Optimal Scheduling using Advanced Cat Swarm Optimization Algorithm to Improve Performance in Fog Computing

Xiaoyan Huo[1]*, Xuemei Wang[2]
Information Construction and Management Center, Jiaozuo University, Jiaozuo, Henan, 454003, China[1]
Academic Affairs Division, Jiaozuo Technical College, Jiaozuo Henan, 454000, China[2]

***Abstract*—Fog computing can be considered a decentralized computing approach that essentially extends the capabilities offered by cloud computing to the periphery of the network. In addition, due to its proximity to the user, fog computing proves to be highly efficient in minimizing the volume of data that needs to be transmitted, reducing overall network traffic, and shortening the distance that data must travel. But this technology, like other new technologies, has challenges, and scheduling and optimal allocation of resources is one of the most important of these challenges. Accordingly, this research aims to propose an optimal solution for efficient scheduling within the fog computing environment through the application of the advanced cat swarm optimization algorithm. In this solution, the two main behaviors of cats are implemented in the form of seek and tracking states. Accordingly, processing nodes are periodically examined and categorized based on the number of available resources; servers with highly available resources are prioritized in the scheduling process for efficient scheduling. Subsequently, the congested servers, which may be experiencing various issues, are migrated to alternative servers with ample resources using the virtual machine live migration technique. Ultimately, the effectiveness of the proposed solution is assessed using the iFogSim simulator, demonstrating notable reductions in execution time and energy consumption. So, the proposed solution has led to a 20% reduction in execution time while also improving energy efficiency by more than 15% on average. This optimization represents a trade-off between improving performance and reducing resource consumption.**

***Keywords*—Scheduling; fog computing; optimal balancing; cat swarm optimization algorithm**

I. INTRODUCTION

In order to overcome the challenges arising from resource limitations in IoT devices, the prevalent approach has been to rely on large-scale cloud data centers for interactions between IoT devices and supporting end servers [1]. However, as the number of IoT devices and their generated data continue to escalate, reliance on cloud-based infrastructure has become costly, inefficient, and often unfeasible [2]. In response to this issue, fog computing has emerged as a solution by offering networking, storage, and computing resources in proximity to IoT devices and users [3, 4]. One notable advantage of fog computing is its ability to reduce service latency for end-user applications, unlike the cloud, which typically exhibits higher latency due to its more extensive computing capacity and remote storage [5]. The exponential growth of the Internet of Things has posed significant challenges for cloud computing, including network failures and increased latency. To tackle these challenges, cloud computing has sought to bring cloud capabilities closer to IoT devices. Fog computing entails the utilization of heterogeneous and distributed processing nodes, presenting challenges for fog-based services in accommodating the diverse aspects of a constrained environment [23]. By examining the structural and service-oriented characteristics of fog computing, various challenges become apparent, with optimal scheduling being particularly significant. Scheduling holds great importance in the realm of the Internet of Things as it has the potential to decrease execution time and minimize energy consumption [6, 24]. However, scheduling problems become increasingly complex with the growing number of services and requests, leading to a rapid increase in the number of possible solutions. Due to the exponential growth of feasible states, it becomes impractical to evaluate all possibilities to determine the best scheduling exhaustively, resulting in these problems falling under the NP-Hard category that deterministic methods cannot be used in solving this category of problems due to their time-consuming nature, and meta-heuristic methods should be developed to solve these problems properly [2, 25]. Meta-heuristic methods have been the attention of researchers due to their simplicity, flexibility, no need for derivation, and escape from local optima. The No Free Lunch (NFL) theorem establishes that no meta-heuristic algorithm can solve all optimization problems perfectly. In other words, an algorithm that performs well on a set of problems may not yield favorable outcomes for another set of problems [3, 26]. Taking this into account, this study introduces an advanced strategy based on the cat swarm optimization (CSO) algorithm to achieve optimal scheduling in cloud infrastructure. The primary objective is to significantly reduce energy consumption in cloud data centers by effectively allocating tasks to processing servers, thereby preventing server overload or underload. The following section presents a literature review on scheduling and load balancing, followed by an examination of the proposed technique in section 3. Finally, in section 4, the proposed technique is implemented in the iFogSim simulator, and the results are evaluated.

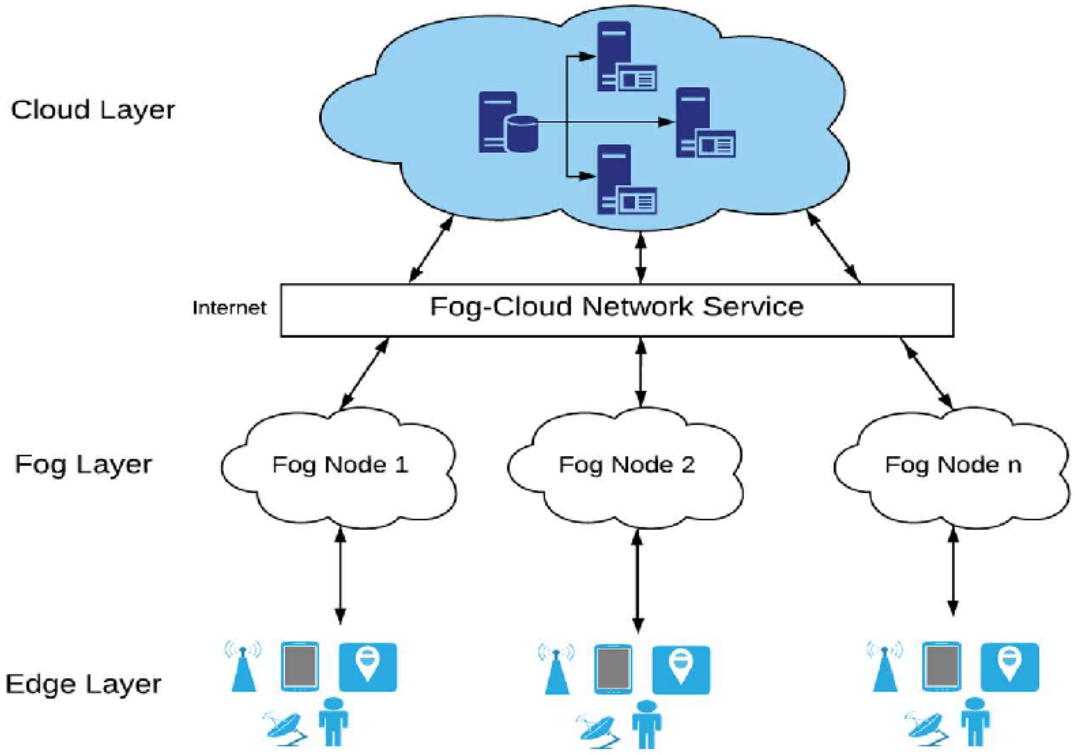

Fig. 1. Fog computing architecture.

A. *Fog Computing*

As illustrated in Fig. 1, fog computing operates by conducting local processing and storage of IoT data on IoT devices rather than transmitting the data to the cloud. In contrast to the cloud, fog computing offers faster response times and improved quality, making it an optimal choice for enabling the Internet of Things. It is known for providing efficient and secure services to a wide range of users [4, 27]. While the cloud serves as an intermediary between endpoint clients and cloud computing, fog computing occurs closer to the edge of the cloud and end devices, resulting in significantly lower latency. Cloud computing services are located on the Internet, whereas fog computing operates at the edge of the local network. Real-time interactions are supported by both cloud and fog computing, but processing data and applications in the cloud can be time-consuming, particularly for large-scale data. Fog computing facilitates centralized resource management, including the allocation of computing, networking, and storage resources. The primary objective of fog computing is to equip network edges and network devices with virtual services for processing, storage, and network provisioning [5, 28]. Table I provides a comparison of the similarities and differences between fog computing and cloud computing.

