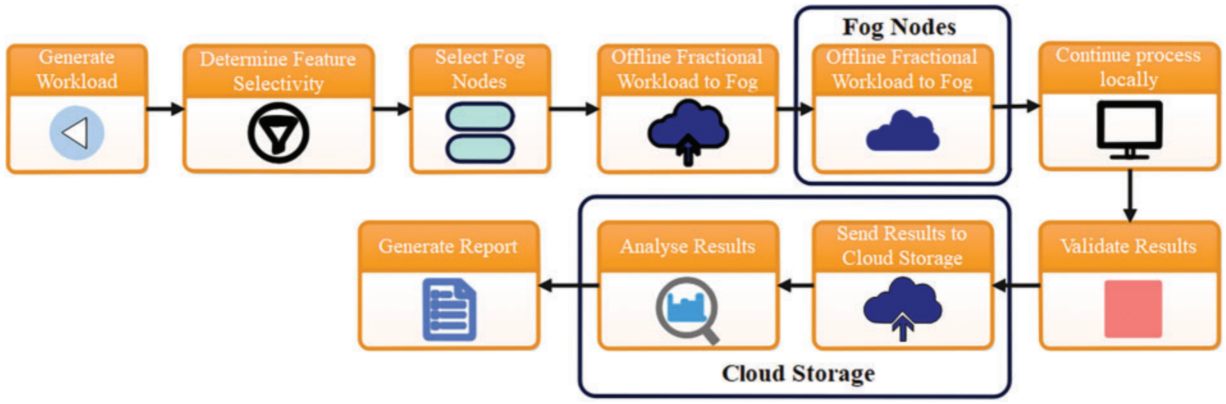

Generate Workload
Determine Feature Selectivity
Select Fog Nodes
Offline Fractional Workload to Fog
Fog Nodes
Offline Fractional Workload to Fog
Continue process locally
Validate Results
Send Results to Cloud Storage
Analyse Results
Generate Report
Cloud Storage

Figure 1: Workflow of the fractional selectivity approach

Fractional selectivity is considered one of the preferred novel methods for resource scheduling since it executes in a fine-grained and flexible manner to optimize resource allocation in distributed computing environments, especially in fog computing. The conventional resource scheduling approaches frequently allocate resources depending on coarse criteria, including prioritizing tasks. Fractional selectivity works at a finer granularity by identifying distinct data elements that permit accurate and data-driven resource allocation decisions. Acquiring precise data requires considering factors, such as data relevance, position, and resource necessities. This method is decisive in fog computing, where data is produced at the edge and must be processed proficiently. The suggested method adapts in real-time to varying settings and workload dissimilarities; it also dynamically allocates the resources by focusing on the changing needs of dissimilar data sources. This adaptability is crucial for managing the dynamic nature of IoT data and the variability of fog computing workloads.

Fractional selectivity optimizes the exploitation of these resources by choosing which data needs to be managed and which one needs to be uploaded to the cloud. This process is crucial for confirming the effectual usage of edge resources. By processing the most applicable data at the edge, fractional selectivity delivers a substantial decrease in latency which is useful for industrial automation and autonomous vehicles. Furthermore, it preserves the bandwidth by minimizing the data count that is required to be transferred to the cloud. This is predominantly beneficial when the bandwidth is inadequate. Fractional selectivity allows data customization decisions based on application-specific standards and strategies. Numerous applications describe their particular guidelines for the selection and processing of data which provides superior adaptability. In the IoT framework, a large quantity of data is produced from different devices and sensors, where fractional selectivity efficiently supports proper data control, which implies that only the most valuable data is processed in the system.

From the overall statements, it is thus inferred that fractional selectivity is indeed an effective novel method for resource scheduling in fog and edge computing environments. Moreover, it focusses on fine-grained resource allocation which makes it a valuable method for enhancing the efficiency and performance of edge computing applications.

3.2 FSCPSO Algorithm for VM Migration and Allocation

In this subsection, the execution procedure of the FSCPSO algorithm that is proposed for VM migration and allocation in fog computing environments, is being explained [38,39]. The conventional PSO algorithm is a bio-inspired metaheuristic optimization algorithm which mimics the foraging behavior of birds. In this algorithm, the foraging process of the bird's flock is considered as a function of the optimization process [40]. In a set of particles, every particle is determined with a velocity and position in order to search for the global solutions. Based on the individual's global optimal position and local optimal position, the particles i iteratively update their positions. In the conventional PSO algorithm, the position X_i and velocity V_i of the particle i is updated based on Eqs. (1) and (2).

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{1}$$

where $V_i(t+1)$ is the new particle's velocity, $X_i(t)$ is the present particle's position, $t+1$ is the post-update, and t is the temporal (current) status [41]. The particle's velocity $V_i(t+1)$ is updated based on Eq. (2).

$$V_i(t+1) = wV_i(t) + c_1 r_1 \left(X_i^p - X_i(t)\right) + c_2 r_2 \left(X^g - X_i(t)\right) \tag{2}$$

where r_1 and r_2 are random variables ranging between 0 to 1, c_1 and c_2 are constants, X^g is specified as the global best position *Gbest*, X^p is the particle's best position *Pbest*, $V_i(t)$ is the present particle's velocity, and w is the weight component of the velocity [42].

However, the conventional PSO algorithm struggles to balance the exploitation and exploration phases, which leads to premature convergence. To address this concern, the fitness sharing mechanism and chaos theory are incorporated into the conventional PSO algorithm, for efficiently balancing local exploitation and global exploration. Initially, in the fitness sharing mechanism, the actual fitness value of a particle is adjusted utilizing the sharing function, and this updated fitness is known as a 'shared fitness', which is utilized in the selection procedure. In the PSO algorithm, this modified fitness value is used to guide the probability of particle selection, wherein particles with higher shared fitness values are likely to be selected and the particles with lower shared fitness values are likely to be eliminated. This process promotes diversity in the population. The shared fitness is mathematically expressed in Eq. (3).

$$Shared\_fitness_i = \frac{Original\_fitness_i}{1 + \sum_{j=1}^{M} \frac{d_{ij}}{\sigma^2}} \tag{3}$$

σ is the scaling factor defining the range of influencing neighbourhood particles, d_{ij} is the distance between the particle i and its neighbourhood particles j, M is the total particles, $Original\_fitness_i$ is the original fitness of a particle i and $Shared\_fitness_i$ is the modified fitness of the particle i. On the other hand, the chaos theory utilizes chaotic functions or maps in order to introduce chaos in the conventional PSO algorithm. In the updated equations, the chaotic map modifies the values of the random numbers r_1 and r_2, as mathematically expressed in Eq. (4). In this study, a logistic map is utilized as a chaotic map.

$$X_{M+1} = r \times X_M \times (1 + V_i(t+1)) \tag{4}$$

X_{M+1} refers to updated particle position; $M+1$ denotes the post-update after utilizing the chaotic map; r is a parameter that defines the chaotic map's behavior which provides various outcomes on a minor change in the initial condition or control parameters. Therefore, formula of the updated particle's velocity $V_i(t+1)$ along with chaotic map's behaviour, is presented in Eq. (5).

