IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON MOBILE EDGE COMPUTING AND MOBILE CLOUD COMPUTING: ADDRESSING HETEROGENEITY AND ENERGY ISSUES OF COMPUTE AND NETWORK RESOURCES

Received May 4, 2019, accepted May 26, 2019, date of publication June 26, 2019, date of current version August 31, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2924958

A Novel Bio-Inspired Hybrid Algorithm (NBIHA) for Efficient Resource Management in Fog Computing

HINA RAFIQUE[1], MUNAM ALI SHAH[1], SAIF UL ISLAM[2], TAHIR MAQSOOD[3], SULEMAN KHAN[4], AND CARSTEN MAPLE[5]

[1]Department of Computer Science, COMSATS University Islamabad, Islamabad 44550, Pakistan
[2]Department of Computer Science, Dr. A. Q. Khan Institute of Computer Science and Information Technology, Rawalpindi 47320, Pakistan
[3]Department of Computer Science, COMSATS University Islamabad, Abbottabad Campus, Abbottabad 22060, Pakistan
[4]Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne NE1 8ST, U.K.
[5]Warwick Manufacturing Group, University of Warwick, Coventry CV4 7AL, U.K.

Corresponding author: Saif ul Islam (saiflu2004@gmail.com)

This work was supported in part by the Alan Turing Institute under EPSRC Grant EP/N510129/1.

ABSTRACT Fog computing has emerged as a revolutionary paradigm to serve the massive data generated in the Internet of Things (IoT) environments. It can be considered a derivative of cloud computing that provides cloud-like services at the edge of the network. As such, it helps address the, often significant, issue of delays encountered when using cloud systems for the IoT. According to the literature, inefficient scheduling of user tasks in fog computing can actually result in higher delays than cloud computing. Hence, the real benefits of fog computing can only be obtained by applying effective job scheduling strategies. In fact, task scheduling is an NP-hard problem and requires optimal and efficient techniques to address issues of latency, response time, and the efficient resource utilization of resources available at the edge of the network. Given this, we propose a novel bio-inspired hybrid algorithm (NBIHA) which is a hybrid of modified particle swarm optimization (MPSO) and modified cat swarm optimization (MCSO). In the proposed scheme, the MPSO is used to schedule the tasks among fog devices and the hybrid of the MPSO and MCSO is used to manage resources at the fog device level. In the proposed approach, the resources are assigned and managed on the basis of the demand of incoming requests. The main objective of the proposed work is to reduce the average response time and to optimize resource utilization by efficiently scheduling the tasks and managing the fog resources available. The simulations are performed using iFogSim. The evaluation results show that the proposed approach (NBIHA) shows promising results in terms of energy consumption, execution time, and average response time in comparison to the state-of-the-art scheduling techniques.

INDEX TERMS Cloud computing, edge computing, fog computing, bio-inspired algorithms, task scheduling, resource management.

I. INTRODUCTION

Fog computing is a type of distributed computing which utilizes units between cloud data centers and IoT devices in a three-tier architecture, providing storage and processing services closer to end devices [1]. Fog computing utilizes and array of networking components, proxy servers, switches, setup boxes, base stations, and routers. These components have their own respective computing, storage, and networking services. The term fog computing was initially introduced by Cisco [2] as an extension of cloud computing to overcome limitations of cloud computing. Fog computing systems are being increasingly developed as efficient real-time systems in areas such as health care, augmented reality and gaming [3]–[5].

It has not been the intention that fog computing should replace cloud computing, but rather to be used in conjunction with more central cloud units to reduce delay, provide fast

The associate editor coordinating the review of this manuscript and approving it for publication was Junaid Shuja.

computing and reduce the cost of processing [6]. Fog nodes can be classified into two types: resource-poor devices such as routers, set-top-units, wireless access points (WAP) [7]; and resource-rich machines such as cloudlets, that can be considered a small scale version of cloud data centers, giving massive processing to mobile devices with low latency [8]. The basic architecture of fog and cloud computing is illustrated in Fig 1.

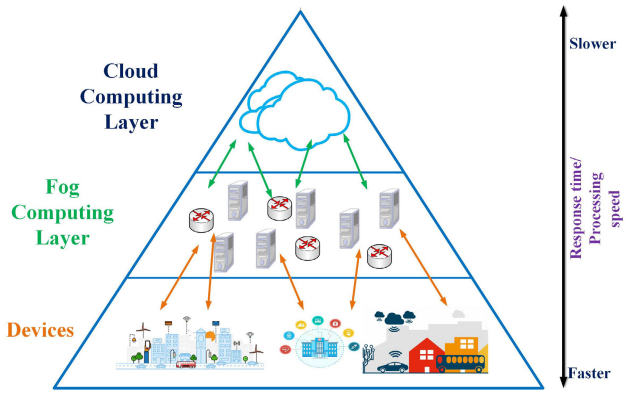


FIGURE 1. Cloud and fog computing architecture.

Driven by technological development by the academic and industrial communities, society is increasingly accepting massive connectivity of everything, everywhere. Academics and industry are advancing IoT-based smart X solutions - smart home, smart cities, smart transportation, smart military operations, smart metering and wearable computing [9]. As the number of IoT applications and implementations increase, so too has the attention of researchers.

As the number of IoT devices increases, there is a significant increase in energy consumption and degradation in performance [10]. Moreover, the high latency issues arising in the cloud-IoT paradigm makes it less practical for delay-sensitive IoT applications. As such, energy and performance-aware computational and storage services have become critical [11]. In the fog computing paradigm, routers behave as servers that provide the resources for the fog services [8], [12], [13]. Routers enhance the computation and storage capacity that can be utilized by computing nodes. Many IoT applications that deal with the real-time scenarios are moving to edge computing nodes [11], [12], [14].

Efficient resource management in IoT-based fog computing environment is a significant challenge due to the dynamic nature of bandwidth, storage, computation, and latency of end devices. For example, we can keep records of on-duty ambulances (such as capabilities, location and so forth) in a connected vehicle scenario and on the other hand, we can also control smart traffic lights to route the ambulance in the case of an emergency. With such demands for computation and speed, it is important to manage resources to ensure quality of service (QoS) requirements. Ottenwälder et al. [15] proposes the placement and migration method, named MigCEP, for the management of resources in both cloud and fog. The reduced network utilization and end-to-end latency restrictions are ensured by complete planning of operator mitigation. Application-aware provisioning helps fog computing to be effectively used with IoT for mobile crowdsourcing or sensing.

Since extending the cloud to the network edge, efficient resource management has become an increasingly challenging task [16]. As such, there is a dire need to propose and develop energy-aware and performance-oriented fog resource management strategies. In this context, we have proposed a novel bio-inspired hybrid algorithm scheme (NBIHA) for efficient resource management in fog computing. This approach is a hybrid of two state-of-the-art bio-inspired algorithms. The proposed approach schedules the tasks and manages the resources to achieve efficient resource utilization and improved performance. According to the proposed approach, the scheduler finds the best match of fog devices for an incoming task depending on its demand of CPU time and memory, and allocates it the resources accordingly. Only if it cannot identify any appropriate resource from the fog devices does it send a task to cloud-based resources. The main contributions of this paper are summarized below:

- The proposition of a novel bio-inspired hybrid algorithm that combines modified particle swarm optimization (MPSO) and modified cat swarm optimization (MCSO)
- The proposed algorithm has two major components - task scheduling and resource allocation in fog computing to optimize the resource utilization and to minimize the response time and processing cost
- The proposed approach is evaluated using iFogSim simulator and is validated through a comparison with state-of-the-art resource management algorithms. The results verify the effectiveness of the proposed approach in terms of energy consumption, processing cost and response time.

The rest of the paper is organized as follows: Section II describes the related work while Section III explains the system architecture and proposed methodology of the system along with the problem formulation. Section IV presents the algorithms of the proposed technique. Section V defines the performance parameters and evaluation metrics which are used for the comparison. Section VI describes the results of the proposed technique. Finally, in Section VII, the conclusion and future work are presented.

II. RELATED WORK

In [13], the Bee Life Algorithm (BLA) is used to assign a bag of jobs to fog or edge nodes that are situated at the edge of the network. It focuses on the reduction of execution time and the memory required by the overall mobile tasks executed on the fog nodes. This time and cost-aware evolutionary scheduling algorithm is used to schedule tasks on fog and cloud resources according to the requirement of tasks, maintaining the trade-off between cost of execution and

time of execution. The paper mainly focuses on resolving the scheduling problem for bag of tasks (BoT) applications in a hybrid cloud-fog computing environment. Time–Cost aware Scheduling (TCaS) is an evolutionary algorithm proposed by Binh et al [17], whose performance is evaluated with different data-sets of tasks on fog and cloud devices. Its optimization criteria provide a trade-off between time and cost and user satisfaction. A three-layered model is presented to optimally allocate resources in cloud-fog environments. The system is divided into three parts: the client layer, the fog layer; and the cloud layer. The algorithm attempts to allocate tasks to the client and fog layer, with the remaining requirements accommodated through cloud resources. The factors considered are overall response time, processing time and cost of data-center. The limitation of this model is that no run time allocation of resources is provided, and it allocates resources before processing.