TABLE I. DIFFERENCES BETWEEN CLOUD AND FOG INFRASTRUCTURE

Ability	Fog Computing	Cloud Computing
Delay	Low	Up (depending on the user's path to DC)
Response time	milliseconds	several minutes
Service location	Network edge	In cloud data centers
Data storage period	transitory	months or years (according to the contract)
Steps between user and server	one step	Several steps
Aware of location	Very local	no
architecture	Distributed	concentrated
Type of communication	wireless	Broadband, MPLS
Possibility of data recording	Low	High
End-to-end security	can be defined	It cannot be defined or controlled
Data collection nodes	unlimited	Very low
Mobility support	supports	Limited support

Fog computing offers significant advantages by reducing data transfer, traffic, and distance traveled. It involves decentralizing computations to the edge of the network, where downstream data processing occurs in cloud services and upstream data processing occurs in Internet of Things (IoT) services [6, 29]. In cloud computing, information storage and processing are directed towards the network edge and closer to the source of information generation. Instead of transmitting IoT-generated data to distant data centers or remote servers for storage and processing, this data is stored on local servers and storage devices through a local gateway. This approach enhances the speed of information analysis and alleviates network congestion [7, 30]. The concept behind fog computing is that computation should be performed in close proximity to data sources. It has the potential to impact society as significantly as cloud computing. Fog computing offers numerous advantages over traditional architectures, particularly optimizing resource utilization within a cloud computing system. By conducting computations at the network edge, fog computing reduces network traffic. Essentially, edge computing operates within the cloud but in proximity to objects interacting with IoT data. As depicted in Fig. 1, fog computing acts as an intermediary between the cloud and end devices, bringing processing, storage, and network services closer to the end devices. These devices, known as edge nodes, can be deployed anywhere with a network connection. In essence, fog computing extends cloud infrastructure to the network edge. While fog and cloud computing share network, computing, and storage resources, they utilize similar mechanisms and features such as virtualization and multi-tenancy [8, 31].

II. Related Work

The study in [9] focuses on addressing concerns related to scheduling and resource allocation within fog computing for various types of applications. In this research, it has been stated that different programs composed of a collection of interrelated services are mapped to cloud computing for processing, but the placement of services has its challenges. Accordingly, this article presents a distributed placement strategy, the main goal of which is to reduce energy consumption and the cost of communication. The proposed method is based on game theory, which uses iterative combinatorial auction (ICA). The proposed solution through game theory makes significant changes in the ICA method. Based on this, the proposed solution is decentralized and is able to interact between fog nodes and applications so that decisions related to placement are made in each round. The results of the evaluations show the optimality of the solution in reducing the cost of communication and increasing the productivity of the fog node processor, which is an important advantage considering the limited resources in the fog nodes. In [10], a solution for autonomous scheduling of services on devices on the edge of the network is presented. In this solution, the dynamic capability of programs based on microservices is used in order to provide an adaptive solution for placement. The main objective of this solution is to minimize the response time of services, ultimately enhancing the quality of user experience, particularly in critical services. In this method, the decision about the placement of the service is made based on the response time of the applications. Various functions have been used for this purpose. Then, through the use of a meta-heuristic algorithm based on PSO, a decision is made regarding the placement of the service. The first input parameter for the algorithm is the graph call related to the application service.

In the end, through a series of evaluations, it was shown that the proposed solution was able to perform the placement of services and microservices in such a way that the response time is minimized. The authors in [11, 32] have presented a lightweight framework for providing fog computing services to be used by IoT devices that are sensitive to delay, and their needs must be answered in real-time. The FogPlan solution is a QoS-aware proposal, and the placement of services in it is done according to the network status. This framework works with minimal assumptions and information about IoT nodes. Then a probabilistic formula for the optimization problem, along with two heuristic algorithms, has been presented to answer the QoS needs. By using this solution, in addition to more accurate positioning, the amount of delay and network cost can also be reduced. In the evaluations, it has been concluded that the solution has reduced the average delay and the scheduling costs through the optimal scheduling of services.

Research [12] introduces a task scheduling policy in fog computing that is based on graph partitions. The main objective of this solution is to enhance accessibility. By employing this method, the quality-of-service delivery is enhanced through program prioritization across all cloud devices and transferring services to these devices. The main idea is based on the premise that the more necessary services are provided near the user, the quality and delay of service delivery can be reduced. Based on this, with the help of graphic partitions, the desired services and related tasks are determined and based on the number of available resources, the best possible placement is done. In the end, the proposed solution has been evaluated with a linear programming approach, and the results indicate an improvement in accessibility and service delivery. The work described in [13] focuses on scheduling application components within hybrid cloud and fog systems based on network function virtualization (NFV) principles. This study investigates the main challenges of deploying components related to cloud and fog applications in NFV. In fact, these challenges are mostly expressed due to the heterogeneity between the components. To solve this problem in this research, a model based on four hierarchies 1) sequence, 2) parallel, 3) selection and 4) compliance loop was used. In this solution, it is assumed that the program components are implemented as VFSs, and the structural diagrams represent an application program that follows the proposed hierarchical structure. In this structure, each VFN becomes a tree, where the leaf nodes represent the program components, and the costs of the model are calculated by gathering the information of the nodes from bottom to top. The main goal of this method is to minimize the processing costs, which are modeled using an ILP integer programmer. The authors of [14] propose a concept to address the challenge of deploying Internet of Things (IoT) services within the fog computing environment. They model the deployment problem of IoT applications on resources as an optimization problem. The concept of a "fog colony" is introduced, and the coordinated fog cell colony and deployment approach are presented as a solution for deploying

services on virtual resources within the fog domain. The optimization problem primarily focuses on the deployment and execution time of the application, aiming to maximize resource utilization in the fog through a greedy heuristic method based on the genetic algorithm. In this model, the communication and subscription of fog cell services are done by the control node. Also, the communication between the cells and the control node with the cloud is made possible by the fog computing management system.

In the end, the results of this research have been compared with the classical approach that executes all tasks in the cloud. The intended evaluations have been carried out in the iFogSim simulator environment, and the results indicate that the optimization method has a favorable effect on the use of fog resources, and the genetic algorithm has a lower delay for execution compared to the cloud. In [15], a service placement algorithm is presented for efficient use of the network and improvement of energy consumption. This algorithm sequentially allocates application modules with the highest needs to the nodes with the highest capacities. To test the proposed algorithms, the program has been implemented on network topologies named JSON. The scenario has been implemented with more than three different network topologies and workloads. The proposed cloud-fog placement and deployment method were evaluated by comparing it with the traditional cloud-based deployment approach. The evaluation considered program delay (response time), network usage, and energy consumption as the primary metrics. Data was collected from the iFoSim simulator using the proposed placement and resource deployment methods. The results demonstrate the favorable impact of the proposed placement approach across all three topologies, showcasing improvements in network utilization, program delay (response time), and energy consumption compared to the traditional cloud-based approach. The authors in [16] presented a tool called FogTorch. Accordingly, the first part of the article presents a model for the use of service quality systems in multi-sector applications of the Internet of Things for fog architecture. In the second part, the results of using Monte Carlo simulations for the FogTroch tool are stated, and the application of this tool is examined in terms of service quality and resource consumption.

