IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 00.0000/ACCESS.2019.DOI

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], CARSTEN MAPLE [5]

[1]Department of Computer Science, COMSATS University Islamabad, Pakistan, Park Road Tarlai Kalan, Islamabad 44550, Pakistan (e-mail: hinarafiquekhan@gmail.com, mshah@comsats.edu.pk)
[2]Dr. A. Q. Khan Institute of Computer Science and Information Technology, Rawalpindi, Pakistan (e-mail: saiflu2004@gmail.com; saif@kicsit.edu.pk)
[3]Department of Computer Science, COMSATS University Islamabad, Abbottabad Campus, Abbottabad, Pakistan (e-mail: tmaqsood@ciit.net.pk)
[4]Monash University Malaysia, Malaysia (e-mail: suleman.khan@monash.edu)
[5]WMG, University of Warwick, Coventry, UK. CV4 7AL (e-mail: cm@warwick.ac.uk)

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

"This work is supported by the Alan Turing Institute under EPSRC grant EP/N510129/1."

ABSTRACT Fog computing has emerged as a revolutionary paradigm to serve massive data in the Internet of Things (IoT) environment. It is a derivative of cloud computing that provides cloud-like services at the edge of the network. Subsequently, it resolves the significant issue of higher delay faced in cloud-IoT paradigm. According to the literature, the inefficient scheduling of users tasks in fog computing may result in higher delays in comparison to 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 that cannot be solved by any specific algorithm to reach an ideal solution. Hence, it requires the optimal and efficient techniques to cater to the issues of latency, response time and efficient resource utilization of the available fog resources at the edge of the network. Given this, we proposed 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, MPSO is used to schedule the tasks among fog devices and the hybrid of 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 available fog resources. The simulations are carried out 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; and Resource Management

I. INTRODUCTION

Fog computing is an extended type of distributed computing which lays between cloud datacenters and IoT devices. It provides storage and computing services closer to the end devices [1]. It consists of networking components, proxy servers, switches, setup boxes, base stations and routers. These components have their own respective computing, storage and networking services. It is considered as an extension of cloud computing and was initially introduced by Cisco [2] to subdue the limitations of cloud. Many technologies are using fog computing like real time efficient systems, health care systems, and augmented reality and gaming [3]–[5].

Cloud computing is providing hardware and software services to users on pay as you go model. It is a combination of cluster and grid computing in which resources are combined at a central point for high level performance. Fog computing is an advanced form of distributed computing services. It provides better performance and utilizes devices processing and storage capacities at the edge of the network to handle

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2924958, IEEE Access

user requests. It does not replace the cloud computing. If fog computing is used with cloud computing, it reduces delay, provides fast computing and reduces cost of processing [6]. Fog nodes are basically edges of the network at which resources are provided for the service utilization by the infrastructure of fog computing. There are two types of fog nodes, i.e. resource-poor devices that includes routers, set-top-units, wireless access points(WAP) [7] end devices, switches, and stations whereas IOx cables and cloudlets are resource-rich machines. Resource-rich machines like cloudlet enhance cloud data centers which are at small scale. The edge of the Internet comprises of nearby mobile devices. A Cloudlet is manufactured, which presents at the edge of the network it contains resources and provides processing to mobile applications. It gives amazing processing to mobile devices with low latency [8]. In Fig 1, the basic architecture of fog and cloud computing is illustrated.

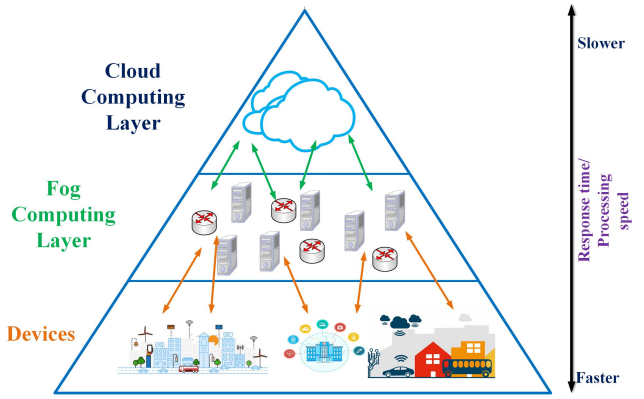

FIGURE 1: Cloud and fog computing architecture.