Gudetti et al. [18], propose a model comprising two components - job allocation to virtual machines (VMs) and allocation/management of fog resources. Two bio-inspired algorithms are utilized - MPSO for job allocation and MCSO for resource assignment/allocation and management. The method provides improved utilization of available resources, reliability and average response time, and improves all criteria for cloud-based applications. This model is only applicable in cloud environments. A recent application for fog computing environments was the development of a fog computing supported medical cyber-physical system (FCMCPS), presented in [19]. In this new model a cost-efficient solution for managing base station links, task division, and VM placement was proposed. The issue is considered to be a mixed-integer nonlinear programming (MINLP) problem for base station association, VM allocation and task assignment. A MILP technique is used, reducing the complexity of the problem. A two-phased linear programming based heuristic algorithm was developed to solve the problem. For a survey of scheduling techniques employed to improve results performance in terms of processing, time, and resource utilization for cloud systems see [31].

Deng et al. [20] have developed a framework to explore the problem of imbalance between consumed power and the delay of workload distribution in cloud-fog environments. This framework decomposes the into three small problems of respective subsystems to provide an optimized solution of communication latency in the system. These problems are then solved by using the Hungarian method. Simulations have been conducted to show reduced communication latency in fog environments, in contrast to cloud environments. However, a shortcoming of this system is that it performs optimization in a centralized manner, rather than a distributed manner. This makes it more difficult to use in fog infrastructures where the system is complex and information exchange and communication overheads are high. Oueis et al, [21], propose a method that involves forming *clusters* of resources for load balancing for fog computing environments. The method allocates resources to meet the expectations of user requests and attempts to minimize power consumption and process complexity. The algorithm uses a two-step method to allocate resources. In the first stage, resources are distributed to smart cells by the use of a specified scheduling rule, whereas in the second stage clusters are formed to handle unfulfilled requests. In this system, the user can change metrics, scheduling rules, and clustering objectives according to the requirement of the specific application and network. While this system has benefits, it is very difficult to use in complex fog infrastructure.

The method proposed by Ningning et al. [23], employs graph theory concepts with fog characteristics to construct a model for load balancing. The method uses a cloud atomization process to convert VMs into different physical nodes. For this purpose, it uses a specified amount of resources and clustering division. Tasks or jobs are allocated to single or multiple VM nodes according to the demand of resources required by the task. This model does not allow dynamic load balancing.

In Table 1, we have summarized the key characteristics of existing techniques, along with their limitations and relevant architecture. The problem at hand is to provide an intelligent solution to task scheduling, load balancing, and resource management in fog-cloud computing environments. The main difference between the present work and previous work is that we are combining two algorithms to achieve two goals, one is to schedule the tasks and the other is to manage the resources to improve the response time and resource utilization. The rationale of using a modified hybrid algorithm is that we are scheduling tasks and managing resources concurrently. The literature shows that in cloud computing environments MPSO has been successfully employed to improve resource utilization [29], whereas MCSO has been used to enhance and improves the search efficiency within the search space as we are finding a best-fit resource for task processing [30].

III. SYSTEM ARCHITECTURE

In this paper, we consider a fog-cloud system which consists of fog and cloud processing nodes. Our system architecture is comprised of three layers - client module, scheduler, and fog devices and cloud data centers. The operation of our proposed NBIHA architecture is that all the client requests are received by the scheduler. The scheduler performs an optimization algorithm to find the best resource match for jobs based upon CPU and memory demand of the tasks. It schedules the task using the MPSO algorithm approach to find the global best (GB) and to perform load balancing before the hybrid MPSO-MCSO algorithm is used to manage resources based on the fitness function. While a number of simulators are available, we have implemented our system model using the iFogSim simulator in order to evaluate the performance of our proposed approach. The evaluation has been discussed in next sections. Fig 2 shows the model of our proposed system.

TABLE 1. Summary of literature review findings.

Approach	Algorithm/ Methodology	Improved Criteria	Limitations	Technology
[13]	Bee Life Algorithm	Execution time, memory, Optimal division of tasks among fog devices. Trade-off between CPU time and allocated memory.	Only fog devices are used.	Fog computing
[18]	Hybrid PSO and CSO	Cloud resource utilization, reliability,response time	If found no match of VM takes long waiting time	Cloud computing
[22]	Efficient Resource Allocation (ERA)	response time, Processing time and cost of data-center	No run time allocation of resources. Allocates resources before processing	Cloud-fog based
[17]	Time Cost aware Scheduling Evolutionary Algorithm-Genetic Algorithm	Trade-off between execution time and total operating cost.	Only one comparison is given	Cloud-fog based
[19]	Mixed-integer Linear programming heuristic algorithm	Minimized cost and Improved QoS requirement.	Complex system	Fog computing
[20]	Non-linear integers, Hungarian Method.	Reduced communication latency, breakdown of primal problem in sub problems.	Problem is decomposed but solved in centralized manner not in distributed manner.	Cloud-fog computing
[21]	A heuristic Algorithm is used to allocate resources.	User Quality of Experience improved.	Difficult to use in complex fog systems.	Fog computing
[8]	MarkovâĂŹs Decision Process, Index Policy.	Power consumption and delay trade off	Computational Overhead and time queuing system.	Fog computing
[23]	Graph theory based on graph partitioning.	Load balancing	No dynamic load balancing	Fog computing
[24]	Mobility-Aware Application Scheduling in fog Computing.	Movement based scheduling of applications using FCFS, delay priority and concurrent strategy at cloudlet level.	If mobility cannot be predicted it can cause delay.	Fog computing
[25]	Context-aware scheduling.	Task provision using context to reduce response time and cost.	Task provisioning on fog devices irrespective of power of device.	Fog computing
[26]	Hungarian and Genetic Algorithm based solution.	Task placement through Hungarian method and virtual machine placement through GA.	Cloud computing solution.	Cloud computing
[27]	Load balancing using scheduling algorithms.	Execution time, fast response.	QoS factors are not included.	Fog computing
[28]	Energy-efficient computation offloading and resource allocation (ECORA).	An algorithm is proposed to offload the work on fog devices to minimize the cost.	Difficult to use for complex systems.	Fog computing

In Fig 3, we have shown the hierarchical system model of proposed approach.

A. PROPOSED METHODOLOGY

The proposed methodology presents a novel bio-inspired hybrid algorithm (NBIHA) for job or task assignment and resource management. In our work, we are focusing on the task assignment of incoming task over fog devices using bio-inspired MPSO algorithm. In this way, we will get a better task assignment with respect to demand for the job and average response time of fog devices. For management of fog devices with respect to CPU and memory demand of a task we have used a hybrid scheme composed of two algorithms, i.e., MPSO and MCSO. If fog devices are not available, tasks will use resources available in the cloud data centers.

As the number of IoT users increases, and technology at the edge improves, there is a greater requirement to make optimal use of resources to ensure adequate service delivery. The delivery of services requires a combination of cloud and fog resources and dynamically allocation remains a challenge. We have based our approach on for resources utilization on MPSO, as it has shown promising results for cloud-based systems [29]. For resource management we have based our

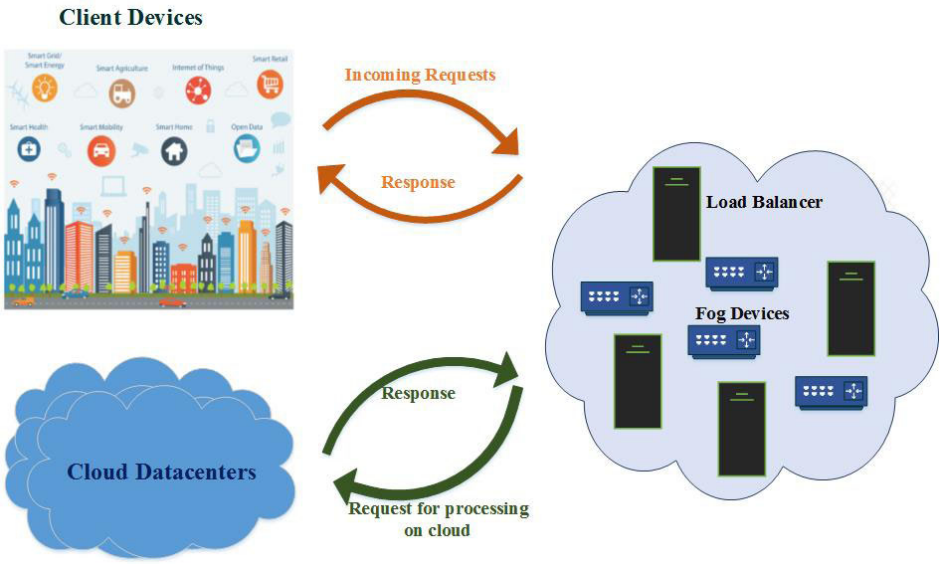


FIGURE 2. NBIHA system model.