$$V_i(t+1) = wV_i(t) + c_1 \times chaotic;(r, r_1) \times \left(X_i^p - X_i(t)\right) + c_2 \times chaotic;(r, r_2) \times \left(X^g - X_i(t)\right) \tag{5}$$

where $chaotic(r, X)$ is the chaotic map with r applied to a random number X. The best particle and other particles in the population are updated in various ways, that are not differentiated in PSO. Therefore, the velocity formula in Eq. (2) is updated by a chaotic map to avoid falling into a local optimum which is represented in Eq. (5). Moreover, the features of chaotic map in the best particle are used to maintain the diversity of population and enhance the global search ability. Also, the proposed FSCPSO algorithm terminates after reaching the maximum of iterations. The pseudo-code of this algorithm is depicted in Algorithm 1:

Algorithm 1: Pseudo-code of the FSCPSO algorithm

```
Input: Total particles in the swarm, total dimensions (parameters) for resource allocation, number of optimization iterations, inertia weight coefficient, acceleration coefficients for personal and global best influences, and chaotic parameter for initialization.
Output: Optimal resource allocation parameters and optimal value of the objective function
Initialize particle swarm:
    For every particle $i = 1$ to $num\_particles$:
        Initialize position $X_i$ randomly
        Initialize velocity $V_i$ randomly utilizing chaotic dynamics
        Evaluate fitness $X_i$ utilizing the objective function
        Set $Pbest_i = X_i$
        Set $Pbest\_value_i = fitness\ of\ X_i$
    Find $Gbest$ among $Pbest_i$ values
    Set $Gbest = X_i$ corresponding to $Gbest$
#Main optimization loop:
    For $every\ iteration = 1$ to $num\_iterations$:
        For every particle $i = 1$ to $num\_particles$:
            Update velocity $V_i$
            Update position $X_i$
            Evaluate the fitness of $X_i$ utilizing the objective function
            If fitness of $X_i$ is better than $Pbest\_value_i$:
                Set $Pbest_i = X_i$
                Set $Pbest\_value_i = fitness\ of\ X_i$
            If fitness of $X_i$is better than fitness of $Gbest$:
                Set $Gbest = X_i$
#Output:
    $best\_position = Gbest$
  $best\_value = fitness\ of\ Gbest$
End
```

The schematic representation of the proposed FSCPSO framework is specified in Fig. 2. Here, the task scheduling is done using the fractional selectivity approach. Those outputs are processed into the VM allocation stage, where the PSO method processes the allocation, which contains initialization parameters and its objective functions. The information acquired by the PSO VM allocation is processed into a fog device, enabling efficient task distribution and reducing response times and overall system cost in the fog computing architecture.

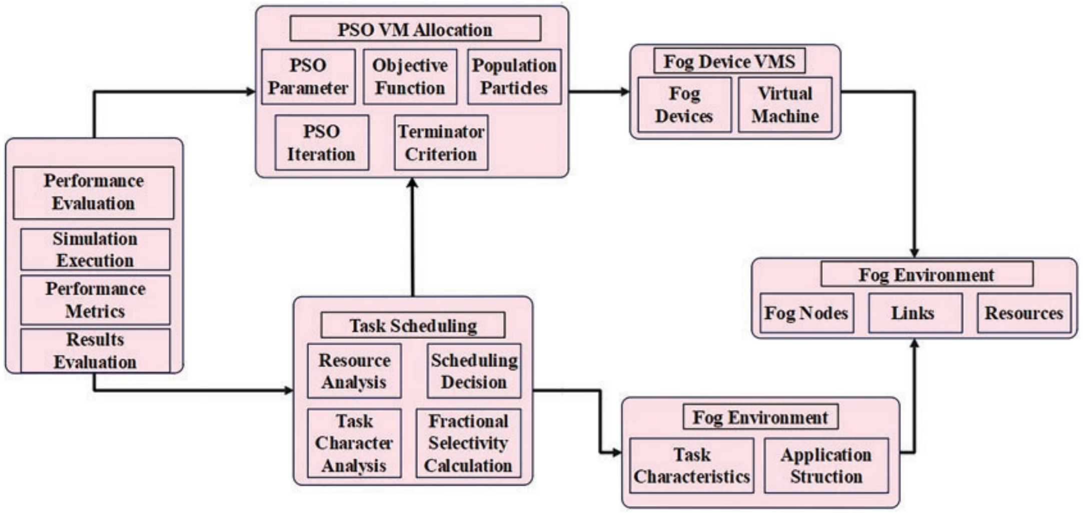


Figure 2: Schematic representation of the proposed framework

The optimization efficiency of FSCPSO is strengthened by selecting appropriate control parameters. The selection of suitable parameters is beneficial for comprehending ideal outcomes with a smaller amount of randomness. The parameter selection for the proposed FSCPSO algorithm majorly influences its performance and effectiveness. It is a complex optimization issue that defines ideal constraints for optimum performance. Until now, there were no wide-ranging approaches to define the optimum constraints designated by user evaluation since there are various parameter spaces among dissimilar parameters and contingency with each other. The regularity of various parameters influences the algorithm's performance. Therefore, in this paper, the performance of the proposed framework which is a combination of the fractional selectivity approach and FSCPSO algorithm is analyzed based on the control parameter variants, including the following settings wherein, the fractional selectivity threshold is 0.7, the iteration limit is 100, the social coefficient is 1.5, the cognitive coefficient is 1.5, inertia weight is 0.7, and the number of particles is 50. Based on the above-stated selection parameters, the results of LBL, ARU, makespan, total cost, energy consumption, and response time are analyzed, as described in the below sections along with their respective graphical representations. The selected condition of stopping iteration is the best position of the searched particle till it reaches the significant adaptive thresholds. The algorithm of the proposed framework is presented in Algorithm 2, and the empirical analysis of the proposed framework is specified in Section 4.

Algorithm 2: Algorithm of the proposed framework

```
Input: Fog environment specifications, task and application models, PSO algorithm parameters, and termination criteria.
Output: Optimized task scheduling and VM allocation
Initialization:
1. Define fog environment (nodes, communication links, and cloud resources)
2. Create task and application models
```

(Continued)

Algorithm 2 (continued)

```
3. Initialize fog devices and VMs
# Task scheduling with fractional selectivity approach:
4. Extract task information from the task model.
5. Analyse fog node capabilities (processing power and communication resources).
6. Calculate fractional selectivity for each task-fog node pair.
7. Make scheduling decisions using fractional selectivity values.
#FSCPSO algorithm for VM allocation:
8. Set up FSCPSO algorithm parameters (number of particles, iteration limit, and inertia weight).
9. Define an objective function based on iFogSim metrics.
10. Initialize the population of particles representing VM allocations.
11. Iterate FSCPSO for VM allocation until termination criteria are met:
    a. Update particle positions based on FSCPSO equations.
    b. Evaluate the performance of each particle using the objective function.
    c. Update personal and global best positions.
    d. Check termination criteria.
#Evaluation measures in iFogSim:
12. Simulate execution using iFogSim simulator
13. Collect evaluation measures (LBL, ARU, total cost, makespan, energy consumption, and response time).
14. Evaluate the efficiency and effectiveness of the optimized solution.
#Output:
15. Optimized task scheduling and VM allocation results.
End
```