The academic and industrial communities are accepting the trend of massive connectivity of everything, everywhere. They are paying more attention towards the IoT-based smart X solutions - smart home, smart cities, smart transportation, smart military operations, smart metering, wearable computing, and wireless sensor network (WSN) [9]. In past few years, IoT grabs the attention of researchers and becoming the future of the Internet. In 2015, International Data Corporation (IDC) predicted expansion of IoT up to 14% from current situation [8].

Traditionally, cloud computing is used for computational processing and storage for the IoT applications. With the passage of time, IoT devices are increasing that leads to the explosive increase in energy consumption and performance degradation [10]. Moreover, the high latency in cloud-IoT paradigm makes it less practical for delay sensitive IoT applications. In this context, energy and performance aware computational and storage services have become highly critical [11]. To overcome these issues, cloud computing is extended and forms the edge or fog computing that performs the processing at the end of the network. In 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 will be utilized by the computing nodes. Many IoT applications that deal with the real-time scenarios use the edge computing nodes as a priority [11], [12], [14].

Efficient resource management in IoT-based fog computing environment is one of the main issues due to the dynamic nature of bandwidth, storage, computation, and latency of end devices. For example, we can keep the record of in-duty ambulance or vehicle in the connected vehicle scenario and on the other hand, we can also control the smart traffic lights for the ambulance in the case of an emergency. We must perform the necessary tasks to achieve the Quality of Service (QoS) requirement such as delay. The resources should be available for service mobility. Ottenwälder et al. [15] proposes the placement and migration method named as MigCEP for the resources of 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 the fog computing to be effectively used with IoT for mobile crowd sourcing or sensing.

After extending the cloud to the network edge, efficient resource management has become a challenging task [16]. Hence, 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 scheme named 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 proficient resource utilization and improved performance. According to the proposed approach, 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. If it does not find any match from fog devices then it sends the task to the cloud. The main contributions of this paper are summarized below:

- The proposition of a novel bio-inspired hybrid algorithm (NBIHA) - MPSO and MCSO
- The proposed algorithm has two major contributions - 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 problem formulation. Section IV holds the algorithms of the proposed technique. Section V describes 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, conclusion and future work are presented.

II. RELATED WORK

In [13], Bee Life Algorithm (BLA) is used to assign a bag of jobs to fog or edge nodes which are situated at the corner or edge of the network. It focuses on the reduction of execution time and the memory required by the overall mobile tasks executed on fog nodes. It is inspired by bee life algorithm. In Time Cost-aware Scheduling Evolutionary Algorithm - Genetic Algorithm is used to schedule tasks on fog and cloud according to the requirement of tasks. It is maintaining trade-off between cost of execution and time of execution. This paper mainly focuses on to resolve scheduling problem for a set of tasks (BoT) applications in hybrid environment of cloud-fog computing. The proposed TCaS algorithm is based on an evolutionary algorithm's class genetic algorithm. Its performance is evaluated with different data-set of tasks on fog and cloud devices. Its improved criteria are a trade-off between time and cost and user satisfaction. A three-layered model is presented to productively allocate resources in the cloud-fog environment. The system is divided into three parts client layer, fog layer, and cloud layer. The algorithm is implemented in client fog layer and the remaining requirements were accommodated through the cloud. The improved factors 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. It allocates resources before processing [17].

The proposed model has two parts job allocation to Virtual Machines (VMs) and allocation/management of resources in the system. Two bio-inspired algorithms are used in it - Modified Particle Swarm Optimization (MPSO) for job allocation and MCSO for resource assignment/allocation and management. It has improved utilization of available resources, reliability and average response time. It is cloud-based model. It improves all criteria for cloud-based applications [18]. Another idea of combining fog computing into Medical Cyber-Physical System (MCPS) which is called as FCMCPS. In this new model, it is providing cost efficient solution for managing base station links, task division, and VM placement. An issue is confined as mixed-integer non-linear programming (MINLP) with respective to base station association, VM allocation, and task assignment. A mixed-integer linear programming (MILP) is used to reduce the complexit of problem. A two-phased Linear programming based heuristic algorithm is used to resolve this problem [19].