FogTroch is a tool that enables the simulation and comparison of different fog scenarios in the design phase, as well as the resource and quality-of-service (QoS) aware deployment of IoT applications through cloud-fog architecture. It takes into account various processing resources such as CPU, RAM, storage, and software, along with QoS constraints like latency and bandwidth, which are essential for real-time fog applications. Notably, FogTroch is the first tool capable of estimating the quality of service resulting from fog-based application deployments based on probability distribution models of bandwidth and delay provided by communication links. Additionally, it provides estimates of resource consumption within the fog layer, allowing for the optimization of resource utilization among different fog nodes. In [17], a dynamic module is presented to schedule the tasks of Internet of Things applications in edge and cloud computing. Through this solution, the problem related to IoT requests in a heterogeneous network environment based on edge cloud has been solved. As a result, a mapping between the application module and the main resources of the devices can be created. In this way, problems related to the delay of tasks and energy consumption can be overcome. To achieve this objective, the application employs a dynamic discovery algorithm that enables the step-by-step execution of operations in the fog computing environment. This algorithm helps to reduce the delay in task execution by efficiently discovering and allocating resources. Through simulations conducted in the iFogSim environment, the results demonstrate the remarkable service quality of the applications, accompanied by a significant reduction in energy consumption compared to other scheduling strategies. The combination of the dynamic discovery algorithm and the fog computing environment contributes to improved application performance and energy efficiency.

After conducting a thorough examination of the aforementioned studies, we have identified their strengths and limitations in several aspects:

- Studies, such as [9], [13], [14] have provided comprehensive introductions to the theoretical concepts and platforms of fog computing. They have significantly contributed to the research on optimization strategies in this field.
- Several existing studies, like [10]-[12] and [15]-[17] have focused on optimizing the trade-off between time and energy consumption in fog computing scenarios. However, one critical aspect they often overlook is the consideration of task characteristics and resource types concerning the problem of balancing delay and device load. This omission can lead to inefficient resource allocation and potential wastage of resources.

Our research falls within the category of optimizing Quality of Service (QoS) and energy consumption simultaneously. It complements and extends existing works in several ways. For instance, unlike the study in [10] where multiple user devices share one Multi-access Edge Computing (MEC) server, or [11] where a single device generates a task, our approach considers a more complex system involving multiple users and multiple fog nodes. Additionally, our focus is on reducing the energy consumption of the fog nodes themselves, which sets us apart from the majority of existing approaches that primarily aim to minimize energy consumption on the mobile devices [9,11,12,16,17]. Furthermore, we present a novel approach that jointly minimizes the overall energy consumption and the execution time while taking into account the computation resource constraints of fog devices. This approach offers a comprehensive optimization strategy for the system.

III. PROPOSED METHOD

The primary objective of the scheduling solution is to efficiently allocate n tasks to m machines, where $n > m$, with the aim of minimizing both the execution time and energy consumption. To achieve this objective, it is crucial to compute the resource availability of all machines. An upper bound is established for the minimization function, which is defined as $m \times L_{max}$. Here, L_{max} represents the maximum execution time

for each node. Thus, for any task scheduling completion time G, it should satisfy the condition $m \times L_{max} > G$. In other words, the execution of the assigned tasks will take at least as long as L_{max} for each container. Let's consider S = (S1, S2, ..., Sn) as the set of user tasks given to the system. During the scheduling process, these user tasks are assigned to available fog nodes. Each node is equipped with $\{PE_1, PE_2, ..., PE_m\}$ processing elements to handle service tasks. Each processing element has a distinct processing speed characteristic denoted as PE_S and the CSO (Cat Swarm Optimization) algorithm [18, 33] is used to allocate resources and schedule tasks in a fog infrastructure effectively.

The CSO algorithm models the two primary behaviors of cats, known as tracking and searching modes, using sub-models. By combining these modes in a defined ratio, the cat crowding optimization algorithm exhibits strong performance. Similar to particle swarm optimization, the positions of the cats serve as solutions, and the algorithm utilizes the cats' behavior to solve optimization problems. In the cat swarm optimization, the number of cats to be used is determined, and each cat possesses a position with M dimensions. Additionally, each cat has a speed for each dimension and a fitness value indicating its level of fitness. This fitness value is obtained using a fitness function. Furthermore, each cat is assigned a flag to identify whether it is in tracking or seeking mode [19, 34].

The aim of this research is to minimize energy consumption and reduce the execution time of user requests in the cloud infrastructure. To achieve this, the cat swarm optimization algorithm is modified to enable the effective allocation of resources to tasks. The modified algorithm optimizes the resource allocation process, leading to improved energy efficiency and reduced request execution time. By leveraging this modified cat swarm optimization algorithm, the cloud infrastructure can allocate resources more effectively, resulting in enhanced performance and reduced energy consumption. In the proposed solution, a set of cats will be used; some of which are in search mode and at the same time, some others are in tracking mode. In this method, each cat represents a specific task-resource mapping. The cats are updated based on their current state, and the goal is to minimize the cost of mapping. The fitness value is assigned to each cat, reflecting the quality of its mapping solution. In each iteration of the algorithm, a new set of cats is selected to be in the search state, where they explore different mappings. Ultimately, the cat with the best fitness value represents the best mapping solution, which results in the lowest cost among all the possible mappings. By iteratively updating the cats and selecting the best solutions, the algorithm aims to find an optimal task-resource mapping that minimizes cost and improves overall efficiency. Accordingly, in the following, the search and tracking modes are examined as a sub-model to model the cat's search behavior to find the desired target: processing servers in this research.

A. Seeking Mode

During the search process in the SM (Seeking Mode) of the algorithm, the exploration of different regions in the search space is conducted. However, the search is limited to the local vicinity of the current position of the seeking cat. This approach focuses on refining existing solutions rather than exploring distant areas of the search space. Fig. 2 illustrates this local search behavior. There are four main factors considered in the seeking mode of each cat:

1) Search Memory Source (SMP): This factor determines the size of the search memory for each cat, representing the positions previously searched by the cat. Based on the fitness functions, the cat chooses one position from its memory as a potential candidate for movement.

2) Number of Dimensions that Change (CDC): This factor determines the number of dimensions in the current position of the cat that will be modified during the search process.

3) Search in the Defined Range of Dimensions (SRD): This factor indicates the rate of change for the selected dimensions. If a dimension is chosen for modification, the difference between the new and old values will be within the range defined by SRD.

4) Attention to the Current Position (SPC): This factor is a Boolean variable that determines whether the current position of the cat is considered a candidate for movement or not. It ensures that the current position, which has already been explored, is not included in the search memory (SMP).

By considering these factors, the seeking mode of the cat swarm optimization algorithm focuses on local search, efficiently exploring and refining solutions within the vicinity of the current positions of the cats.

The seeking mode algorithm, as depicted in Fig. 3, outlines the steps followed when a cat is in seeking mode [35]. The algorithm proceeds as follows:

1) Create j copies of the current position of cat k (Cat_k), where j is equal to SMP (Search Memory Source). If the value of SPC (Attention to the position) is true, then j is adjusted to (SMP-1), allowing the current position to be considered one of the candidates.