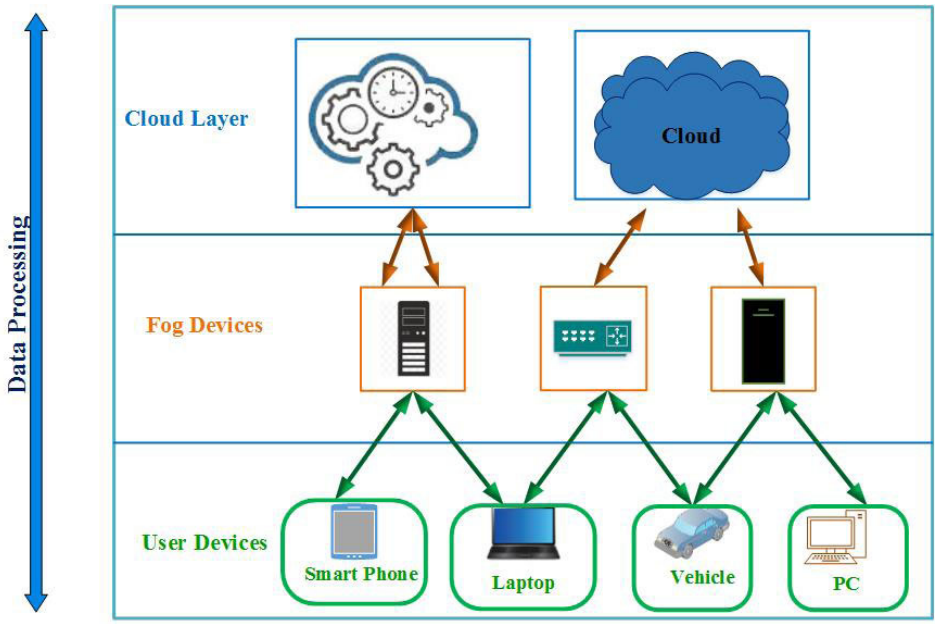


FIGURE 3. NBIHA system model.

solution on a modified form of MCSO since it has previously shown to be effective in this regard [30].

B. PROBLEM FORMULATION

We designed a system for task load balancing and resource management in fog-cloud computing environments. We describe a task as a method that defines a service demand made by a user that could take the form of a mobile user, web user, or other Internet users. Incoming requests, i.e. tasks t_i $\{t_1, t_2, t_3, t_4 \ldots, t_n\}$ are to be scheduled on available fog devices, i.e. f_k $\{f_1, f_2, f_3, f_4 \ldots, f_m\}$ and cloud resources. Our proposed approach uses MPSO to schedule tasks and to balance the load by selecting the best fit fog and cloud devices for processing of requests. After scheduling tasks, we will find the average response time of the fog nodes by using the following equation (1).

$$AVT = t_2[\sum_{x=1}^{m} FD(x)] - t_1[\sum_{x=1}^{m} FD(x)] \quad (1)$$

To find the fitness value of the best fit we have used following formula

$$FV = \frac{(R)\sum_{FD=0}^{m} FD(k)}{\sum_{x=1}^{n}(RD)} \quad (2)$$

We have used equation (2) to calculate the resource demand of the task.

C. ALGORITHMS

In this section, we will briefly explain the algorithms we have used in our proposed approach and after that, we will explain how we have used NBIHA to resolve our problem. In [29], it is mentioned that many researchers have used PSO to solve scheduling problems in cloud computing environments. We can consider fog computing to be a derivative of cloud computing and so use the best performing meta-heuristic techniques as a basis for solving issues in the fog environment.

1) ILLUSTRATION OF MPSO ALGORITHM

In this algorithm, particles are considered where each particle has two properties - position $\{x_{i1}, x_{i2}, x_{i3} \ldots, x_{in}\}$ and the velocity of the particle $\{v_{i1}, v_{i2}, v_{i3} \ldots, v_{in}\}$ in the x axis. In this process, each particle has its personal (local) best position identified by itself, whereas the global best is found among all of the particles. Fig 4 illustrates the MPSO algorithm. The MPSO has following steps:

1) Initiate each particle with position and velocity.
2) Evaluate the fitness value of each particle using fitness function.
3) Compare particle with the largest fitness value calculated in above step, initiate its position and update value; and compare this particle with the smallest (optimal) fitness value and check whether its new position is suitable, if yes, change and update its personal position (pbest), otherwise, assign a new position to this particle randomly in its surroundings with radius r and then update the position and velocity of other particles according to the fitness function;
4) Now compare each particle's current fitness value with its personal best (pbest), if the current fitness value is better, then change its fitness value and personal best (pbest) to the new best.
5) Now find the best particle among the group with the best fitness value, and compare the current fitness value and the fitness value of global best (gbest), if yes, then renew its fitness value and global best gbest) with the current position;
6) Check the set criteria (fitness function) to find the optimal solution, if it has been achieved, end the iteration of the algorithm; otherwise, return to step 3) [32], [33]

The Fig 4 shows the activity of the MPSO.

2) ILLUSTRATION OF MCSO ALGORITHM

We employ the seeking (or tracing) mode of MCSO in our algorithm. In the seeking mode of MCSO, the condition of the cat is depicted as resting, looking and seeking for the position to make next movement. This mode has four basic components: seeking for memory pool (SMP), seeking for a range of the dimension (SRD), count of measurement to change (CDC), and self-position consideration (SPC). SMP is used to specify the memory pool for each in which it will seek for the next position. The cat picks up a point from the SMP by following the below-mentioned steps. SRD is used to state the proportion of the dimensions. In this mode, when a dimension has changed the difference between both values would not be out of the range of the SRD. CDC stores the number of dimensions that can be changed. These are the essential elements of the seeking mode. SPC has knowledge of the point where the cat stands, and identifies whether it is possible to move to a particular point; it is a Boolean variable. Whether SPC is true or false, it does not affect SMP [34], [35].

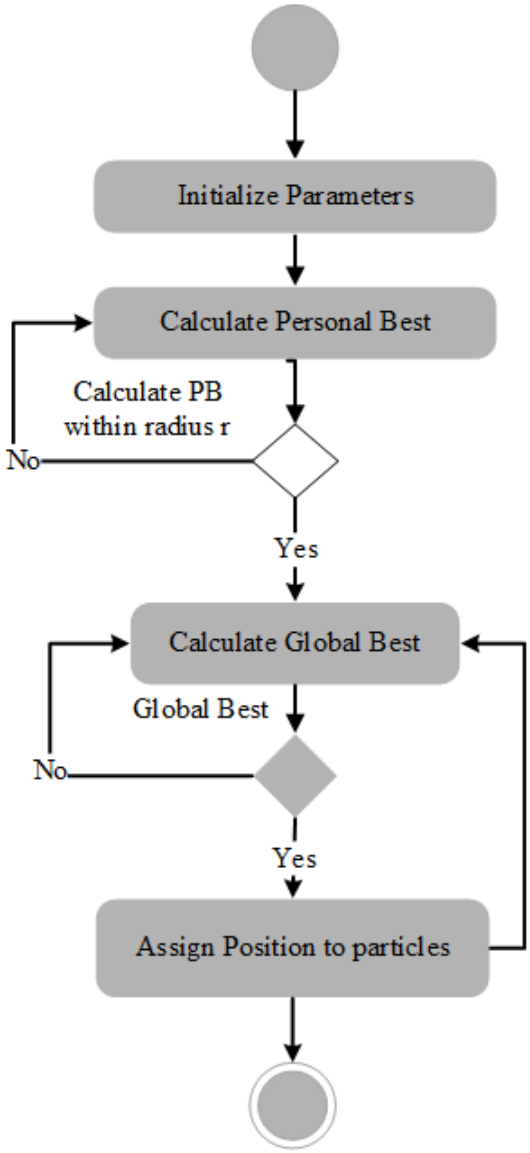


FIGURE 4. Data flow of MPSO.