4 Results

The proposed FSCPSO algorithm is implemented utilizing the iFogSim simulator, Eclipse Integrated Development Environment (IDE) 2022, and Java Development Kit (JDK) 18. This proposed algorithm is executed on a system with 1 TB hard drive, 128 GB memory, and Windows OS. In this manuscript, the FSCPSO algorithm is utilized for VM migration and allocation in fog nodes, and the fractional selectivity approach is applied for task scheduling. In this scenario, the minimum and maximum instance count ranges between 90 to 150, and the allocated memory is 256 MB. The FSCPSO algorithm's performance is analyzed utilizing the following evaluation measures: LBL, response time, ARU, makespan, energy consumption, and total cost. Firstly, the total cost is determined as the overall expenditure related to maintaining, operating, and deploying a fog computing infrastructure. In this particular context, the total cost is computed utilizing Eq. (6) for a specific set of tasks.

$$Totalcost = \sum_{i=0}^{n} CP\left(ti\right) + CM\left(ti\right) + CB\left(ti\right) \tag{6}$$

The total tasks are represented as n, bandwidth utilization cost is denoted as CB, memory utilization cost is indicated as CM, and processing cost is specified as CP. Next, the LBL is determined as the ratio of total overloaded fog servers to the balanced fog servers. In a fog computing environment, ARU is stated as the average level of resource consumption across different fog nodes.

The mathematical representation of the LBL and ARU are depicted in Eqs. (7) and (8).

$$LBL = \frac{BS}{FS} \times 100\% \tag{7}$$

$$ARU = \frac{(BS + OL)}{FS} \times 100\% \tag{8}$$

The available fog servers are denoted as FS, total overloaded fog servers are represented as OL, and balanced fog servers are indicated as BS. Furthermore, makespan is the time consumed by the proposed FSCPSO algorithm for completing all tasks. The mathematical derivation of makespan is represented in Eqs. (9)–(11).

$$Makespan = Max\,(CT\,(ti)) \tag{9}$$

$$CT\,(ti) = AT\,(ti) + TAT\,(ti) \tag{10}$$

$$TAT\,(ti) = WT\,(ti) + BT\,(ti) \tag{11}$$

where ti denotes task, CT is the time taken to complete a task, BT is the burst time, arrival time is indicated as AT, waiting time is represented as WT, and turnaround time is denoted as TAT. Additionally, in fog computing environments, energy consumption E_{total} is determined as the total electrical power consumed by fog computing infrastructure, specified in Eq. (12).

$$E_{total} = \sum\nolimits_{i=1}^{n} PC_i \times Time \tag{12}$$

where $Time$ is the time duration, PC_i is the power consumption of i^{th} fog node, and total tasks are n. Furthermore, the response time RT is the time consumed by the FSCPSO algorithm for processing and responding to a given request, as depicted in Eq. (13).

$$RT = RT_{processing} + RT_{communication} \tag{13}$$

wherein $RT_{communication}$ is the time consumed during communication between central cloud and fog nodes, and $RT_{processing}$ is the time consumed for local processing of fog nodes.

The overall energy consumption of all fog devices and cloud servers must be kept to a minimum in order to achieve the trade-off between fog energy consumption, latency, and cost. This study formulates a processing model for the challenge of the trade-off between energy consumption, delay, and cost in fog processing workloads. The FSCPSO method is used to solve this multi-objective problem model. The numerical findings demonstrate that it is possible to reduce energy consumption, latency, and cost when the suggested technique is employed for workload allocation in a fog scenario, which is clearly mentioned in Eqs. (6)–(13).

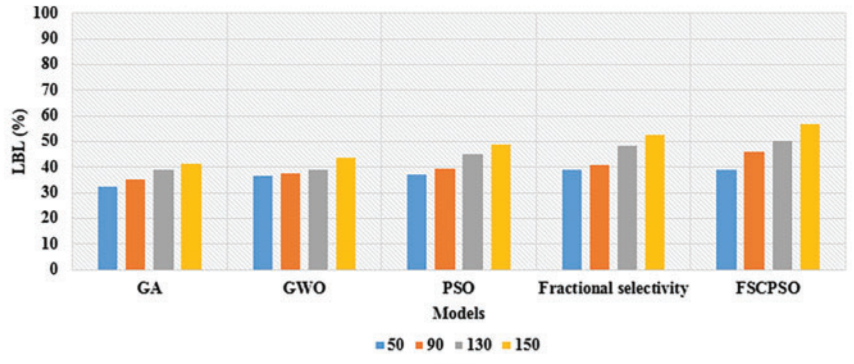
4.1 Quantitative Evaluation

In this context, the FSCPSO algorithm's performance is analyzed using six various evaluation measures and its obtained results are compared with those achieved by conventional algorithms GA, Grey Wolf Optimization (GWO), PSO, and fractional selectivity approach. The performance of the proposed and conventional optimization algorithms are analyzed for the varied number of tasks, i.e., 50, 90, 130, and 150. The LBL analyses of the FSCPSO as well as the conventional optimization algorithms, are presented in Table 1. By investigating Table 1, it is evident that the FSCPSO algorithm has gained better LBL outcomes as compared to other optimization algorithms when analyzed at different thresholds of 0.7, 0.5, and 0.3. The FSCPSO algorithm attains LBL of 39.12%, 45.85%, 50.12%, and 56.86% respectively for 50, 90, 130, and 150 tasks at 0.7 thresholds, which are higher values when compared with other algorithms' threshold values. These results reveal that the proposed FSCPSO's improvement in LBL is almost 0.5% to 7% greater than other comparative optimization algorithms. The LBL analysis for the FSCPSO algorithm *vs.* existing optimization algorithms for threshold values 0.7, 0.5, and 0.3 is depicted in Fig. 3.

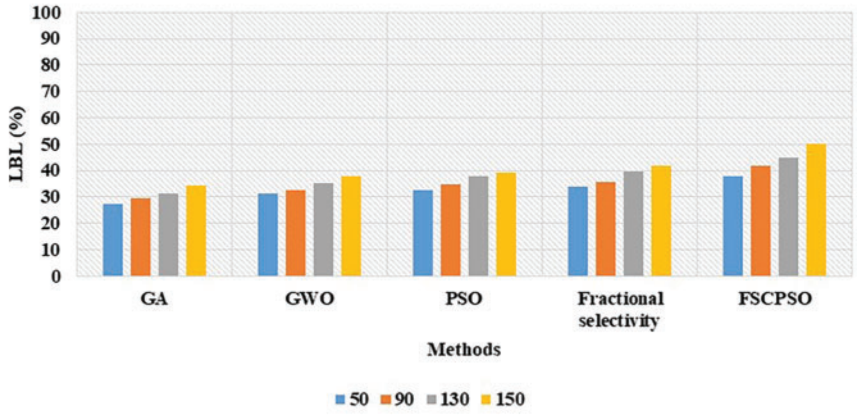
Table 1: LBL analysis of FSCPSO algorithm and conventional optimization algorithms