A framework is used to explore the problem of imbalance between consumed power and delay for workload distribution in the cloud-fog environment. This framework is based on mathematical structure. A primal problem is developed and approximated into three small problems of respective subsystems to provide the optimized solution of communication latency in the system. These problems are solved by using the Hungarian method. Simulations have conducted to show the reduced communication latency in fog environment in contrast with the cloud environment. The shortcoming of this system is it performed optimization in a centralized manner, not in a distributed manner. It is difficult to use in fog infrastructure where the system is complex and information exchange and communication overhead are high [20]. A customized innovative algorithm is proposed and described to form clusters of resources and load balancing for fog computing. It allocates resources to meet the expectations of user requests such that to fulfill them and also keeps low the power consumption and process complexity. To fulfill this work a two-step method is proposed to allocate resources. In the first stage, resources are distributed to smart cells by using specified scheduling rule whereas in second stage clusters are formed for unfulfilled requests. In this algorithm, the user can change metrics, scheduling rules, and clustering objectives according to the requirement of application and network. This system is very difficult to use for complex fog infrastructure [21].

An approach is proposed in which policies were designed to assign the tasks produced at the mobile users on edge cloud servers. These approaches provide a trade-off between optimal power-delay. Delay faced by mobile users in waiting and execution of request sent. The system is evaluated with theoretical analysis of policies. Two main contributions are done in the proposed method. First, Markov decision procedure is utilized to find the optimal arrangement for the algorithm of task scheduling whereas in the second approach to remove the complexity of the first proposed approach an effectively implementable index policy is proposed. Simulations are utilized to assess the viability of the proposed strategies. The disadvantage of this methodology is it has computational overhead and it has time lining framework. It performs static task assignment [8]. In this approach, the author used graph theory concepts with fog characteristics and constructed a model for load balancing. It uses 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 a single or multiple VM nodes according to the demand of resources made by the task. This model does not propose dynamic load balancing [23].

In Table 1, we have summarized the existing techniques, algorithms and their limitations in this context. Our goal is to provide an intelligent solution to task scheduling, load balancing, and resource management in fog computing paradigm. Conventional scheduling algorithms are used to solve this issue; however an intelligent and efficient system is required for this purpose. The rationale of using a modified hybrid algorithm is that we are scheduling tasks and managing resources at the same time. The literature shows that in cloud computing MPSO is used to improve the resource utilization [29]; whereas, in modified form of MCSO the improved criteria is that it enhances and improves the search efficiency within the search space as we are finding a best-fit resource for task processing so it fulfills our requirement

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2924958, IEEE Access

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	Fog-Cloud 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.	Fog-Cloud 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, 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.	TTask 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

[30].

III. SYSTEM ARCHITECTURE

In this paper, a fog system is assumed which consists of fog and cloud processing nodes. Our system architecture is comprised of three layers - client module, scheduler, and fog devices and cloud datacenters. The working of our proposed NBIHA architecture is that all the client requests will be received by the scheduler. The scheduler will perform the algorithms and find the best resource match for jobs on the basis of CPU and memory demand of the tasks. It schedules the task using MPSO algorithm approach by finding the global best (GB) and performs load balancing where as a new algorithm composed of (MPSO and MCSO) algorithm will be performed for the management of resources on the basis of the fitness function. We have implemented our system model using the simulator. Different simulators are available for fog environment. We are using iFogSim simulator to evaluate performance of our proposed NBIHA approach. In this system, we took three basic machines client machine, fog devices, and cloud servers. We have configured them and implemented our algorithms for evaluation purpose. The evaluation has been discussed in next sections. Fig 2 shows the model of our proposed system.