2) For each copy, based on the CDC (Number of Dimensions that Change), increase the SRD (Search in the defined range of dimensions) by a percentage of the current values and add it to the previous values. This step determines the range within which the selected dimensions can change during the search.

3) Calculate the fitness function (FS) for each candidate point generated in the previous step.

4) If not, all FS values are exactly equal, proceed to calculate the probability of selecting each candidate position based on the normalized fitness (fit) of that position. If all FS values are equal, set the probability of selecting all candidate positions to be equal.

By following this algorithm, the seeking mode effectively explores the search space by creating multiple copies of the current position, determining the range of changes in dimensions, evaluating the fitness for each candidate point, and selecting the next position based on the calculated probabilities.

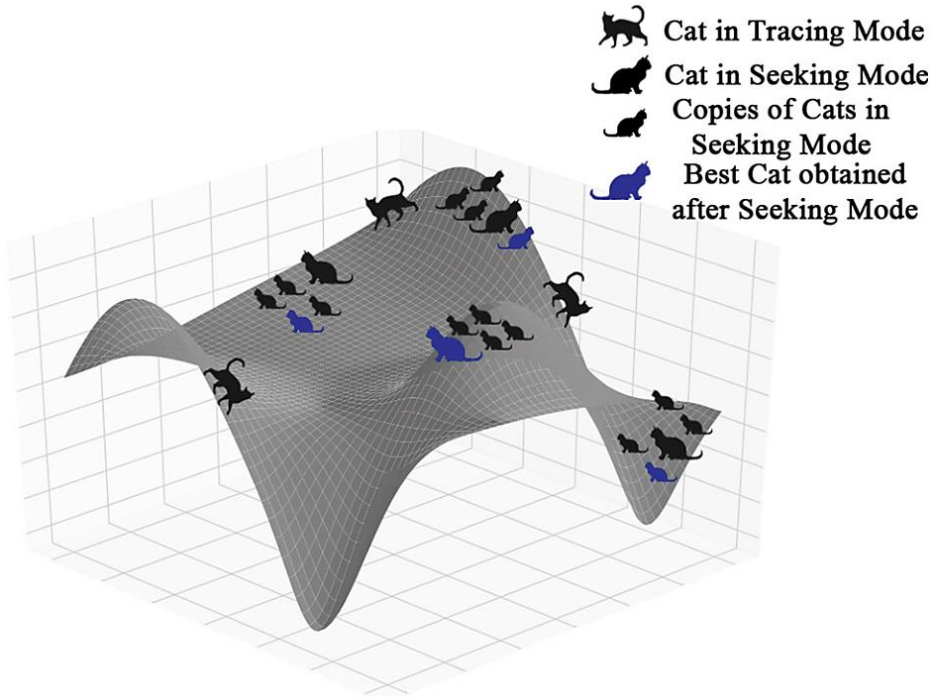

Fig. 2. Seeking mode.

- Create j copies of the kth cat, represented by SMP (Search Memory Source).
- Modify the CDC (Number of Dimensions that Change) dimensions of each copy randomly, introducing variations in the solution space.
- Evaluate the fitness of each copy by assessing its cost or quality based on the fitness function.
- Identify the best solutions among all the copies, which correspond to the mappings with the minimum cost.
- Randomly select one solution from the best solutions and replace it with the current position of the ith cat, updating its state.

Fig. 3. Seeking mode algorithm.

B. Tracing State

The tracking mode algorithm, which describes the behavior of the cat when tracking the target, can be outlined in the following three steps:

1) Calculate the new position of the cat based on its current position and speed in each dimension. The cat moves in the search space according to its velocity, exploring different locations.

2) Evaluate the fitness of the new position using the fitness function. This assesses the quality or cost associated with the new mapping.

3) Update the cat's position to the best position found during the tracking process. The cat moves towards the position that yields the minimum cost or maximum quality, improving its mapping.

By following these steps, the cat in tracking mode continuously adjusts its position based on its velocity, evaluates the fitness of the new position, and updates its location to the best position encountered during the tracking process. This allows the cat to gradually converge towards an optimal solution by iteratively exploring and refining its mappings.

In the first step, the speed synchronization for each dimension (vk, d) is calculated according to the following relationship:

$$v_{k,d} = v_{k,d} + r_1 \times c_1 \times (x_{best,d} - x_{k,d}) \quad , \quad d = 1,2,\dots,M$$

$x_{best,d}$ shows the position of the cat that has the highest value of fitness and $x_{k,d}$ is the position of the kth cat. c_1 is a fixed number, and r1 is a random number in the interval [0,1]. In the second step, it is checked that the speeds are within the defined range. If there were a higher speed, it would be replaced with the maximum possible value in the desired range, and finally, in the third step, the position of the cat will update according to the following relationship:

$$x_{k,d} = x_{k,d} + k_{k,d}$$

In fact, TS is for Teaching-Learning-Based Optimization. During this phase, the cats aim to exploit the information about the best position found so far to reach the optimal solution. Even though a cat's position change may be large during this phase, the search is still focused on the best solution found so far. Please refer to Fig. 4 for an illustration.

In other words, at this stage, the set of answers obtained from the best location of the cat in the current iteration is updated. As seen, cat swarm optimization uses two sub-models of seek and tracking mode; the way to combine these two sub-models to perform scheduling operations is described in the next section.

C. Scheduling

The optimal utilization of available resources is the main goal of task scheduling that provides a basis for load balancing. In order to reduce the cost of services provided to users, fog service providers adopt different policies depending on the type of user and the desired services. However, the aim of this study is to reduce the resource consumption cost. Thus, parameters such as energy consumption and traffic consumption are considered. These parameters are formulated based on the following equations to be used in the CSO-based approach.

D. Modelling Energy Consumption

In order to model the amount of energy consumed, the proposed approach in [20, 36, 40] is used. The idea of this model is based on the fact that there is a linear relationship between processor efficiency and power consumption. In other words, if we know the processing time of a task and the efficiency of the processor, we can calculate the energy consumed by that task. This means that by understanding how long a task takes to complete and how efficiently the processor performs it, we can determine the energy usage associated with that task.

The efficiency of a given resource r_j in a given time is calculated as follows.

$$U_j = \sum_{j=1}^{n} u_{i,j} \quad (1)$$

Here, n represents the number of tasks currently in progress, and u_{ij} represents the amount of resources consumed by task t_j. Accordingly, the energy consumption (E_j) of resource r_j in a given time can be calculated using the following formula:

$$E_j = (P_{max} - P_{min}) \times U_j + P_{min} \quad (2)$$

E. Traffic Consumption

This section aims to model the network's bandwidth usage and the volume of traffic produced by individual physical machines.

$$D_j = \sum_{\forall m \epsilon V_j} \lambda(j,m) \sum_{i=1}^{\rho(j,m)} C_i \quad (3)$$