LBL (%) at Threshold value of 0.7					
Number of tasks	GA	GWO	PSO	Fractional selectivity	FSCPSO
50	32.56	36.78	37.11	38.78	39.12
90	35.25	37.62	39.45	40.84	45.85
130	38.77	39.05	45.11	48.26	50.12
150	41.29	43.56	48.88	52.74	56.86
LBL (%) at Threshold value of 0.5					
Number of tasks	GA	GWO	PSO	Fractional selectivity	FSCPSO
50	27.25	31.44	32.42	33.73	37.97
90	29.37	32.73	34.63	35.81	41.89
130	31.46	35.38	37.84	39.64	44.84
150	34.33	37.88	39.27	41.93	49.92
LBL (%) at Threshold value of 0.3					
Number of tasks	GA	GWO	PSO	Fractional selectivity	FSCPSO
50	23.72	26.42	28.43	31.28	34.66
90	25.19	28.49	31.55	33.34	37.91
130	26.64	30.68	33.46	34.96	39.73
150	29.27	32.77	35.21	37.72	44.84

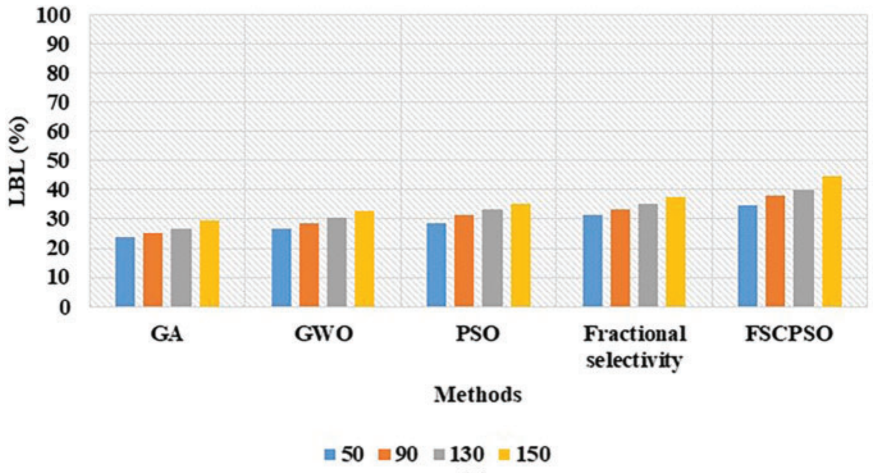
Furthermore, the ARU analyses of the FSCPSO algorithm with the other traditional optimization algorithms namely GA, GWO, PSO, and fractional selectivity, at the different threshold values of 0.7, 0.5, and 0.3 are presented in Table 2. As depicted in Table 2, the FSCPSO algorithm achieves higher ARU at threshold 0.7 with the values of 58.15%, 64.42%, 75.55%, and 89.85% for 50, 90, 130, and 150 numbers of tasks, and therefore it is inferred that the achieved results are superior in comparison to those achieved by other existing algorithms, for the same analyses. The ARU analyses for the FSCPSO algorithm *vs.* the existing optimization algorithms for threshold values of 0.7, 0.5, and 0.3 are specified in Fig. 4. The FSCPSO algorithm continuously adjusts and explores resource allocations on the basis of the current workload. The FSCPSO algorithm rapidly responds to the change in demand because of its adaptive/flexible nature. This mechanism ensures that the resources are effectively assigned in order to meet the evolving needs of this system. The FSCPSO algorithm prevents resource under-utilization by flexibly adapting to the workload variation. This flexibility maintains high ARU, even when handling unpredictable workloads.



(a)



(b)

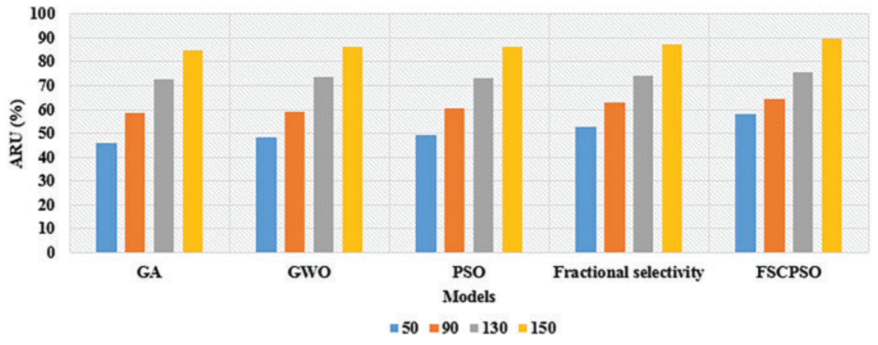


(c)

Figure 3: Pictorial presentation of LBL analysis for FSCPSO algorithm and existing optimization algorithms at various threshold values (a) 0.7, (b) 0.5, (c) 0.3

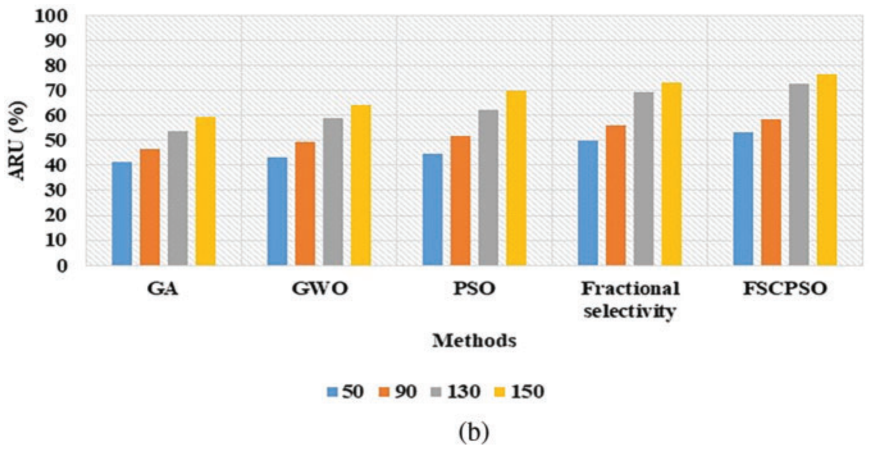
Table 2: ARU analysis of FSCPSO algorithm and traditional optimization algorithms

ARU (%) at Threshold value of 0.7					
Number of tasks	GA	GWO	PSO	Fractional selectivity	FSCPSO
50	46.15	48.44	49.12	52.86	58.15
90	58.35	59.15	60.33	62.84	64.42
130	72.48	73.56	73.22	74.15	75.55
150	85.06	86.52	86.45	87.26	89.85
ARU (%) at Threshold value of 0.5					
Number of tasks	GA	GWO	PSO	Fractional selectivity	FSCPSO
50	41.38	43.18	44.67	49.76	53.37
90	46.49	49.63	51.58	56.28	58.19
130	53.73	58.71	62.33	69.17	72.43
150	59.52	64.17	69.86	73.26	76.38
ARU (%) at Threshold value of 0.3					
Number of tasks	GA	GWO	PSO	Fractional selectivity	FSCPSO
50	34.88	39.42	43.41	47.68	50.73
90	38.29	45.55	49.69	53.85	55.49
130	41.15	57.60	60.43	62.74	68.16
150	50.77	54.81	63.25	69.46	73.31