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2924958, IEEE Access

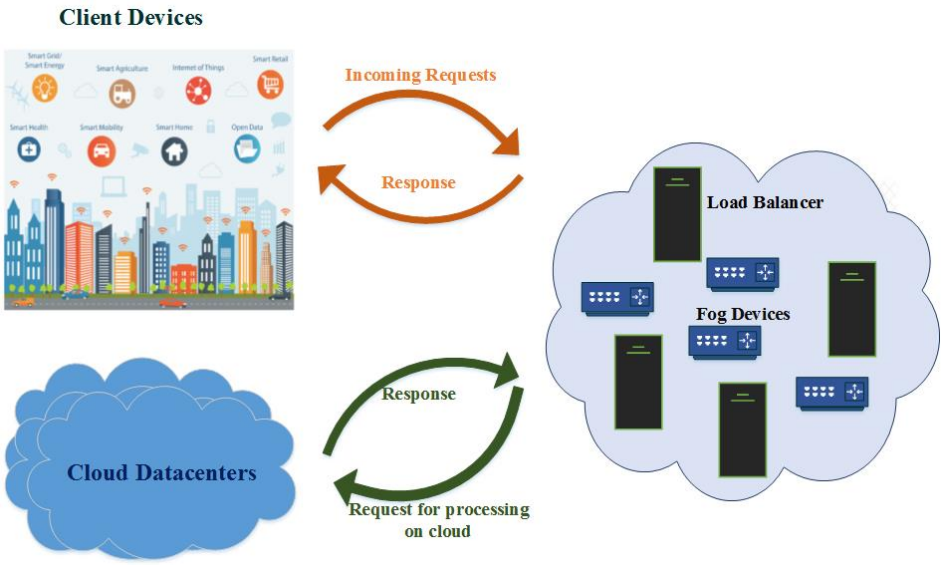

FIGURE 2: NBIHA system model.

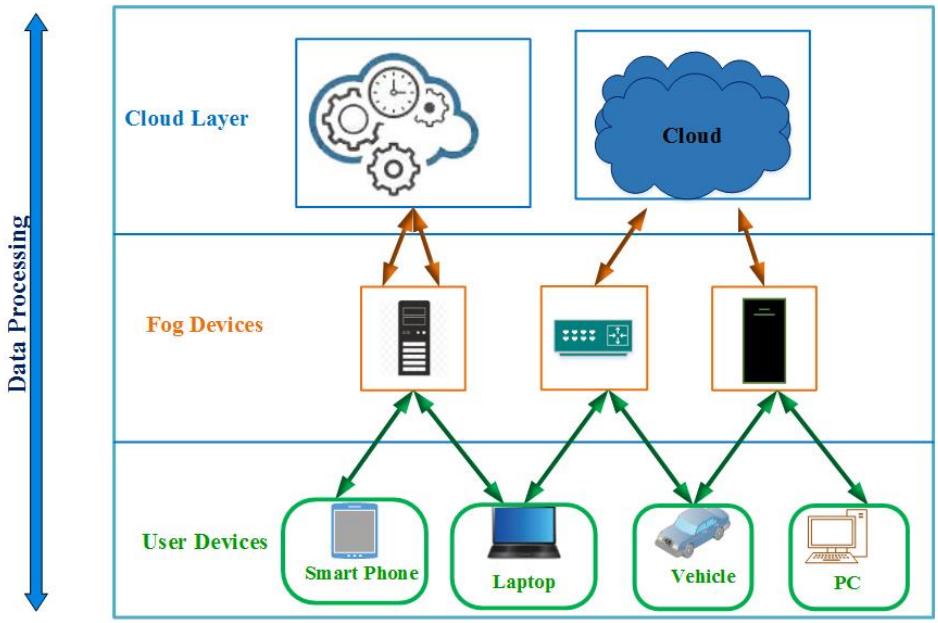

FIGURE 3: NBIHA system model.

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 datacenters.

Nowadays, number of IoT users are increasing day by day which in return increases the load on fog and cloud nodes for processing. It requires an efficient approach to schedule tasks and to manage fog and cloud resources. To overcome these issues, we have proposed NBIHA a hybrid of MPSO and MCSO for load balancing and resource management. The rationale of using a modified hybrid algorithm is that we are scheduling tasks and managing resources at the same time. The literature shows that in cloud computing MPSO is used to improve the resource utilization [29]; whereas, in modified

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2924958, IEEE Access

form of MCSO the improved criteria is that it enhances and improves the search efficiency within the search space as we are finding a best-fit resource for task processing so it fulfills our requirement [30].

B. PROBLEM FORMULATION

In our proposed approach, we designed the task load balancing in the fog computing environment. We describe a task as a method that defines a service demand made by a fog computing user that could be any of these, a mobile user, web user, and other Internet users. Incoming requests, i.e. tasks tn $\{t1, t2, t3, t4 \ldots, tn\}$ are to be scheduled on available fog devices, i.e. fk $\{f1, f2, f3, f4 \ldots, fm\}$ 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

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

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 problem in cloud computing. We know that fog computing is the derivative of cloud computing we can use best performing metaheuristics techniques in it as well. The findings in [29] show that it optimizes the energy usage. The literature [31] depicts that by using scheduling techniques in cloud computing it improves the performance in terms of processing, time, and resource utilization.