The function of seeking mode is depicted in 5 steps shown as below:

1) Make n copies of the present position of cati, where n = SMP. In the start consider that the estimation of SPC is valid, let n = (SMP-1), by then hold the present position as one of the candidates.
2) For each copy, as shown by CDC, randomly add or remove SRD percent of the present characteristics and replace the old ones.
3) Compute the fitness value (FS) of all candidate points.
4) If all fitness values are not actually equivalent, determine the choosing possibility of every candidate point by condition mentioned in equation(3). Generally, the probability is considered as 1.
5) Randomly pick the next point to move to from the current points and change the position of cat_i.

$$P_k = \frac{|FS_k - FS_a|}{FS_max - FS_min} where 0 < k < n \quad (3)$$

For minimum solution FS_a=FS_max

IV. NBIHA FOR FOG COMPUTING JOB SCHEDULING AND RESOURCE MANAGEMENT

A number of modified particle swarm optimization (MPSO) algorithms have been proposed since the derivation of the original particle swarm optimization (PSO) algorithm. In our approach, we will use modifications to the PSO to resolve task allocation and load balancing in a cloud-fog computing environment.

A. TASK SCHEDULING AND LOAD BALANCING

In our Algorithm 1, we have used MPSO for task scheduling and allocation. MPSO is used to find the best fit fog device for processing of incoming tasks. In this Algorithm, we have created a resource pool in which all the resources are saved. We have fog devices such as $F = \{f_1, f_2, f_3, f_4 \ldots, f_j\}$ and there are incoming requests $X = \{x_1, x_2, x_3, x_4 \ldots, x_n\}$ for processing. Each cluster finds the personal, or local, best (LB), which is considered as the least loaded fog device. From these least-loaded fog devices, the smallest is considered to be the global best (GB) and is assigned to process the requested task. If there is no match found, the task is sent to the cloud for processing.

B. RESOURCE ALLOCATION AND MANAGEMENT

1) RESOURCE ALLOCATION AND MANAGEMENT USING MPSO

In Algorithm 2, we have used the MPSO algorithm to allocate and manage fog devices. In this algorithm, we have created a resource pool in which all the resources are saved. We have fog devices defined by F and incoming requests for processing defined by X. Initially, resources are distributed based on the demand of the tasks. Subsequently, from the remaining resources, we find the two best values (*bestfitres1* and *bestfitres2*). *Neededres* contains the required resources for incoming requests. We compare *Neededres* with *bestfitres1* - if it fits we assign it for processing, otherwise we compare with *bestfitres2*. If this matches we send a request for processing, otherwise the resources are sent to *respool* and requests are sent to the cloud for processing. In Algorithm 2, we have used the exact matches to identify the resource for processing. This has a drawback: if it does not find exact best fit match then it sends requests to the cloud, which results in an increase in the computational overhead.

Algorithm 1 Load Balancing by Scheduling Tasks Using MPSO

Result: *No of executed tasks*
initialization:
count $\leftarrow 0$
PersonalBest$(lbz) \leftarrow 0$
GlobalBest$(gb) \leftarrow 0$
$F \leftarrow f1, f2, f3, \ldots, fj$
Clusters $Cz \leftarrow c1, c2, c3, \ldots, cz$
Clustersize $\leftarrow j/Cz$

for *incoming requests* $x1, x2, x3, \ldots, xn$ **do**
  for $Cz = c1, c2, c3, \ldots, cz$ **do**
    $Cz \leftarrow F(leastloaded)$
    Assignlbz $\leftarrow Cz$
  end
  Assign gb = leastlbz
  Allocate next task x ←F(gb)
  if *nextallocation* == *lastusedGB* **then**
    goto step 3(leastloadedF)
  else
    goto step 7
  end
end
for *all unallocated tasks* $x1, x2, x3, xn$ **do**
  Assign to Cloud
end

2) RESOURCE ALLOCATION AND MANAGEMENT USING NBIHA

In Algorithm 3, we propose the NBIHA approach to allocate and manage fog devices. This algorithm will overcome the drawbacks of the MPSO algorithm. We have created a resource pool in which all the resources are saved, have fog devices defined in F and incoming requests for processing given in X. We begin with resources distributed based on the demand of the tasks, and then, from the remaining resources, the two best values (*bestfitres1* and *bestfitres2*) are identified. *Neededres* contains the required resources for incoming requests. We now use the seeking memory pool of MCSO to store resources, with the exception of *bestfitres1* and *bestfitres2*. Four distinct types of memory pools are used for the purpose of using the MCSO in seeking mode. Based on the SMP, the cat scans for the accurate counterpart for

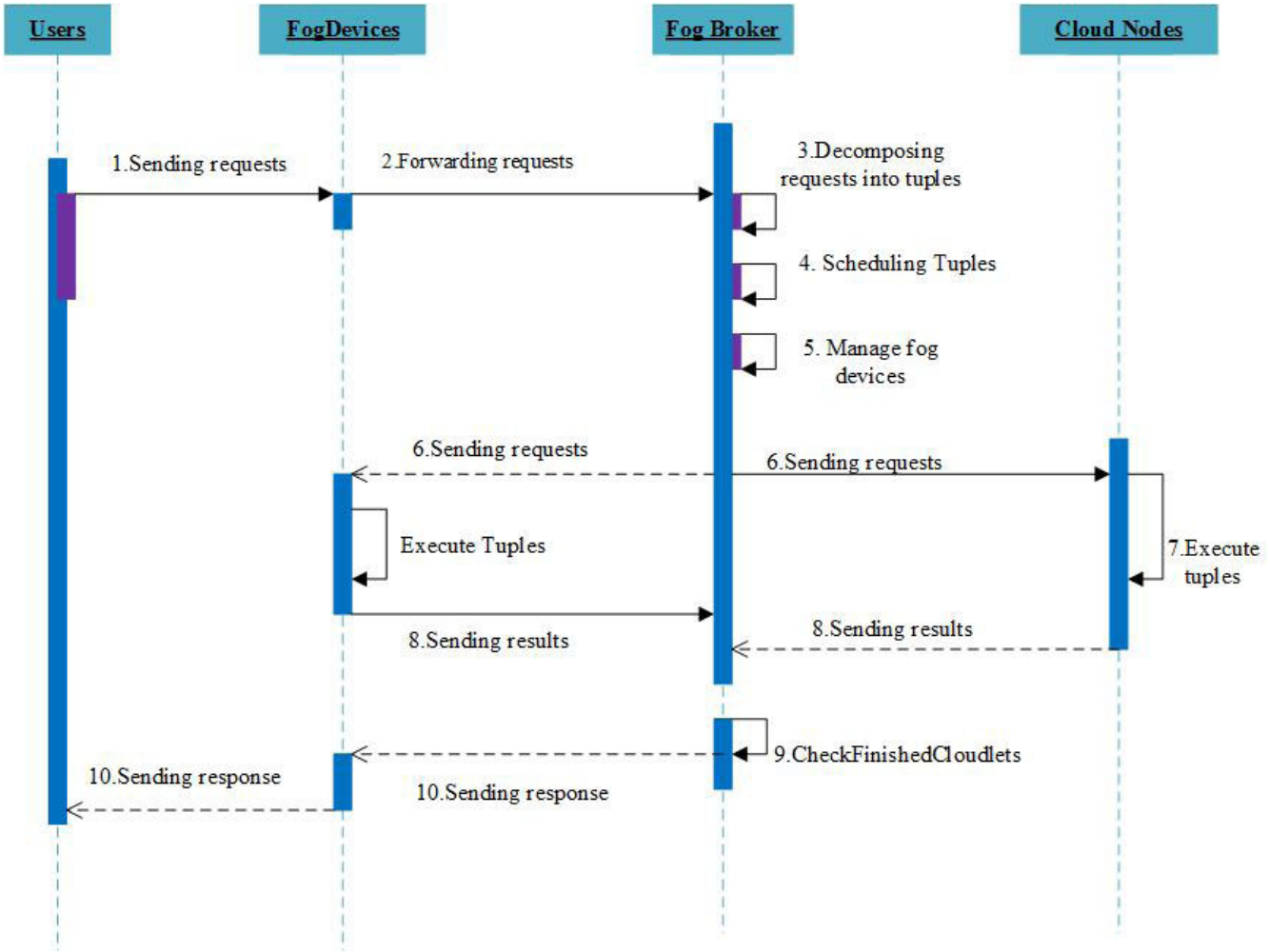


FIGURE 5. Sequence diagram of NBIHA.

future resource demand of the request made by the task. In the event that there is a match, at that point the status is retained within the scope SRD. If SRD has another update, at that point the fog device starts executing the assignment and this status is used to change the CDC. Each update of SMP of MCSO effects a change in position of Cats; this is maintained in SPC. In MCSO, the result of the seeking mode is used as input to the next mode and we then use the method of Algorithm 2. In the hybrid algorithm the upper bounds of the *bestfitres1*, *bestfitres2*, and *bestfitres3* are checked. After assigning processing resources to tasks, extra resources are sent to the fog resource pool. This approach will remove the drawbacks of an exact match that results in the additional computational overhead. It also improves the execution time and average response time of the fog nodes.