(a)

Figure 4: (Continued)



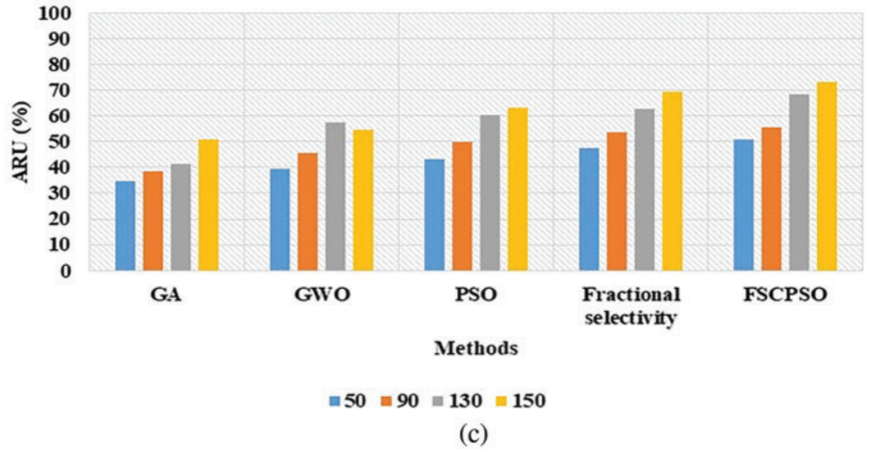


Figure 4: Pictorial presentation of ARU analysis for FSCPSO algorithm and existing optimization algorithms at various threshold values (a) 0.7, (b) 0.5, (c) 0.3

The comparative evaluation between the FSCPSO algorithm and the existing optimization algorithms namely GA, GWO, PSO, and fractional selectivity is carried out on the basis of total cost and makespan as is presented in Table 3. By viewing Table 3, the proposed FSCPSO algorithm accomplishes minimum total costs of 1175, 1185, 1191, and 1198, respectively for the varied task numbers of 50, 90, 130, and 150. Similarly, it achieves the makespans of 87.87, 116.85, 186.78, and 210.22 ms for varying task quantities of 50, 90, 130, and 150, respectively. While processing the 50th task, the makespan of fractional selectivity is less since it executes a smaller number of tasks. Moreover, FSCPSO is an extension of the fractional selectivity process, hence, it implements an even smaller number of tasks, thereby, consuming less makespan than the fractional selectivity process alone. Thus, the combinational model of the FSCPSO algorithm has indeed proven to be more effective. In this context, based on the present fitness value, the proposed FSCPSO algorithm effectively adjusts the allocation of computational tasks. The concept of fitness sharing prevents premature convergence to sub-optimal solutions and encourages diversity in the solutions. Additionally, the fitness sharing enables this proposed FSCPSO algorithm to explore and exploit the solution space. Effective task scheduling reduces makespan and total costs in the fog computing environment. Integrating fitness sharing mechanism and chaos theory in the PSO algorithm efficiently balances local exploitation and global exploration.

Table 3: Total cost and makespan of FSCPSO algorithm and traditional optimization algorithms

			Total cost		
Number of tasks	GA	GWO	PSO	Fractional selectivity	FSCPSO
50	2790	3100	1375	1269	1175
90	2806	3105	1385	1278	1185
130	2856	3115	1390	1282	1191
150	2889	3120	1395	1294	1198
			Makespan (ms)		
50	85.48	90	89.45	82.45	87.87
90	120.55	128.8	120.26	118.56	116.85
130	230.18	218.45	210.75	198.04	186.78
150	240.44	220.77	215.92	208.52	210.22

This balance ensures that the proposed FSCPSO algorithm explores an extensive range of solutions and results in minimal makespan and total cost when converging towards optimal task scheduling. Figs. 5 and 6 depict the pictorial visualization of the total cost and makespan outcomes produced by the FSCPSO algorithm and comparative optimization algorithms.

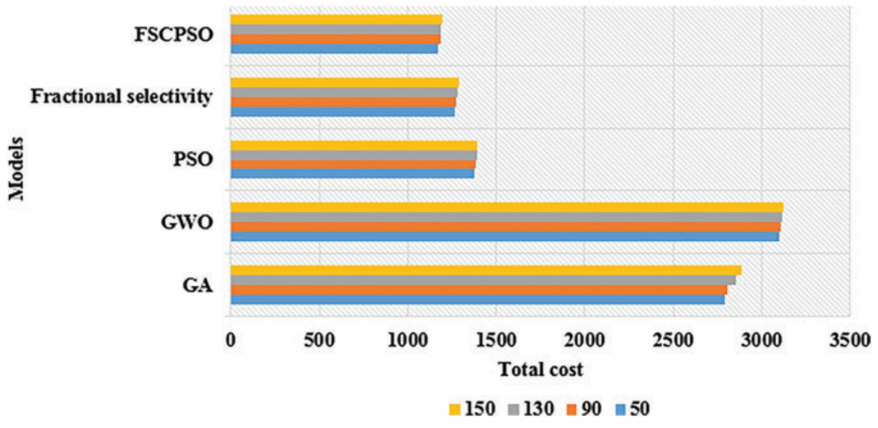


Figure 5: Pictorial visualization of the total cost for the FSCPSO algorithm and existing optimization algorithms

In addition to other evaluation measures, the energy consumption and response time of the FSCPSO algorithm and comparative optimization algorithms of GA, GWO, PSO, and fractional selectivity, are mentioned in Table 4. The FSCPSO algorithm consumes minimal energies of 1.2, 1.9, 2.8, and 3.8 J, respectively, for varying task quantities of 50, 90, 130, and 150. Due to its fitness-sharing ability, it identifies the best suitable resources for VM, in contrast to other existing methods. Compared to this proposed FSCPSO algorithm, the existing optimization algorithms consume maximal energy consumption.

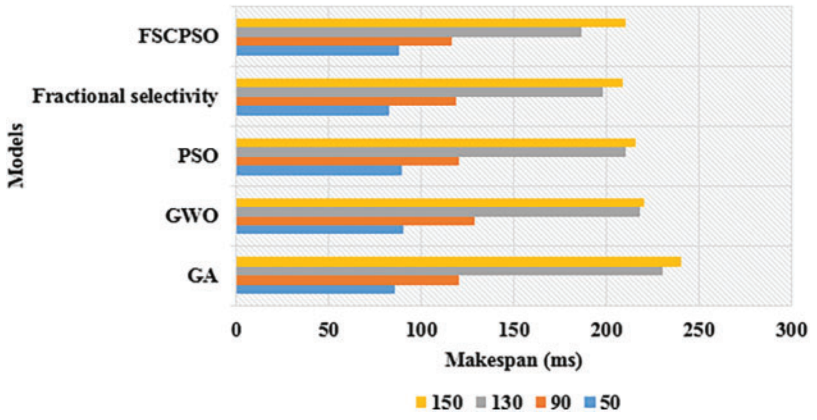


Figure 6: Pictorial analysis of makespan for FSCPSO algorithm and comparative optimization algorithms

Table 4: Energy consumption and response time of FSCPSO algorithm and comparative optimization algorithms