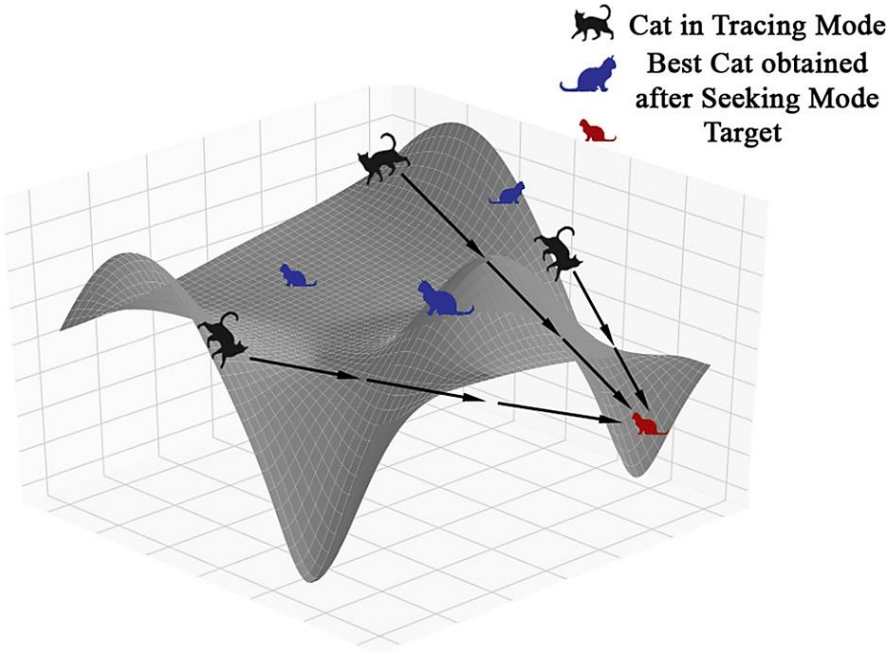

Fig. 4. Tracing state.

D_j represents the communication between physical machine j and other physical machines. λ(j, m) denotes the traffic load between physical machines j and m. V_j indicates a set of physical machines connected to physical machine j. C_i represents the weight of the communication link between two physical machines at level i. ρ(m, j) represents the communication level between physical machines m and j.

The multi-objective cost function for modeling the fog computing environment is formulated based on these two parameters as follows:

$$\text{Minimize} \quad \sum_{j=1}^{m} E_j \tag{4}$$

$$\text{Minimize} \quad \sum_{j=1}^{m} D_j \tag{5}$$

The minimum cost for task scheduling is obtained using the above functions. In the following, it is described how to combine two states tracking and seeking:

1) The initial population consists n cats.

2) The cats in this scenario are distributed randomly within a d-dimensional search space by assigning each cat a random speed for each dimension within a given range.

3) A number of cats are randomly selected. According to the Mixture Ratio (MR), some cats are put in the searching state, and the remaining ones are considered for the tracking state.

4) The movement of cats is based on their flag value which shows the tracking or seeking states.

5) The fitness value of each cat is evaluated, and the position of the best cat (Xbest) is recorded and stored.

6) The situation of cats is updated according to their position. Then, step 3 is repeated.

7) Steps 5 and 6 are repeated iteratively until the termination criterion is met. i.e. a complete task scheduling is reached.

The pseudo-code for scheduling based on CSO is depicted in Fig. 5.

It is important to note that while executing scheduled commands on processing servers using cat swarm optimization, it is possible for some servers to fail to execute the commands or experience additional traffic and load due to delays in processing previous commands or hardware and software errors. Meanwhile, some servers may remain idle after completing their assigned commands. To address this, the resources of processing servers are periodically monitored, and the following three conditions are used to determine the status of their resources:

- If the resource consumption of a server has exceeded 95% of its maximum efficiency, it is categorized as an overloaded machine.
- If the resource consumption of a server is lower than the average efficiency determined (usually in the range of 10%), it is categorized as a server with a low load.
- Servers that do not meet the above conditions are considered normal servers.

```
Population of cats, Ci (i = 1, 2, ..., n)
Initialize number of iterations, MR, SPC
While
    → Calculate resource wastage and traffic cost using equation 2 & 3
    → Ci = the cat with best solution
    → → For = 1 : N
    → → → If seeking mode = true
    → → → → goto seeking mode
    → → → → Else
    → → → → goto tracing mode
    → → → End if
    → → End for i
End while

End
```

Fig. 5. Pseudo-code of algorithm.

Now to balance the load, the following load-balancing strategies can be applied:

- Servers in overloaded mode: Loads of these servers can be migrated to other servers in the P_{under} (low load) or normal mode. This can be achieved through the virtual machine migration technique, where the tasks and loads are transferred to another server that has sufficient resources to accommodate them.
- Servers with an efficiency below 10%: These servers can be considered candidates for task migration and subsequent shutdown. By migrating their tasks to other servers, the load on the underutilized servers can be increased, and energy consumption can be reduced by shutting down the inefficient servers.

By applying these load-balancing strategies, the goal is to optimize resource utilization, minimize overload conditions, and reduce energy consumption in the system.

IV. EVALUATION

The implementation and evaluation of the proposed approach require a set of programming tools. These tools are necessary and useful for implementing different scenarios and evaluating the results. The comparison of the proposed approach with other existing techniques can also be made using these tools. For this purpose, the iFogSim2 simulator is used in this study [21, 37]. In order to evaluate the performance of the proposed approach, it is compared with metaheuristic algorithms such as PSO, ACO, Genetic (GA) and Random (RND) scheduling algorithms. In this study, the workload of simulations in the real traced data of the CoMon project is used, which is a monitoring infrastructure for PlanetLab [22, 38]. The main attributes of the cat swarm optimization algorithm are illustrated in Table II.

A. Evaluation Results

The outcomes of the assessment are depicted in Fig. 6 to 10. The initial evaluation encompassed three distinct experiments involving varying numbers of virtual machines. The objective was to explore how algorithms operate with diverse virtual resources and examine the correlation between resource quantity, execution time, and energy consumption. The subsequent results are presented below. Fig. 6 illustrates the energy consumption for all three solutions corresponding to the number of available virtual resources.

TABLE II. MAIN CHARACTERISTICS OF THE CSO ALGORITHM

Decision-making variables	The position of cats in each dimension
Solution	The position of cats
The previous solution	Previous position of cats
New solution	The new position of cats
Better answer	Any cats exhibit the highest level of fitness.
Initialization of answers	Random location of cats
How to create a new answer	Use tracking and seeking modes.

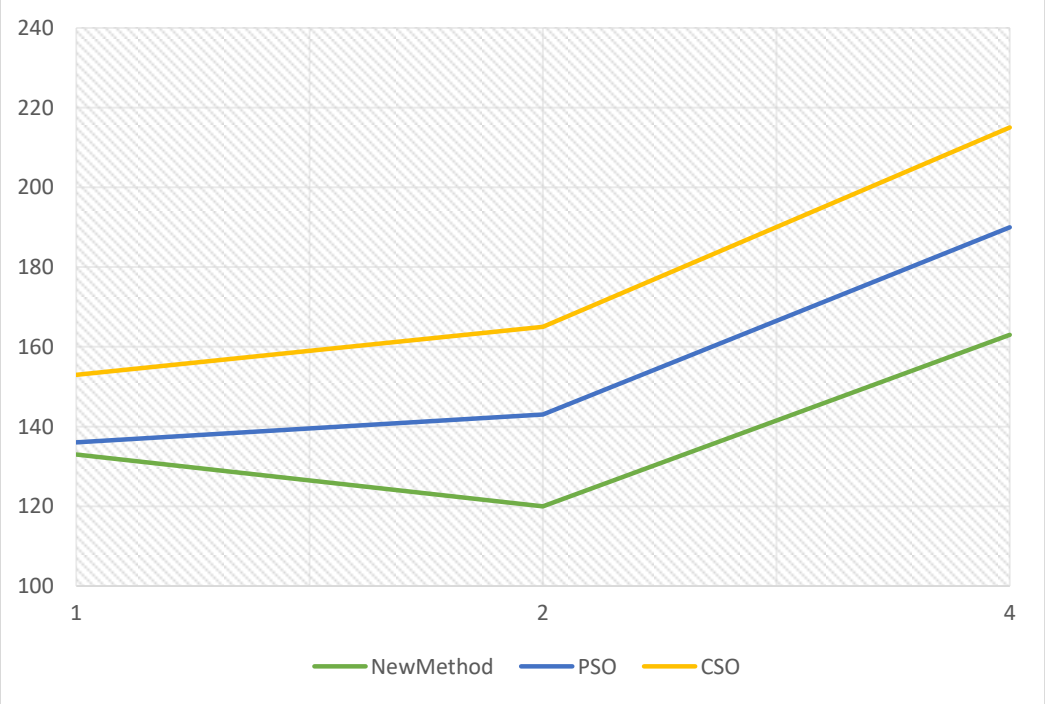

Fig. 6. Energy consumption in different tests.