Fig 5 shows the operation of NBIHA in cloud-fog computing environments. The user sends requests, which are collected by the fog devices. The fog broker manages the fog devices as well as the requests generated by the user. The fog broker divides the requests into tuples and schedules tasks and manages fog devices based on the MPSO algorithm. The tuples are then sent to fog devices and cloud devices as identified. After processing, the fog broker checks the completion of the task, compiles the results and sends these back to the user through the fog devices.

V. PERFORMANCE EVALUATION

Extensive simulations have been conducted using iFogSim in a cloud-fog environment to assess the performance of our NBIHA approach. Cloud and fog nodes have varied processing power and usage cost. We assumed that each fog node or device has its own processing limit (estimated by MIPS - million instruction per second), alongside CPU, memory, and transmission capacity utilization cost. There are 15 processing nodes in the fog framework, with the specifications as shown in Table 2. The unit of cost is given as Grid dollars, often used in simulations, as a replacement for real money. In the fog layer, fog nodes have restricted processing power, for example, switches, doors, workstations, or PCs. In the cloud layer, servers or virtual machines in elite server farms,

Algorithm 2 Resource Management and Allocation Using MPSO Algorithm

Result: *No of executed tasks*
initialization:
Respool, Neededres, minres = 2 $sumres \leftarrow 0$
$Fd \leftarrow f1, f2, f3, ., fj$
$C \leftarrow cloud$
Clusters $Cz \leftarrow c1, c2, c3, , cz$
$Clustersize \leftarrow j/Cz$

for *incoming requests* $x1, x2, x3, \ldots, xn$ **do**
  $Resdemand \leftarrow totalresourcesrequiredbyrequests$
  $NeededRes \leftarrow resourcesrequiredbyeachrequest$
  for $Ff1, f2, f3, fj$ **do**
    $sumres \leftarrow sumres + Resdemand$
  end
  if *start* **then**
    $Respool \leftarrow Respool - sumres$
  else
    for *all Cluster size,* Cz $c1, c2, c3, \ldots, cz$ **do**
      $bestfitres1 \leftarrow firstbestcz$
      $bestfitres2 \leftarrow secondbestcz$
    end
    for *all* $Fd = fd1, fd2, fd3, ., fdj$ **do**
      for *all Cluster size,* $Cz = c1, c2, c3, \ldots, cz$ **do**
        if $bestfitres1 == Neededres[]$ **then**
          Fd uses excessres1
        else
          $Respool \leftarrow Respool - Neededres[]$
        end
        if $bestfitres2 == Neededres[]$ **then**
          Fd uses excessres2
        else
          $Respool \leftarrow Respool - Neededres[]$
        end
      end
    end
  end
  $Respool \leftarrow Respool + leftNeededres[]$
end
for *all unallocated tasks* $x1, x2, x3, xn$ **do**
  Assign to Cloud
end

Algorithm 3 Resource Management and Allocation Using NBIHA

Result: *No of executed tasks*
initialization:
Respool, Neededres, minres = 2 $sumres \leftarrow 0$
$Fd \leftarrow f1, f2, f3, ., fj$
$C \leftarrow cloud$
Clusters $Cz \leftarrow c1, c2, c3, , cz$
$Clustersize \leftarrow j/Cz$

for *incoming requests* $x1, x2, x3, \ldots, xn$ **do**
  $Resdemand \leftarrow totalresourcesrequiredbyrequests$
  $NeededRes \leftarrow resourcesrequiredbyeachrequest$
  for $Fdf1, f2, f3, fj$ **do**
    $sumres \leftarrow sumres + Resdemand$
  end
  if *start* **then**
    $Respool \leftarrow Respool - sumres$
  else
    for *all Cluster size,* Cz $c1, c2, c3, \ldots, cz$ **do**
      $bestfitres1 \leftarrow firstbestofcz$
      $bestfitres2 \leftarrow secondbestofcz$
      $bestfitres3[] \leftarrow exceptfirstand\ secondbestofcz$
    end
    for *all* $Fd = fd1, fd2, fd3, ., fdj$ **do**
      for *all Cluster size,* $Cz = c1, c2, c3, \ldots, cz$ **do**
        if $bestfitres1 >= Neededres[]$ **then**
          Fd uses excessres1
        else
          $Respool \leftarrow Respool - Neededres[]$
        end
        if $bestfitres2 >= Neededres[]$ **then**
          Fd uses excessres2
        else
          $Respool \leftarrow Respool - Neededres[]$
        end
        while *size of(excessres3[])* **do**
          if $bestfitres3 >= Neededres[]$ **then**
            Fd uses excessres3
          else
            $Respool \leftarrow Respool - Neededres[]$
          end
        end
      end
    end
  end
  $Respool \leftarrow Respool + leftNeededres[]$
end
for *all unallocated tasks* $x1, x2, x3, xn$ **do**
  Assign to Cloud
end

are in charge of taking care of requests. In this manner, the preparation rate of cloud nodes is much quicker than fog nodes. Interestingly, the expense of utilizing resources in the cloud is more costly than in the fog.

The fog framework is responsible for the execution of all incoming requests from the clients. Each request is divided into number of task tuples, which are decomposed and evaluated based upon the processing that they required. Each

TABLE 2. Fog simulation environment parameters.

Parameters	Fog	Cloud
Number of Nodes	10	5
CPU MIPS	[500,2000]	[3000-10000]
CPU usage cost(G$)	0.4	2.0

TABLE 3. Properties of incoming requests.

Property	Value
Number of Instructions (MIPS)	[1-200]
Memory Required (MB)	[50-200]

TABLE 4. Characteristics of simulation setup.

Properties	Value
System	Intel Core i5 4th Gen1.7Ghz
Memory	4 GB
Operating system	Windows 10 Professional

task has a number of characteristics such as the memory required, input document size, and output. We have tested using between 20 and 60 tasks evaluate our proposed NBIHA approach. The characteristics of the tasks are presented in Table 3. With randomness, the simulations may cover different situations in light of the fact that numerous kinds of requests are made, some of them require a large measure of preparation while others need more memory or data transfer capacity, and so forth. There are issues in simulating fog and edge computing environments due to the of diverse nature of sensors and fog devices [36]. The settings of the experimental environment are shown in Table 4. The simulations are created in Java with Eclipse editorial manager, and utilizing iFogSim [37], [38]. iFogSim is selected because it is dependent on CloudSim, a cloud computing simulator that has been widely used and approved in various experiments to perform simulations. In Fig 6, we have shown the topology of our system which has been created using iFogSim to evaluate the performance of our system.

A. EVALUATION METRICS

We believe that by scheduling task assignment and resource management will improve resource utilization and average response time and will evaluate on such metrics. Our null hypothesis and alternate hypothesis are given below:

- H0: There will be no difference in resource utilization and average response time of the resources available in the cloud-fog system after scheduling and managing resources using bio-inspired heuristic algorithms.
- H1: The system having been scheduled using NBIHA will have better resource utilization and average response time of the resource.

VI. RESULTS

Results of our proposed approach are analyzed with respect to: resource utilization; average response time; energy consumption; and execution time. Moreover, the proposed algorithm is compared with benchmark algorithms for scheduling such as shortest job first (SJF), first come first served (FCFS). We have also compared our proposed approach with particle swarm optimization (PSO) meta-heuristic approach in the case of resource management and allocation to fog nodes.

A. LOAD BALANCING BY USING TASK SCHEDULING

As mentioned earlier, our work is divided into two parts, load balancing and, resource management. To accomplish the first goal, we have used the proposed MPSO algorithm for task scheduling which will, in fact, balance the load of fog resources. In this section, we have compared our results with scheduling algorithms like the SJF and FCFS. We have used these algorithms for comparison because these are benchmark algorithms for scheduling. Figure 7 shows that the average response time of the proposed MPSO approach is smaller than all of the other schemes. Using FCFS, the incoming requests have to wait if the resources are busy since it allocates resources in the order of arrival whereas using SJF large incoming requests have to wait longer since it prioritizes processing shortest jobs first. In our proposed algorithm the tasks are scheduled with respect to the resource demand. It balances the load as resources are utilized with respect to the demands of the incoming requests. We can see in Fig 7, that MPSO uses all fog nodes equally whereas the other two algorithms use a smaller number of resources thereby increasing the load. We have not considered MCSO and the hybrid approach for scheduling since they take more time. We have evaluated our task scheduling and load balancing by considering resource utilization when we use the MPSO which efficiently schedules the tasks and increases the maximum and provides approximately equal utilization of all fog nodes.