Energy consumption (Joule)					
Number of tasks	GA	GWO	PSO	Fractional selectivity	FSCPSO
50	2	2.4	1.8	1.6	1.2
90	2.8	2.5	2.8	2.2	1.9
130	3.5	3.8	3.9	3.5	2.8
150	4	4.3	4.5	4	3.8
Response time (ms)					
50	205.23	190.82	183.54	177..44	120.85
90	328.70	316.40	294.80	277.38	230.95
130	380.75	340.98	302.50	280.96	250.26
150	390.40	355.46	308.44	304.82	270.52

Different task quantities of 50, 90, 130, and 150 are taken to analyze the scalability, wherein the FSCPSO algorithm respectively consumes 120.85, 230.95, 250.26, and 270.52 ms of minimal response times. Thus, the FSCPSO algorithm's response time is the least when compared to the time consumption of the existing algorithms GA, GWO, PSO, and fractional selectivity, as mentioned in Table 4. As discussed in the prior section, the proposed FSCPSO algorithm adaptively assigns computational tasks to fog nodes based on their fitness values. This proposed algorithm achieves a balanced task distribution by considering the chaos theory and fitness sharing. The dynamic resource allocation by the FSCPSO algorithm prevents resource overloading and under-utilization, resulting in optimized response time and improved energy efficiency. The pictorial evaluation of energy consumption and response time for the FSCPSO algorithm and existing optimization algorithms is depicted in Figs. 7 and 8.

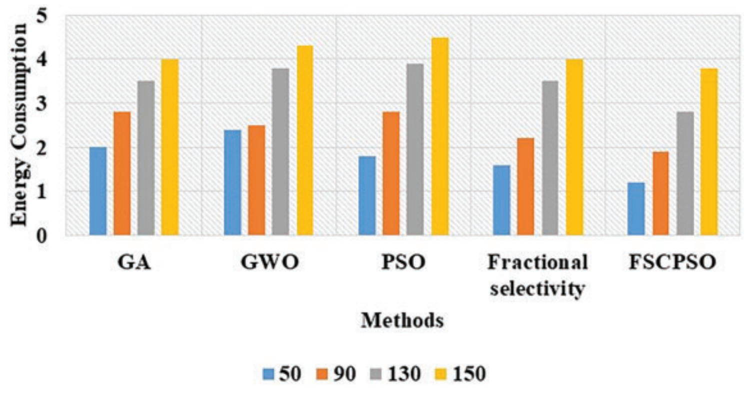


Figure 7: Pictorial evaluation of energy consumption for the FSCPSO algorithm and existing optimization algorithms

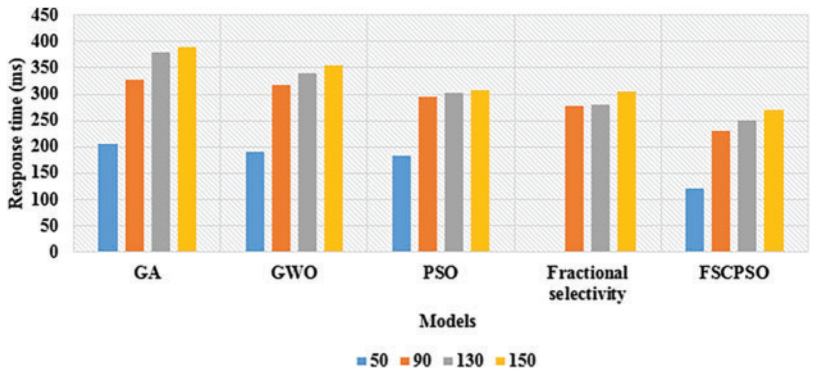


Figure 8: Pictorial visualization of response time for the FSCPSO algorithm and existing optimization algorithms

4.2 Comparative Evaluation with Discussion

In the context of VM migration and allocation, and task scheduling in fog computing environments, the FSCPSO algorithm's performance is related to the existing algorithms presented by Talaat et al. [19], who integrated a modified PSO algorithm with the CNN model for effective task scheduling in fog systems. The comparative outcomes between the modified PSO-CNN algorithm and the FSCPSO algorithm are presented in Table 5. By studying Table 5, it is evident that the proposed FSCPSO algorithm achieves an impressive performance in task scheduling and VM allocation, especially in fog computing environments. As stated in Table 5, the following four evaluation measures, LBL, ARU, makespan, and total cost, are utilized for investigating the performance between the modified PSO-CNN algorithm and the FSCPSO algorithm.

By inspecting Tables 1–5, it is observed that the proposed FSCPSO algorithm along with the fractional selectivity approach achieves an impressive performance in task scheduling and VM allocation in the fog computing environment. The major significance of this research is that fitness sharing improves the diversity in solutions. Furthermore, the inclusion of fitness sharing in the PSO algorithm enhances response time, reduces energy consumption, and effectively utilizes resources by preventing the concentration of tasks on particular nodes. On the other hand, the incorporation of chaos theory in the PSO algorithm introduces randomness in the particle movement, thereby aiding in the global exploration of the solution space. The deficiency of mathematical analysis is a substantial drawback, especially with regards to efficiency and delay. For most metaheuristic algorithms, convergence analysis is still open, making it difficult to ascertain their convergence qualities. The random exploration of chaos theory provides the optimum configuration of task scheduling. This in turn results in faster response time, improved energy efficiency, and avoids local optima problems.

Table 5: Comparative outcomes between the modified PSO-CNN and FSCPSO algorithms

Models	Tasks	LBL (%)	ARU (%)	Makespan (ms)	Total cost
Modified PSO-CNN [19]	50	35.99	56	89.15	1363
	90	39.45	59	151	1365
	130	49.77	75.18	229	1367
	150	52.76	87	250	1370
FSCPSO	50	39.12	58.15	85.87	1175
	90	45.85	64.42	116.85	1185
	130	50.12	75.55	186.78	1191
	150	56.86	89.85	208.52	1198

5 Conclusion

In this manuscript, a new framework is proposed for effective task scheduling and VM allocation in fog computing environments. The proposed framework comprises two phases namely, task scheduling using a fractional selectivity approach and VM allocation utilizing the FSCPSO algorithm. Initially, the fractional selectivity approach adaptively utilizes the resources that overcome the problem of resource under-utilization. The adaptive task scheduling by fractional selectivity approach in fog systems includes advantages such as improved fault tolerance and task prioritization, flexible load balancing, optimized cost, and energy efficiency, alongside being adaptive to dynamic workloads. In addition to this, the FSCPSO algorithm applies a chaotic component for balancing exploration and exploitation stages and employs fitness sharing for increasing the diversity between solutions. These two processes make this algorithm more adaptive to dynamic workloads, ensuring effective resource usage and providing a resilient solution to optimize VM allocation in the fog computing environment.