As depicted in Fig. 6, an increase in the number of virtual machines leads to a rise in energy consumption across all three solutions. However, the proposed method consistently exhibits lower energy consumption at each stage. Although the optimality rate was initially low in the first test, it improved in subsequent evaluations as the number of available resources increased. This improvement was facilitated by the optimal arrangement of virtual machines achieved through CSO scheduling. Consequently, the proposed solution outperforms other methods with a significantly higher optimality rate. Moving forward, Fig. 7 evaluates the implementation time of the solutions in three distinct modes.

As depicted in Fig. 7, the execution time of the solutions increases with the growth in the number of virtual machines. However, the proposed method effectively addresses this issue through optimal resource allocation, the precise arrangement of virtual machines, and achieving load balancing. Consequently, the proposed solution outperforms the others by completing tasks in a shorter period in all three tests. The optimization carried out with the assistance of the CSO algorithm enables the identification of the best available nodes, resulting in the allocation of resources to the most suitable machines. As a result, both the execution time and energy consumption are reduced.

Fig. 7. Execution time of solutions in different experiments.

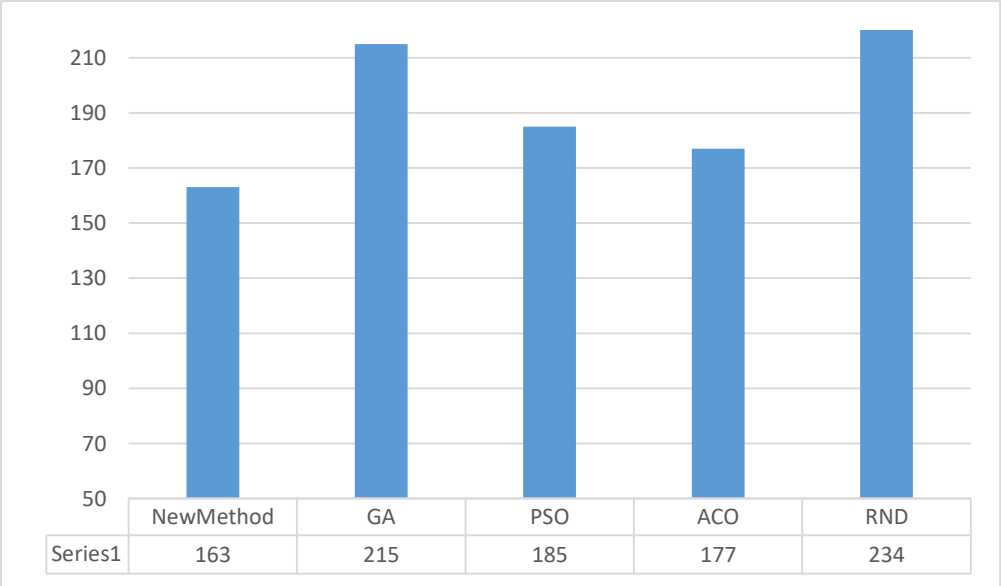

Fig. 8. Energy consumption (KW/hr).

The next test was done with a constant amount of virtual and available resources, as well as applying a workload based on the CoMon project [38, 39]. The results of the evaluation of all algorithms are shown below. First, in Fig. 8, the amount of energy consumption of the proposed solution is shown in comparison with the other solutions.

Fig. 8 clearly illustrates a significant reduction in energy consumption within the proposed solution compared to other algorithms. This achievement can be attributed to the allocation and optimal arrangement of resources facilitated by the enhanced cat swarm optimization algorithm. In the proposed solution, the process of identifying available nodes has been optimized using the algorithm, and resource allocation is performed based on the status of these available resources. Consequently, energy consumption is minimized through the creation of optimal scheduling, alongside a reduction in execution time.

By adopting the proposed approach, notable improvements can be observed. For instance, in comparison to the RND-based solution, energy consumption has been reduced by approximately 24%. Similarly, compared to the PSO algorithm, the proposed solution demonstrates a reduction of approximately 10% in energy consumption. These results highlight the effectiveness and efficiency of the proposed method in optimizing energy consumption. Due to the fact that the amount of available resources is checked in every resource allocation operation, this energy reduction was predictable. In the following Fig. 9, the implementation time of the solutions is shown.

Indeed, in fog computing, minimizing the execution time of requests is crucial alongside reducing energy consumption. As depicted in the diagram in Fig. 9, the proposed solution exhibits a remarkable reduction in execution time compared to alternative approaches. Specifically, compared to schedules based on the PSO and ACO algorithms, the proposed solution achieves a reduction of approximately 21%. Furthermore, in comparison to the RND-based schedule, the execution time is reduced by approximately 33%. These results highlight the significant gains in efficiency and speed achieved through the proposed solution, further enhancing the benefits of fog computing.

This significant reduction in execution time can be attributed to the scheduling and optimal resource arrangement, as well as the establishment of load balancing through the proposed solution based on the cat swarm optimization algorithm. The solution efficiently utilizes the processing servers' resources by considering their status and allocating resources accordingly. Additionally, Fig. 10 presents the level of violation of the service level agreement (SLA).

The service level agreement (SLA) serves as the framework for establishing the expected level of service. It defines the formal conditions for the provided service, including aspects such as performance and availability. Fig. 10 illustrates that the proposed solution, through effective load balancing, enhances the reliability and availability of servers within the fog computing infrastructure compared to alternative solutions. In other words, the proposed solution exhibits a lower percentage of SLA violations, indicating its superior ability to meet the agreed-upon service level requirements.