In Figure 7 we also see that there is fluctuation in resource utilization while using different approaches. Since FCFS allocates resources on the basis of arrival of the demands of incoming requests, there is an uneven utilization of resources. Conversely, MPSO finds the best fit using global best and balances the load. For this reason resource utilization is evenly divided on all the fog nodes.

B. RESOURCE ALLOCATION AND MANAGEMENT

In Fig 8, the results of experimentation with a set of 10 and 20 fog devices in groups, with 60 incoming tasks. After task scheduling using MPSO, we use the MCSO approach to manage the resources. In this *bestfitres3* will be compared for the future demand of the tasks. In Fig 8, we can see the results for MPSO, SJF, FCFS and the hybrid approach. We see that

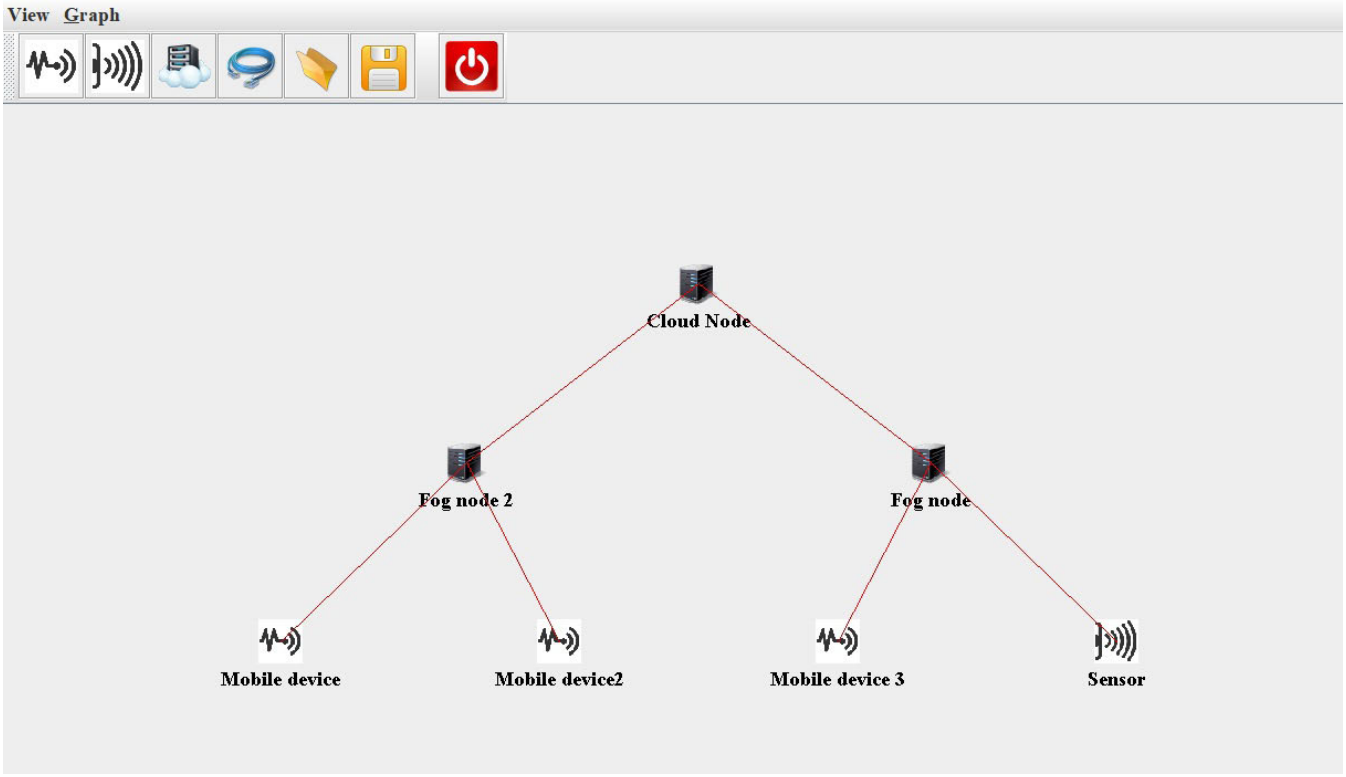


FIGURE 6. Fog computing environment topology designed in iFogSim.

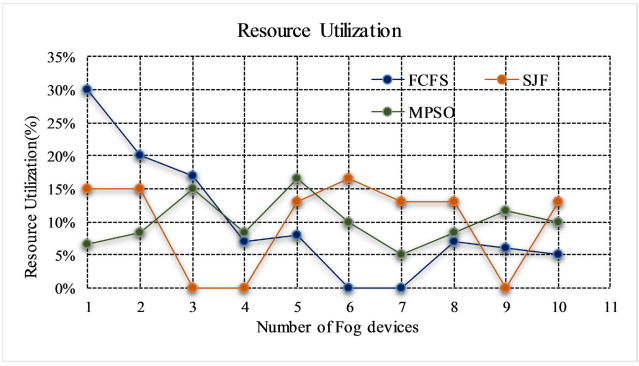


FIGURE 7. Resource utilization in terms of tasks performed by each fog device.

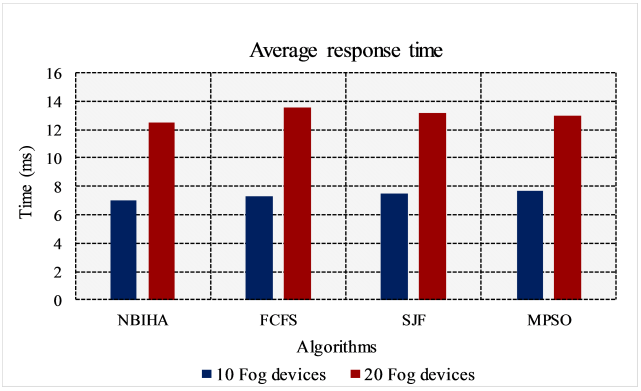


FIGURE 8. Average response time of algorithms on the basis of task completion time.

the hybrid approach provides reasonable results with respect to average response time.

C. ENERGY CONSUMPTION

In Fig 9, we have presented the results of experimentation using our approach and using FCFS and SJF with respect to the energy consumption of fog devices. In Fig 9, we have considered 24 fog devices to evaluate the energy consumption of resources by using NBIHA, FCFS and SJF. Energy consumption in the case of NBIHA is lower than the other two approaches. It can be seen in Fig 9 that due to load balancing, all nodes are approximately equally utilized that is why the energy consumption in case of the proposed approach is lower than the other two approaches. For our approach, the fitness value is determined on the basis of which resource is allocated to the job for processing. The energy consumption is almost equal because in the first iteration of our proposed approach it provides resources to the incoming requests and after that it manages the load of resources, rather than simply in order of arrival or prioritizing shortest jobs, as in FCFS and SJF, respectively.

In Fig 10, we have used sets of 5, 10, 15, and 20 fog devices and evaluated the performance of our proposed approach with respect to energy consumption. It can be seen that when we increase the number of fog devices the usage of energy

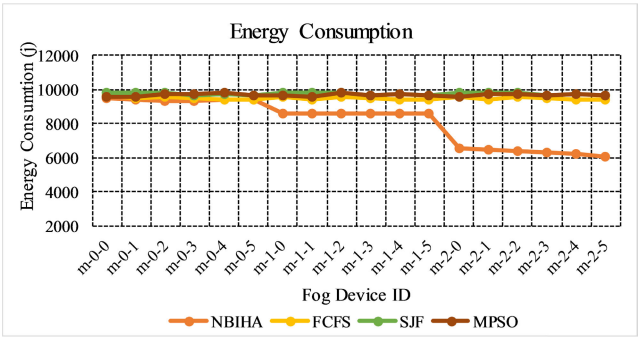


FIGURE 9. Energy consumption of algorithms with respect to each fog device.