Six evaluation measures known as LBL, ARU, total cost, makespan, energy consumption, and response time, are utilized to investigate the proposed framework's performance. In comparison to the existing optimization algorithms: GA, GWO, and PSO for a task quantity of 50, the proposed FSCPSO algorithm obtains a minimal total cost of 1175, a makespan of 85.87 ms, an energy consumption of 1.2 J, a response time of 120.85 ms, maximal LBL of 39.12%, and ARU of 58.15%. Furthermore, the proposed FSCPSO algorithm achieves impressive task scheduling and VM allocation performance as opposed to other optimization algorithms for task quantities of 90, 130, and 150. Fog computing requires resource allocation strategies that protect privacy since fog nodes are susceptible to many types of attacks and threats against their data. Authentication at different levels in the fog layer is another major issue that calls for the best possible approach to ensure security. As a future extension, data security, reliability and analytics may be considered in the fog computing environment by employing task scheduling and dynamic VM allocation algorithms for real-time applications (hospitals, hotels), wherein latency is considered as the major criteria.

Acknowledgement: Not applicable.

Funding Statement: This work was supported in part by the National Science and Technology Council of Taiwan, under Contract NSTC 112-2410-H-324-001-MY2.

Author Contributions: The authors confirm their contribution to the paper as follows: study conception and design: Prasanna Kumar Kannughatta Ranganna, Siddesh Gaddadevara Matt, Ananda Babu Jayachandra; data collection: Prasanna Kumar Kannughatta Ranganna, Chin-Ling Chen; analysis and interpretation of results: Prasanna Kumar Kannughatta Ranganna, Siddesh Gaddadevara Matt, Chin-Ling Chen, Yong-Yuan Deng; draft manuscript preparation: Prasanna Kumar Kannughatta Ranganna, Siddesh Gaddadevara Matt, Ananda Babu Jayachandra. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Ethics Approval: The study not included human or animal subjects.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] A. S. Kadhim and M. E. Manaa, "Hybrid load-balancing algorithm for distributed fog computing in internet of things environment," *Bull. Electr. Eng. Inform.*, vol. 11, no. 6, pp. 3462–3470, Dec. 2022. doi: 10.11591/eei.v11i6.4127.

[2] A. Abuhamdah and M. Al-Shabi, "Hybrid load balancing algorithm for fog computing environment," *Int. J. Softw. Eng. Comput. Syst.*, vol. 8, no. 1, pp. 11–21, Feb. 2022. doi: 10.15282/ijsecs.8.1.2022.2.0092.

[3] V. Gowri and B. Baranidharan, "Dynamic energy efficient load balancing approach in fog computing environment," in *Lecture Notes on Data Engineering and Communications Technologies*, 1st ed. Singapore: Springer Nature Singapore, 2022, vol. 131, pp. 145–160. doi: 10.1007/978-981-19-1844-5_13.

[4] S. Singhal *et al.*, "Energy aware load balancing framework for smart grid using cloud and fog computing," *Sensors*, vol. 23, no. 7, pp. 3488, Mar. 2023. doi: 10.3390/s23073488.

[5] M. Kaur and R. Aron, "An energy-efficient load balancing approach for scientific workflows in fog computing," *Wireless Pers Commun*, vol. 125, no. 4, pp. 3549–3573, Aug. 2022. doi: 10.1007/s11277-022-09724-9.

[6] N. Tahmasebi-Pouya, M. A. Sarram, and S. Mostafavi, "A reinforcement learning-based load balancing algorithm for fog computing," *Telecommun. Syst.*, vol. 84, no. 3, pp. 321–339, Nov. 2023. doi: 10.1007/s11235-023-01049-7.

[7] M. Fahad, M. Shojafar, M. Abbas, I. Ahmed, and H. Ijaz, "A multi-queue priority-based task scheduling algorithm in fog computing environment," *Concurrency Comput. Pract. Exper.*, vol. 34, no. 28, pp. e7376, Dec. 2022. doi: 10.1002/cpe.7376.

[8] M. M. Razaq, S. Rahim, B. Tak, and L. Peng, "Fragmented task scheduling for load-balanced fog computing based on Q-learning," *Wireless Commun. Mobile Comput.*, vol. 2022, no. 3, pp. 4218696, Mar. 2022. doi: 10.1155/2022/4218696.

[9] S. K. Mani and I. Meenakshisundaram, "Improving quality-of-service in fog computing through efficient resource allocation," *Comput. Intell.*, vol. 36, no. 4, pp. 1527–1547, Nov. 2020. doi: 10.1111/coin.12285.

[10] S. Selvaganapathy and M. Chinnadurai, "Virtual machine placement in energy aware load balancer using fog classifier," *J. Cloud Comput.*, vol. 12, no. 1, pp. 180, Dec. 2023. doi: 10.1186/s13677-023-00559-8.

[11] M. I. Khaleel and M. M. Zhu, "Adaptive virtual machine migration based on performance-to-power ratio in fog-enabled cloud data centers," *J. Supercomput.*, vol. 77, no. 10, pp. 11986–12025, Oct. 2021. doi: 10.1007/s11227-021-03753-0.

[12] C. Ahlawat and R. Krishnamurthi, "HCDQN-ORA: A novel hybrid clustering and deep Q-network technique for dynamic user location-based optimal resource allocation in a fog environment," *J. Supercomput.*, vol. 80, no. 8, pp. 1–52, Jan. 2024. doi: 10.1007/s11227-023-05832-w.

[13] F. M. Talaat, M. S. Saraya, A. I. Saleh, H. A. Ali, and S. H. Ali, "A load balancing and optimization strategy (LBOS) using reinforcement learning in fog computing environment," *J. Ambient Intell. Hum. Comput.*, vol. 11, no. 11, pp. 4951–4966, Nov. 2020. doi: 10.1007/s12652-020-01768-8.

[14] R. Ghafari and N. Mansouri, "E-AVOA-TS: Enhanced African vultures optimization algorithm-based task scheduling strategy for fog-cloud computing," *Sustainable Comput. Inf. Syst.*, vol. 40, no. 1, pp. 100918, Dec. 2023. doi: 10.1016/j.suscom.2023.100918.

[15] R. Ghafari and N. Mansouri, "An efficient task scheduling in fog computing using improved artificial hummingbird algorithm," *J. Comput. Sci.*, vol. 74, pp. 102152, Dec. 2023. doi: 10.1016/j.jocs.2023.102152.

[16] H. A. Alharbi, T. E. Elgorashi, and J. M. Elmirghani, "Energy efficient virtual machines placement over cloud-fog network architecture," *IEEE Access*, vol. 8, pp. 94697–94718, May 2020. doi: 10.1109/ACCESS.2020.2995393.

[17] C. Jiang, L. Yang, and R. Shi, "An energy-aware virtual machine migration strategy based on three-way decisions," *Energy Rep.*, vol. 7, no. 2, pp. 8597–8607, Nov. 2021. doi: 10.1016/j.egyr.2021.02.029.

[18] D. B. Abdullah and W. Hadeed, "Container live migration in edge computing: A real-time performance amelioration," *Int. J. Appl. Sci. Eng.*, vol. 19, no. 3, pp. 2022121, Feb. 2022. doi: 10.6703/IJASE.202209_19(3).007.

[19] N. A. Sultan and D. B. Abdullah, "Investigating scientific collaboration networks in Iraq using cloud computing and data mining," *Int. J. Appl. Sci. Eng.*, vol. 20, no. 4, pp. 2023241, Jul. 2023. doi: 10.6703/IJASE.202312_20(4).003.