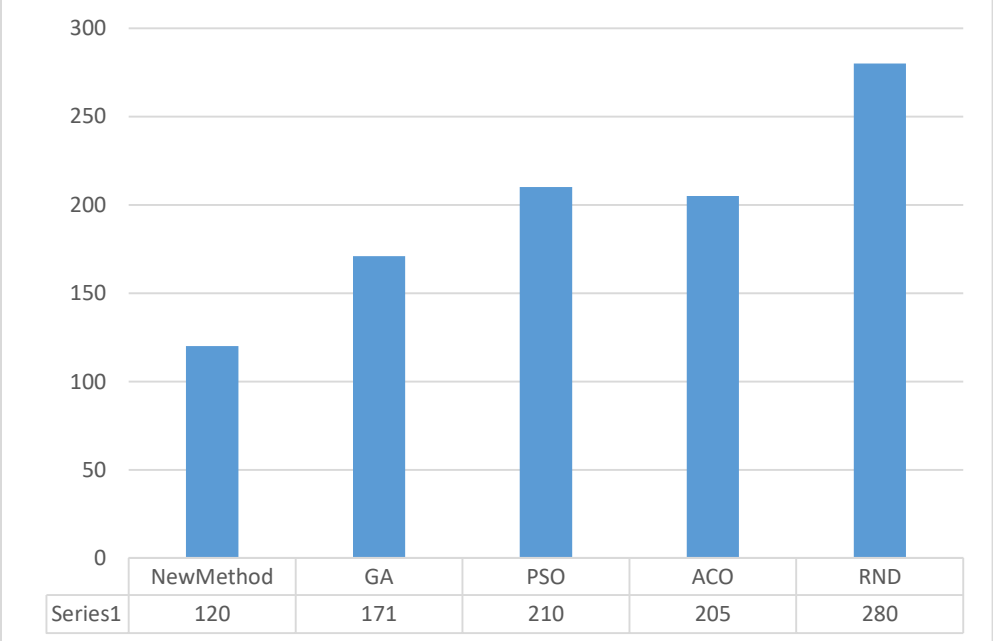

Fig. 9. Execution time (millisecond).

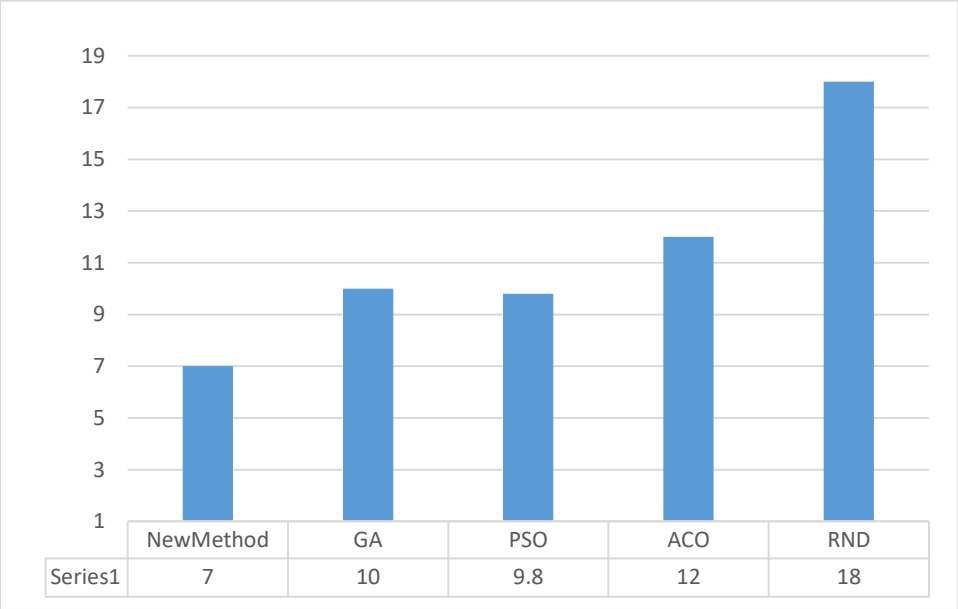

Fig. 10. Percentage of Service Level Agreement (%SLA) violation.

V. Conclusion

This research introduces an approach based on the CSO algorithm to achieve optimal scheduling and load balancing. The proposed technique aims to prevent server overload or underload by efficiently allocating tasks to physical servers. To enhance the performance further, the CSO algorithm is extended by incorporating new parameters, including the amount of available resources and network traffic. By considering these factors, the proposed approach minimizes the execution time of requests by appropriately distributing tasks among candidate servers. Finally, the proposed approach is implemented and simulated in iFogSim, allowing for a comparison with other algorithms such as PSO, ACO, GA, and Random. The results obtained from the simulation demonstrate that the proposed approach successfully achieves its objectives and surpasses the other methods in terms of execution time and energy consumption. This highlights the effectiveness and superiority of the proposed approach in optimizing these important performance metrics. It could improve the energy consumption compared to ACO and PSO by the values of 24% and 10%, respectively.

Additionally, the execution time is significantly reduced, and in compared to ACO and PSO, the proposed approach could improve the execution time by 21% and 22%, respectively. The observed improvement can be attributed to the load balancing achieved in the fog environment through the combined utilization of the CSO-based scheduling method and virtual machine migration technique. This combination effectively distributes tasks and resources across the fog network, ensuring optimal utilization and minimizing resource imbalances. As a result, the proposed approach enhances load balancing, leading to improved performance in terms of execution time and energy consumption. The results also show that the proposed approach has less violations in SLA when compared to the other algorithms and consequently provides more reliable servers with higher availability in the fog infrastructure.

VI. Future Work

The proposed approach can be applied to Content Delivery Networks (CDNs). Therefore, the CSO-based approach enables the identification and replication of the best content within alternative servers. By leveraging the capabilities of the CSO algorithm, the proposed approach effectively identifies the optimal servers to store and replicate content. This ensures that content is efficiently distributed across multiple servers, enhancing availability, fault tolerance, and overall system performance. In fact, by generalizing this approach and using the tracking mode of cats, the best contents can be obtained to be cached and replicated in servers found in the seeking mode. As a result, the performance of such cloud servers can greatly improve.

References

[1] Cleveland, S. M., & Haddara, M. (2023). Internet of Things for Diabetics: Identifying Adoption Issues. Internet of Things, 100798.

[2] Hazra, A., Rana, P., Adhikari, M., & Amgoth, T. (2022). Fog computing for next-generation internet of things: fundamental, state-of-the-art and research challenges. Computer Science Review, 48, 100549.

[3] Saad, Z. M., & Mhmood, M. R. (2023). Fog computing system for internet of things: Survey. Texas Journal of Engineering and Technology, 16, 1-10.

[4] Vambe, W. T. (2023). Fog Computing Quality of Experience: Review and Open Challenges. International Journal of Fog Computing (IJFC), 6(1), 1-16.

[5] Das, R., & Inuwa, M. M. (2023). A review on fog computing: issues, characteristics, challenges, and potential applications. Telematics and Informatics Reports, 100049.

[6] Royer, C. W. (2022). Fog Computing in Optically Networked Space Constellations. IEEE Aerospace and Electronic Systems Magazine.

[7] Yadav, A. M., Tripathi, K. N., & Sharma, S. C. (2023). An opposition-based hybrid evolutionary approach for task scheduling in fog computing network. Arabian Journal for Science and Engineering, 48(2), 1547-1562.

[8] Goudarzi, M., Palaniswami, M., & Buyya, R. (2022). Scheduling IoT applications in edge and fog computing environments: a taxonomy and future directions. ACM Computing Surveys, 55(7), 1-41.

[9] Kayal, P., & Liebeherr, J. (2019, July). Distributed placement in fog computing: An iterative combinatorial auction approach. In 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS) (pp. 2145-2156). IEEE.

[10] Alsmadi, A. M., Ali Aloglah, R. M., Smadi, A. A., Alshabanah, M., Alrajhi, D., Alkhaldi, H., & Alsmadi, M. K. (2021). Fog computing scheduling algorithm for smart city. International Journal of Electrical & Computer Engineering (2088-8708), 11(3).

[11] Upadhyay, G. M., & Gupta, S. (2022). A Study on Optimal Framework with Fog Computing for Smart City. Smart IoT for Research and Industry, 133-143.