1) Illustration of MPSO Algorithm

In this algorithm, particles are considered where each particle has two properties - position $\{xi1, x12, xi3 \ldots, xin\}$ and direction which is represented by the velocity of the particle $\{vi1, vi2, vi3 \ldots, vin\}$ in Dimension X. In this process, each particle has its personal best position found by itself, other than this a global best is found by a particle among all of the particles. Fig 4 shows activity of 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 the 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 have used seeking mode of MCSO in our algorithm. In seeking mode of MCSO, the condition of the cat is depicted, which is 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 considering (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 is 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 factors of the seeking mode. SPC keeps record of the point where cat stands, is whether a possible point to move to. It is a Boolean variable. If estimation of SPC is true or false, it does not affect SMP [34], [35]. 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.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2924958, IEEE Access

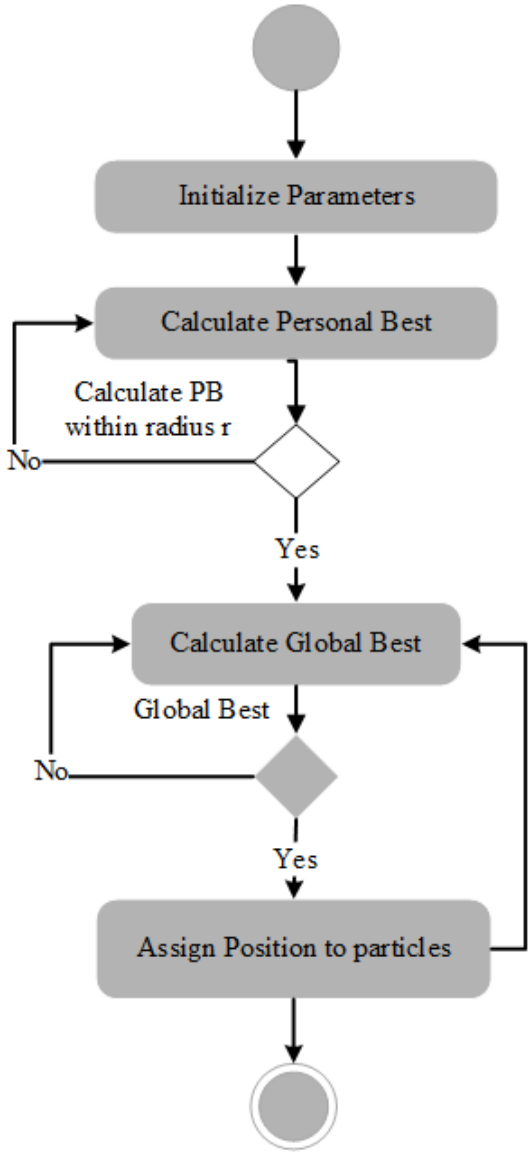

FIGURE 4: Data flow of MPSO

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

For minimum solution FS_a=FS_max

IV. NBIHA FOR FOG COMPUTING JOB SCHEDULING AND RESOURCE MANAGEMENT

Several Modified Particle Swarm Optimization (MPSO) algorithms have been proposed after Particle Swarm Optimization (PSO) algorithm. Many authors tried to get the best optimal function by modifying the PSO algorithm. In this approach, we will use these modifications to resolve task allocation and load balancing in a 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=$\{f1, f2, f3, f4 \ldots, fj\}$ and there are incoming requests $\{x1, x2, x3, x4 \ldots, xn\}$ for processing. Each cluster finds the personal best (LB) which is considered as the least loaded fog device. From these least loaded fog devices, the smallest is considered as global best (GB) and it is assigned to process the requested task. If the match is not found the task is sent to the cloud for processing.