[20] A. M. Yadav, K. N. Tripathi, and S. C. Sharma, "An enhanced multi-objective fireworks algorithm for task scheduling in fog computing environment," *Cluster Comput.*, vol. 25, no. 2, pp. 983–998, Apr. 2022. doi: 10.1007/s10586-021-03481-3.

[21] F. M. Talaat, H. A. Ali, M. S. Saraya, and A. I. Saleh, "Effective scheduling algorithm for load balancing in fog environment using CNN and MPSO," *Knowl. Inf. Syst.*, vol. 64, no. 3, pp. 773–797, Mar. 2022. doi: 10.1007/s10115-021-01649-2.

[22] I. Z. Yakubu and M. Murali, "An efficient meta-heuristic resource allocation with load balancing in IoT-Fog-cloud computing environment," *J. Ambient Intell. Hum. Comput.*, vol. 14, no. 3, pp. 2981–2992, Mar. 2023. doi: 10.1007/s12652-023-04544-6.

[23] S. D. Vispute and P. Vashisht, "Energy-efficient task scheduling in fog computing based on particle swarm optimization," *SN Comput. Sci.*, vol. 4, no. 4, pp. 391, May 2023. doi: 10.1007/s42979-022-01639-3.

[24] D. Baburao, T. Pavankumar, and C. S. R. Prabhu, "Load balancing in the fog nodes using particle swarm optimization-based enhanced dynamic resource allocation method," *Appl. Nanosci.*, vol. 13, no. 2, pp. 1045–1054, Feb. 2023. doi: 10.1007/s13204-021-01970-w.

[25] S. Subbaraj, R. Thiyagarajan, and M. Rengaraj, "A smart fog computing based real-time secure resource allocation and scheduling strategy using multi-objective crow search algorithm," *J. Ambient Intell. Hum. Comput.*, vol. 14, no. 2, pp. 1003–1015, Feb. 2023. doi: 10.1007/s12652-021-03354-y.

[26] X. Li, Z. Zang, F. Shen, and Y. Sun, "Task offloading scheme based on improved contract net protocol and beetle antennae search algorithm in fog computing networks," *Mobile Netw. Appl.*, vol. 25, no. 6, pp. 2517–2526, Dec. 2020. doi: 10.1007/s11036-020-01593-5.

[27] A. Kishor and C. Chakarbarty, "Task offloading in fog computing for using smart ant colony optimization, Wireless Pers," *Wireless Pers. Commun.*, vol. 127, no. 2, pp. 1683–1704, Nov. 2022. doi: 10.1007/s11277-021-08714-7.

[28] M. Abbasi, E. M. Pasand, and M. R. Khosravi, “Workload allocation in IoT-fog-cloud architecture using a multi-objective genetic algorithm,” *J. Grid Comput.*, vol. 18, no. 1, pp. 43–56, Mar. 2020. doi: 10.1007/s10723-020-09507-1.

[29] S. S. Hajam and S. A. Sofi, “Spider monkey optimization based resource allocation and scheduling in fog computing environment,” *High-Confid. Comput.*, vol. 3, no. 3, pp. 100149, Sep. 2023. doi: 10.1016/j.hcc.2023.100149.

[30] A. Ragmani, A. Elomri, N. Abghour, K. Moussaid, and M. Rida, “FACO: A hybrid fuzzy ant colony optimization algorithm for virtual machine scheduling in high-performance cloud computing,” *J. Ambient Intell. Hum. Comput.*, vol. 11, no. 10, pp. 3975–3987, Oct. 2020. doi: 10.1007/s12652-019-01631-5.

[31] A. Kaur *et al.*, “Algorithmic approach to virtual machine migration in cloud computing with updated SESA algorithm,” *Sensors*, vol. 23, no. 13, pp. 6117, Jul. 2023. doi: 10.3390/s23136117.

[32] J. Sha, A. G. Ebadi, D. Mavaluru, M. Alshehri, O. Alfarraj and L. Rajabion, “A method for virtual machine migration in cloud computing using a collective behavior-based metaheuristics algorithm,” *Concurrency Comput. Pract. Exper.*, vol. 32, no. 2, pp. e5441, Jan. 2020. doi: 10.1002/cpe.5441.

[33] S. G. Sutar, P. J. Mali, and A. Y. More, “Resource utilization enhancemnet through live virtual machine migration in cloud using ant colony optimization algorithm,” *Int. J. Speech Technol.*, vol. 23, no. 1, pp. 79–85, Mar. 2020. doi: 10.1007/s10772-020-09682-2.

[34] Y. Dai, Q. Zhang, and L. Yang, “Virtual machine migration strategy based on multi-agent deep reinforcement learning,” *Appl. Sci.*, vol. 11, no. 17, pp. 7993, Aug. 2021. doi: 10.3390/app11177993.

[35] E. Torre *et al.*, “A dynamic evolutionary multi-objective virtual machine placement heuristic for cloud data centers,” *Inf. Softw. Technol.*, vol. 128, no. 5e, pp. 106390, Dec. 2020. doi: 10.1016/j.infsof.2020.106390.

[36] K. Ajmera and T. K. Tewari, “VMS-MCSA: Virtual machine scheduling using modified clonal selection algorithm,” *Cluster Comput.*, vol. 24, no. 4, pp. 3531–3549, Dec. 2021. doi: 10.1007/s10586-021-03320-5.

[37] S. Gharehpasha, M. Masdari, and A. Jafarian, “Virtual machine placement in cloud data centers using a hybrid multi-verse optimization algorithm,” *Artif. Intell. Rev.*, vol. 54, no. 3, pp. 2221–2257, Mar. 2021. doi: 10.1007/s10462-020-09903-9.

[38] N. Geng, Z. Chen, Q. A. Nguyen, and D. Gong, “Particle swarm optimization algorithm for the optimization of rescue task allocation with uncertain time constraints,” *Complex Intell. Syst.*, vol. 7, no. 2, pp. 873–890, Apr. 2021. doi: 10.1007/s40747-020-00252-2.

[39] M. Li, C. Liu, K. Li, X. Liao, and K. Li, “Multi-task allocation with an optimized quantum particle swarm method,” *Appl. Soft. Comput.*, vol. 96, no. 6, pp. 106603, Nov. 2020. doi: 10.1016/j.asoc.2020.106603.

[40] S. A. Alsaidy, A. D. Abbood, and M. A. Sahib, “Heuristic initialization of PSO task scheduling algorithm in cloud computing,” *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 6, pp. 2370–2382, Jun. 2022. doi: 10.1016/j.jksuci.2020.11.002.

[41] X. Ren, Z. Zhang, S. Chen, and K. Abnoosian, “An energy-aware method for task allocation in the Internet of things using a hybrid optimization algorithm,” *Concurrency Comput. Pract. Exper.*, vol. 33, no. 6, pp. e5967, Mar. 2021. doi: 10.1002/cpe.5967.

[42] M. Roshanzamir, M. A. Balafar, and S. N. Razavi, “A new hierarchical multi group particle swarm optimization with different task allocations inspired by holonic multi agent systems,” *Expert. Syst. Appl.*, vol. 149, no. 1, pp. 113292, Jul. 2020. doi: 10.1016/j.eswa.2020.113292.