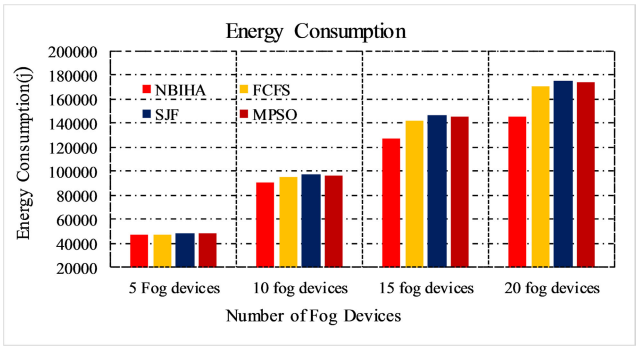


FIGURE 10. Energy consumption analysis of algorithms on the basis of 5,10,15 and 20 fog devices.

increase in all cases. The energy consumption increases with the increase in fog devices because the more we add fog devices to the system the energy of switching and processing increases. When our approach is compared to FCFS and SJF, we see that NBIHA has a lower energy consumption level in all sets of fog devices.

D. EXECUTION TIME ANALYSIS

In Fig 11, we consider the best, average and worst case results of the four algorithms. We have explored different combinations regarding the tasks and fog devices with state- of-the-art conventional scheduling algorithms and NBIHA. In Fig 11, the worst scenario of execution time for the proposed MPSO and NBIHA is an unusual case where the resource match never occurs with the incoming requests. Different resources are used by the FCFS to fulfill the demands of the incoming request. These resources are retrieved from the buffer of cloud resources, consequently, the execution time of FCFS will be same as before for all these cases. In SJF, the resources requirement has been matched with the fog devices that have sufficient processing power to process the shortest job first. Since most of the time there would not be any ideal resource match for incoming requests or due to the waiting of long jobs hence, SJF takes approximately the same execution time for all cases to execute tasks. In the worst-case scenario, the execution time for MPSO, and NBIHA approach is increases because the delay in matching of demand and resource has a direct effect on processing time. In the MPSO, whenever

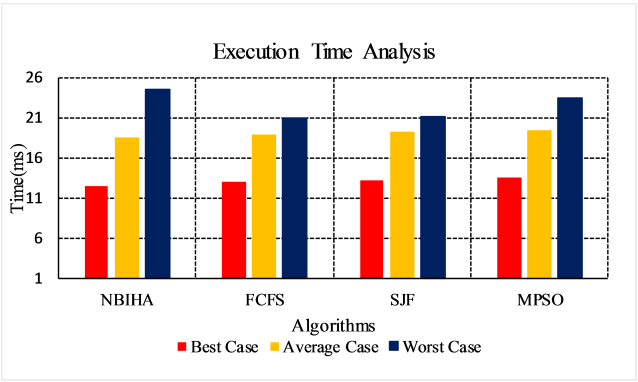


FIGURE 11. Comparison of execution time of algorithms in best, average and worst scenario.

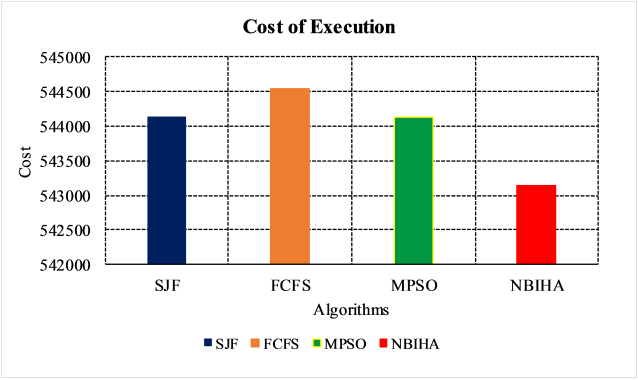


FIGURE 12. Cost analysis of algorithms.

the best two matches from each group of resources exactly match with the upcoming demand of tasks it presents as the best case. In the normal-case, the resource match is changing and so MPSO more efficiently than FCFS and SJF. Further, our approach has a shorter execution time in comparison to other approaches in both normal and best cases. If all the resources are matched with demands the hybrid approach is more efficient than when MPSO is considered independently.

Finally, we have evaluated NBIHA and other benchmark algorithms with respect to the cost of processing in fog environment. As can be seen in Fig 12, the cost of our proposed approach is lower than the other algorithms. In case of NBIHA, the load is balanced by scheduling tasks and this results in reducing the computational overhead and queuing time and therefore the cost is reduced. Conversely, in case of FCFS most of the tasks are sent to the cloud which increases cost of processing since the cost of processing in cloud is more than that of in fog computing environment.

E. PROCESSING COST ANALYSIS

It is a similar story in the case of SJF, because it increases time due to mismatch of the resource. In the case of MPSO, it increases the cost due to the exact match of the resources to the demand of the tasks and this in turn increases time and cost of processing due to waiting time. In NBIHA, the task scheduling and management of resources reduces waiting time and cost of processing as seen in Fig 12.

F. DISCUSSION

We have analyzed our proposed approach using following parameters: resource utilization (%), average response time (ms), energy consumption (j) and execution time (ms). We have compared the proposed algorithm with benchmark algorithms for scheduling, SJF and FCFS. We have also compared our proposed approach with the particle swarm optimization (PSO) meta-heuristic approach in the case of resource management and allocation to fog nodes. The results from the simulations show that our proposed approach has better nt results in the case of resource utilization, average response time and energy consumption when compared to other benchmark state-of-the-art algorithms. In the case of execution time, if it does not gets a match from the resource pool it increases execution time.

VII. CONCLUSION

Fog computing is a paradigm that aims to bring the benefits of cloud computing to the very edge of the network, where latency overheads should be much lower. This paper aims to provide effective resource utilization in cloud-fof IoT systems by proposing a novel hybrid resource management approach, named NBIHA. The proposed scheme balances the load among fog nodes and manages available fog resources. Given this, the contributions of the paper are two-fold - task scheduling and resource allocation. MPSO is used to schedule the tasks, resulting in efficient balancing of the load among fog nodes. A hybrid of bio-inspired algorithms is used to achieve efficient resource allocation. The simulation results show that the proposed approach optimizes the resource utilization and reduces the average response time when compared with state-of-the-art benchmark and conventional scheduling algorithms. In the future, we intend to utilize reinforcement learning techniques for managing resources in the fog-IoT environment.