Algorithm 1 Load balancing by scheduling tasks Using MPSO

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

for *incoming requests* $x1, x2, x3, ..., xn$ **do**
  for *Cz*= $c1, c2, c3, ..., cz$ **do**
    $Cz \leftarrow F(leastloaded)$
    $Assignlbz \leftarrow Cz$
  end
  Assign gb = leastlbz
  Allocate next task $x \leftarrow$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

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 such as F= $\{f1, f2, f3, f4 \ldots, fj\}$ and there are incoming requests $\{x1, x2, x3, x4 \ldots, xn\}$ for processing. In the start, the resources are distributed based on the demand of the tasks. After that from the remaining resources the find the two best values (bestfitres1 and bestfitres2). Neededres contains the required resources for incoming requests. It compares Neededres with the bestfitres1 if it matches it assigns it for processing and if it does not match then it compares with bestfit res 2. If it gets match it send a request for processing otherwise resources are sent to res pool and requests are sent to cloud for processing.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2924958, IEEE Access

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* $x1, x2, x3, ..., xn$ **do**
  $Resdemand \leftarrow totalresourcesrequiredbyrequests$
  $NeededRes \leftarrow resourcesrequiredbyeachrequest$
  for $F\ f1, f2, f3, fj$ **do**
    $sumres \leftarrow sumres + Resdemand$
  end
  if *start* **then**
    $Respool \leftarrow Respool - sumres$
  else
    for *all Cluster size,* $Cz\ c1, c2, c3, ..., cz$ **do**
      $bestfitres1 \leftarrow firstbestcz$
      $bestfitres2 \leftarrow secondbestcz$
    end
    for *all Fd =* $fd1, fd2, fd3, ., fdj$ **do**
      for *all Cluster size,* $Cz = c1, c2, c3, ..., cz$ **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

In algorithm 2, we have used the exact matches to find the resource for processing this has two drawbacks if it does not find bestfit exact match then it sends requests to cloud which in result increases the computational overhead.

2) Resource Allocation and Management using NBIHA

In Algorithm 3, we have proposed the NBIHA approach to allocate and manage fog devices. This algorithm will overcome the drawbacks of the MPSO algorithm. In this algorithm, we have created a resource pool in which all the resources are saved. We have fog devices such as F= $\{f1, f2, f3, f4 \ldots, fj\}$ and there are incoming requests $\{x1, x2, x3, x4 \ldots, xn\}$ for processing. In the start, the resources are distributed based on the demand of the tasks. After that from the remaining resources the find the two best values (bestfitres1 and bestfitres2). Needed res contains the required resources for incoming requests. It compares needed res with the bestfit res1 if it matches it assigns it for processing and if does not match then it compares with bestfitres2. If it gets match it sends a request for processing otherwise resources are sent to respool and requests are sent to cloud for processing. Seeking memory pool of MCSO is utilized to store the resources other than bestfitres1 and bestfitres2. Four distinct types of memory pools are used for the purpose of using the MCSO of seeking mode. Based on the SMP, the cat scans for the accurate counterpart for the future resource demand of the request made by the task. In the event that there is a match, at that point status is retained within scope SRD. If SRD has another update, at that point fog device starts executing the assignment and this status is used to as change the CDC. Each update of SMP of MCSO effects change in position of Cats and it is maintained in SPC. In MCSO result of seeking mode is used as input to the next mode and the methodology of algorithm 2 is used. 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. It will remove the drawbacks of an exact match which results in computational overhead. It improves the execution time and average response time of the fog nodes in fog computing.

Fig 5 Shows the sequence of working of NBIHA in fog computing environment. User send the requests which are collected by the fog devices. Fog broker manages the fog devices and requests generated by the user. Fog broker divides the requests into the tuples. It schedules the tuples on the basis of MPSO and manages the fog devices as well. The tuples are then sent to fog devices and cloud devices as per requirement. After processing, fog broker checks the completion of task and compiles results and send back to user through fog devices, i.e. actuators.