[12] Chen Cao, Jianhua Wang, Devin Kwok, Zilong Zhang, Feifei Cui, Da Zhao, Mulin Jun Li, Quan Zou. webTWAS: a resource for disease candidate susceptibility genes identified by transcriptome-wide association study. Nucleic Acids Research.2022, 50(D1): D1123-D1130.

[13] Ning Xu, Zhongyu Chen, Ben Niu, and Xudong Zhao. Event-Triggered Distributed Consensus Tracking for Nonlinear Multi-Agent Systems: A Minimal Approximation Approach, IEEE Journal on Emerging and Selected Topics in Circuits and Systems, DOI: 10.1109/JETCAS.2023.3277544, 2023.

[14] Samiei, M., Hassani, A., Sarspy, S., Komari, I. E., Trik, M., & Hassanpour, F. (2023). Classification of skin cancer stages using a AHP fuzzy technique within the context of big data healthcare. Journal of Cancer Research and Clinical Oncology, 1-15.

[15] Sun, J., Zhang, Y., & Trik, M. (2022). PBPHS: a profile-based predictive handover strategy for 5G networks. Cybernetics and Systems,53(6), 1-22.

[16] Khezri, E., Zeinali, E., & Sargolzaey, H. (2022). A novel highway routing protocol in vehicular ad hoc networks using VMaSC-LTE and DBA-MAC protocols. Wireless Communications and Mobile Computing, 2022.

[17] Trik, M., Akhavan, H., Bidgoli, A. M., Molk, A. M. N. G., Vashani, H., & Mozaffari, S. P. (2023). A new adaptive selection strategy for reducing latency in networks on chip. Integration, 89, 9-24.

[18] Haoyu Zhang, Quan Zou, Ying Ju, Chenggang Song, Dong Chen. Distance-based Support Vector Machine to Predict DNA N6-methyladine Modification. Current Bioinformatics. 2022, 17(5): 473-482.

[19] Faticanti, F., De Pellegrini, F., Siracusa, D., Santoro, D., & Cretti, S. (2020). Throughput-aware partitioning and placement of applications in fog computing. IEEE Transactions on Network and Service Management, 17(4), 2436-2450.

[20] Mouradian, C., Kianpisheh, S., Abu-Lebdeh, M., Ebrahimnezhad, F., Jahromi, N. T., & Glitho, R. H. (2019). Application component placement in NFV-based hybrid cloud/fog systems with mobile fog nodes. IEEE Journal on Selected Areas in Communications, 37(5), 1130-1143.

[21] Hossain, M. R., Whaiduzzaman, M., Barros, A., Tuly, S. R., Mahi, M. J. N., Roy, S., ... & Buyya, R. (2021). A scheduling-based dynamic fog computing framework for augmenting resource utilization. Simulation Modelling Practice and Theory, 111, 102336.

[22] Haoyan Zhang, Xudong Zhao, Huangqing Wang, Ben Niu, Ning Xu, Adaptive Tracking Control for Output-Constrained Switched MIMO Pure-Feedback Nonlinear Systems with Input Saturation, Journal of systems science & complexity, 36: 960–984, 2023.

[23] Heng Zhao, Huanqing Wang, Ben Niu, Xudong Zhao, K. H. Alharbi, Event-Triggered Fault-Tolerant Control for Input-Constrained Nonlinear Systems With Mismatched Disturbances via Adaptive Dynamic Programming, Neural Networks, 164: 508-520, 2023.

[24] Khezri, E., Zeinali, E., & Sargolzaey, H. (2023). SGHRP: Secure Greedy Highway Routing Protocol with authentication and increased privacy in vehicular ad hoc networks. Plos one, 18(4), e0282031.

[25] Zhongwen Cao; Ben Niu; Guangdeng Zong; Xudong Zhao; Adil M. Ahmad, "Active Disturbance Rejection-Based Event-Triggered Bipartite Consensus Control for Nonaffine Nonlinear Multiagent Systems", International Journal of Robust and Nonlinear Control, DOI:10.1002/rnc.6746.

[26] Trik, M., Molk, A. M. N. G., Ghasemi, F., & Pouryeganeh, P. (2022). A Hybrid Selection Strategy Based on Traffic Analysis for Improving Performance in Networks on Chip. Journal of Sensors, 2022.

[27] Mokhlesi Ghanevati, D., Khorami, E., Boukani, B., & Trik, M. (2020). Improve replica placement in content distribution networks with hybrid technique. Journal of Advances in Computer Research, 11(1), 87-99.

[28] Sandhiya, B., & Canessane, R. A. (2023, March). An Extensive Study of Scheduling the Task using Load Balance in Fog Computing. In 2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS) (pp. 1586-1593). IEEE.

[29] Mehta, R., Sahni, J., & Khanna, K. (2023). Task scheduling for improved response time of latency sensitive applications in fog integrated cloud environment. Multimedia Tools and Applications, 1-24.

[30] Kaur, A., & Auluck, N. (2023). Real-time trust aware scheduling in fog-cloud systems. Concurrency and Computation: Practice and Experience, e7680.

[31] Trick, M., & Boukani, B. (2014). Placement algorithms and logic on logic (LOL) 3D integration. Journal of mathematics and computer science, 8(2), 128-136.

[32] Hai, T., Alizadeh, A. A., Ali, M. A., Dhahad, H. A., Goyal, V., Metwally, A. S. M., & Ullah, M. (2023). Machine learning-assisted tri-objective optimization inspired by grey wolf behavior of an enhanced SOFC-based system for power and freshwater production. International Journal of Hydrogen Energy.

[33] Ahmed, A. M., Rashid, T. A., & Saeed, S. A. M. (2020). Cat swarm optimization algorithm: a survey and performance evaluation. Computational intelligence and neuroscience, 2020.

[34] Khezri, E., & Zeinali, E. (2021). A review on highway routing protocols in vehicular ad hoc networks. SN Computer Science, 2, 1-22.

[35] Saleh, D. M., Kadir, D. H., & Jamil, D. I. (2023). A Comparison between Some Penalized Methods for Estimating Parameters: Simulation Study. QALAAI ZANIST JOURNAL, 8(1), 1122-1134.

[36] Yahya, R. O., Mahmood, N. H., Kadir, D. H., & Aziz, S. J. (2023). The Use of Factor Analysis and Cluster Analysis Methods to Identify the Most Crucial Key Factors Influencing the Psychological Stability of University Students. Polytechnic Journal of Humanities and Social Sciences, 4(1), 779-789.

[37] Ihsan, R. R., Almufti, S. M., Ormani, B., Asaad, R. R., & Marqas, R. B. (2021). A survey on Cat Swarm Optimization algorithm. Asian Journal of Research in Computer Science, 10(2), 22-32.

[38] Ramachandran, M., & Ganesh, E. N. (2020). Energy Optimized Joint Channel Assignment and Routing using Cat Swarm Optimization (CSO) Algorithm in CRAHN. Journal of Green Engineering, 202, 3434-3449.

[39] Mahmud, R., Pallewatta, S., Goudarzi, M., & Buyya, R. (2022). iFogSim2: An extended iFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments. Journal of Systems and Software, 190, 111351.

[40] Rukmini, S., & Shridevi, S. (2023). An optimal solution to reduce virtual machine migration SLA using host power. Measurement: Sensors, 25, 100628.