REFERENCES

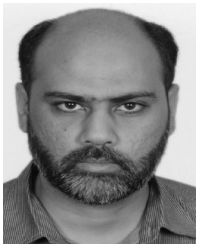
[1] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," in *Internet of Everything*. Singapore: Springer, 2018, pp. 103–130.
[2] S. Sarkar and S. Misra, "Theoretical modelling of fog computing: A green computing paradigm to support IoT applications," *IET Netw.*, vol. 5, no. 2, pp. 23–29, 2016.
[3] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the Internet of Things realize its potential," *Computer*, vol. 49, no. 8, pp. 112–116, Aug. 2016.
[4] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, "Fog computing for the Internet of Things: A survey," *ACM Trans. Internet Technol.*, vol. 19, no. 2, p. 18, 2019.
[5] H. A. Khattak, H. Arshad, S. ul Islam, G. Ahmed, S. Jabbar, A. M. Sharif, and S. Khalid, "Utilization and load balancing in fog servers for health applications," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, p. 91, 2019.
[6] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 1826–1857, 3rd Quart., 2018.
[7] D. Willis, A. Dasgupta, and S. Banerjee, "ParaDrop: A multi-tenant platform to dynamically install third party services on wireless gateways," in *Proc. 9th ACM Workshop Mobility Evolving Internet Archit.*, 2014, pp. 43–48.
[8] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. Workshop Mobile Big Data*, 2015, pp. 37–42.
[9] F. Bhatti, M. A. Shah, C. Maple, and S. Ul Islam, "A novel Internet of Things-enabled accident detection and reporting system for smart city environments," *Sensors*, vol. 19, no. 9, p. 2071, 2019.
[10] A. Toor, S. ul Islam, G. Ahmed, S. Jabbar, S. Khalid, and A. M. Sharif, "Energy efficient edge-of-things," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, p. 82, 2019.
[11] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of Internet of Things," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 46–59, Jan./Mar. 2018.
[12] S. Wang, T. Lei, L. Zhang, C.-H. Hsu, and F. Yang, "Offloading mobile data traffic for QoS-aware service provision in vehicular cyber-physical systems," *Future Gener. Comput. Syst.*, vol. 61, pp. 118–127, Aug. 2016.
[13] S. Bitam, S. Zeadally, and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm," *Enterprise Inf. Syst.*, vol. 12, no. 4, pp. 373–397, 2018.
[14] Y. Kong, M. Zhang, and D. Ye, "A belief propagation-based method for task allocation in open and dynamic cloud environments," *Knowl.-Based Syst.*, vol. 115, pp. 123–132, Jan. 2017.
[15] B. Ottenwälder, B. Koldehofe, K. Rothermel, and U. Ramachandran, "Migcep: Operator migration for mobility driven distributed complex event processing," in *Proc. 7th ACM Int. Conf. Distrib. Event-Based Syst.*, 2013, pp. 183–194.
[16] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: Architecture, key technologies, applications and open issues," *J. Netw. Comput. Appl.*, vol. 98, pp. 27–42, Nov. 2017.
[17] H. T. T. Binh, D. B. Son, P. A. Duc, and B. M. Nguyen, "An evolutionary algorithm for solving task scheduling problem in cloud-fog computing environment," in *Proc. 9th Int. Symp. Inf. Commun. Technol.*, 2018, pp. 397–404.
[18] S. Domanal, R. M. Guddeti, and R. Buyya, "A hybrid bio-inspired algorithm for scheduling and resource management in cloud environment," *IEEE Trans. Services Comput.*, to be published.
[19] L. Gu, "Cost efficient resource management in fog computing supported medical cyber-physical system," *IEEE Trans. Emerg. Topics Comput.*, vol. 5, no. 1, pp. 108–119, Dec. 2017.
[20] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016.
[21] J. Oueis, E. C. Strinati, and S. Barbarossa, "The fog balancing: Load distribution for small cell cloud computing," in *Proc. IEEE 81st Veh. Technol. Conf. (VTC Spring)*, May 2015, pp. 1–6.
[22] S. Agarwal, S. Yadav, and A. K. Yadav, "An efficient architecture and algorithm for resource provisioning in fog computing," *Int. J. Inf. Eng. Electron. Bus.*, vol. 8, no. 1, p. 48, 2016.
[23] S. Ningning, G. Chao, A. Xingshuo, and Z. Qiang, "Fog computing dynamic load balancing mechanism based on graph repartitioning," *China Commun.*, vol. 13, no. 3, pp. 156–164, Mar. 2016.
[24] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Comput.*, vol. 4, no. 2, pp. 26–35, Mar./Apr. 2017.
[25] M.-Q. Tran, D. T. Nguyen, V. A. Le, D. H. Nguyen, and T. V. Pham, "Task placement on fog computing made efficient for iot application provision," *Wireless Commun. Mobile Comput.*, vol. 2019, Jan. 2019, Art. no. 6215454.
[26] S. B. Akintoye and A. Bagula, "Improving quality-of-service in cloud/fog computing through efficient resource allocation," *Sensors*, vol. 19, no. 6, p. 1267, 2019.
[27] M. Verma, N. Bhardwaj, and A. K. Yadav, "Real time efficient scheduling algorithm for load balancing in fog computing environment," *Int. J. Inf. Technol. Comput. Sci*, vol. 8, no. 4, pp. 1–10, 2016.
[28] Q. Li, J. Zhao, Y. Gong, and Q. Zhang, "Energy-efficient computation offloading and resource allocation in fog computing for Internet of everything," *China Commun.*, vol. 16, no. 3, pp. 32–41, Mar. 2019.
[29] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," *Egyptian Informat. J.*, vol. 16, no. 3, pp. 275–295, 2015.
[30] K.-C. Lin, Y.-H. Huang, J. C. Hung, and Y.-T. Lin, "Modified cat swarm optimization algorithm for feature selection of support vector machines," in *Frontier and Innovation in Future Computing and Communications*. Dordrecht, The Netherlands: Springer, 2014, pp. 329–336.

[31] A. R. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Future Gener. Comput. Syst.*, vol. 91, pp. 407–415, Feb. 2019.
[32] L. Zhang, Q. Fu, J. Chen, H. Bai, and X. Zhou, "A modified particle swarm optimization algorithm—CPSODE," in *Proc. 29th Chin. Control Decis. Conf. (CCDC)*, 2017, pp. 6659–6663.
[33] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Int. Conf. Evol. Comput., IEEE World Congr. Comput. Intell.*, May 1998, pp. 69–73.
[34] B. Santosa and M. K. Ningrum, "Cat swarm optimization for clustering," in *Proc. Int. Conf. Soft Comput. Pattern Recognit.*, Dec. 2009, pp. 54–59.
[35] S.-C. Chu, P.-W. Tsai, and J.-S. Pan, "Cat swarm optimization," in *Proc. Pacific Rim Int. Conf. Artif. Intell.* Malacca, Malaysia: Springer, 2006, pp. 854–858.
[36] S. Svorobej, P. T. Endo, M. Bendechache, C. Filelis-Papadopoulos, K. M. Giannoutakis, G. A. Gravvanis, D. Tzovaras, J. Byrne, and T. Lynn, "Simulating fog and edge computing scenarios: An overview and research challenges," *Future Internet*, vol. 11, no. 3, p. 55, 2019.
[37] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments," *Softw., Pract. Exper.*, vol. 47, no. 9, pp. 1275–1296, 2017.
[38] R. Mahmud and R. Buyya, "Modelling and simulation of fog and edge computing environments using iFogSim toolkit," in *Fog and Edge Computing: Principles and Paradigms*. Hoboken, NJ, USA: Wiley, 2019, ch. 17, pp. 1–35.

HINA RAFIQUE received the B.Sc. degree in software engineering from COMSATS University Islamabad, Pakistan, in 2017, where she is currently pursuing the master's degree in software engineering with the Department of Computer Science. She has been with the Virtual University of Pakistan as an Instructor, since 2018. Her Research interests include the domain of cloud computing, fog computing, and software process improvement.

MUNAM ALI SHAH received the B.Sc. and M.Sc. degrees in computer science from the University of Peshawar, Pakistan, in 2001 and 2003, respectively, the M.S. degree in security technologies and applications from the University of Surrey, U.K., in 2010, and the Ph.D. degree from the University of Bedfordshire, U.K., in 2013. Since 2004, he has been a Lecturer with the Department of Computer Science, COMSATS Institute of Information Technology, Islamabad, Pakistan. He is the author of over 50 research articles published in international conferences and journals. His research interests include MAC protocol design, QoS, and security issues in wireless communication systems. He received the Best Paper Award of the International Conference on Automation and Computing, in 2012.

SAIF UL ISLAM received the Ph.D. degree in computer science from the University Toulouse III Paul Sabatier, France, in 2015. He is currently an Assistant Professor with the Department of Computer Science, Dr. A. Q. Khan Institute of Computer Science and Information Technology, Rawalpindi, Pakistan. Previously, he has served as an Assistant Professor with the COMSATS University, Islamabad, Pakistan, for three years. He has been a part of the European Union-funded research projects during his Ph.D. He was a focal person of a research team at COMSATS, working in O2 project in collaboration with CERN Switzerland. His research interests include resource and energy management in large-scale distributed systems (edge/fog, cloud, content distribution networks (CDN)), and the Internet of Things (IoT).



TAHIR MAQSOOD received the M.Sc. degree in computer networks from Northumbria University, U.K., in 2007, and the Ph.D. degree in computer science from the COMSATS Institute of Information Technology, Pakistan, in 2017. He is currently an Assistant Professor with the COMSATS Institute of Information Technology, Abbottabad, Pakistan. His research interests include task scheduling, application mapping, energy-efficient systems, and network performance evaluation.

SULEMAN KHAN received the Ph.D. degree (Hons.) from the Faculty of Computer Science and Information Technology, University of Malaya, Malaysia, in 2017. He was a Faculty Member with the School of Information Technology, Monash University, Malaysia, from 2017 to 2019. He is currently a Faculty Member with the Department of Computer and Information Sciences, Northumbria University, Newcastle, U.K. He has published over 45 high-impact research articles in reputed international journals, including the IEEE Communications Surveys and Tutorials, the *ACM Computing Surveys*, and the IEEE Transactions on Intelligent Transportation Systems. He has published in the 2018 Local Computer Networks Conference. His research areas include, but are not limited to, network forensics, software-defined networks, the Internet of Things (IoT), cloud computing, and vehicular communications.

CARSTEN MAPLE is currently the Director of research in cyber security and a Professor of cyber systems engineering with the Cyber Security Centre, University of Warwick, where he leads the GCHQ-EPSRC Academic Centre of Excellence in Cyber Security Research. He is the Privacy and Trust Stream Lead and has led the project constellation in transport and mobility with PETRAS, the U.K. research hub for cyber security of the Internet of Things. He is also a Principal or Co-Investigator for a number of projects in cyber security. He is currently, or has recently been, funded by a range of sponsors, including EPSRC, EU, DSTL, the South Korean Research Agency, and Innovate UK, and private companies. He has published over 200 peer-reviewed papers and has provided evidence and advice to governments and organizations across the world, including being a high-level scientific advisor for cyber security to the European Commission. He is a member of various boards and expert groups. He is also a Fellow of the Alan Turing Institute. He is the Immediate Past Chair of the Council of Professors and Heads of Computing, U.K.