V. PERFORMANCE EVALUATION

Extensive simulations are performed using iFogSim in a fog environment to assess the working of our proposed 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 developing the fog framework with the specifications

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2924958, IEEE Access

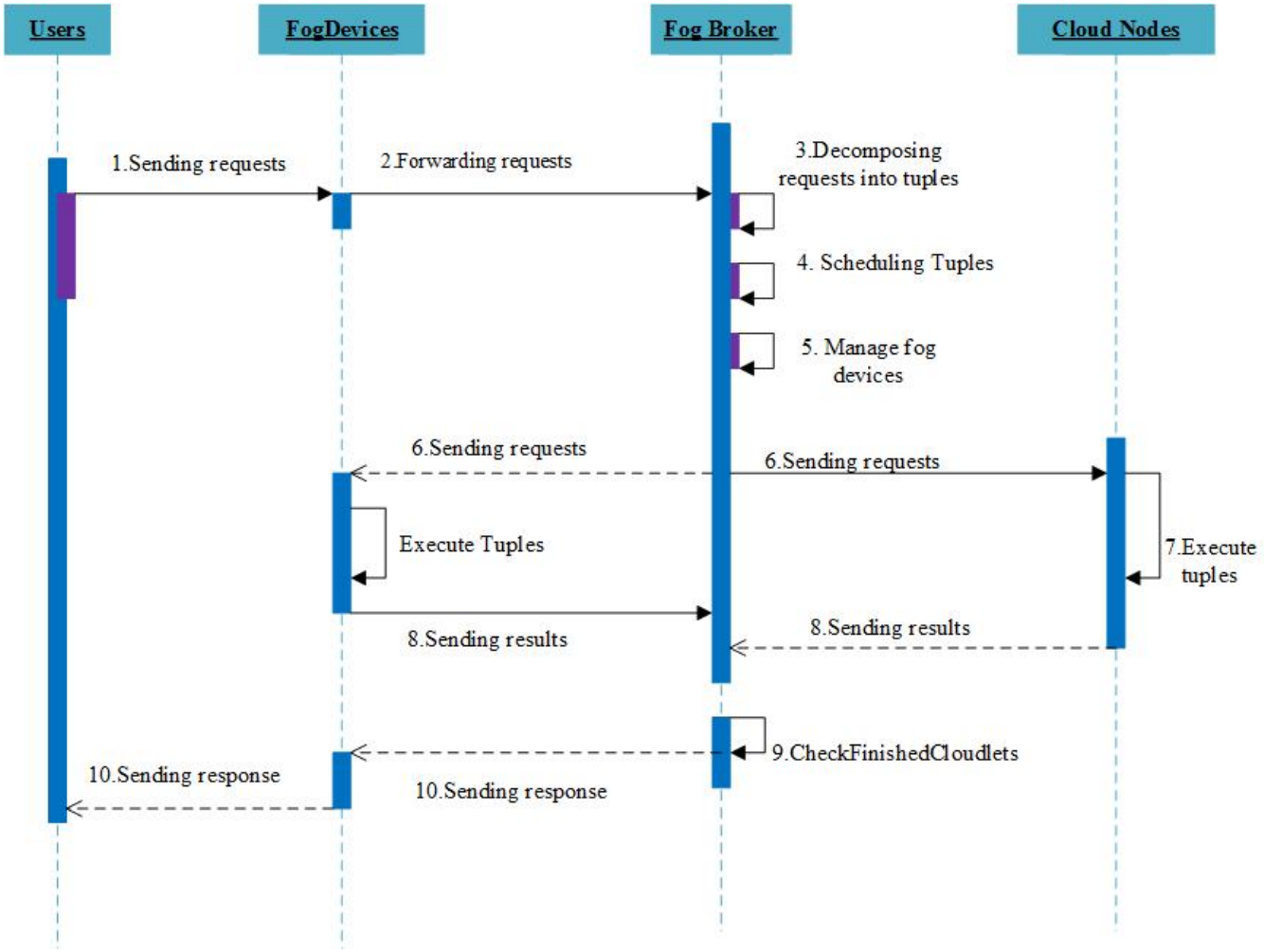

FIGURE 5: Sequence diagram of NBIHA

appeared in Table 2.The unit of cost is grid dollars used in simulations in replacement of real money. In the fog layer, fog nodes have restricted processing power, for example, switches, doors, workstations, or PCs. While in the cloud layer, servers or virtual machines in elite server farms are in charge of taking care of requests. In this manner, the preparing rate of cloud nodes is a lot quicker than fog nodes. Interestingly, the expense of utilizing resources in the cloud is more costly than in the fog.

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

The fog framework is responsible to execute all incoming requests from the client. Each request is divided into number of tasks known as tuples, which are broke down and evaluated processing that they required. Expecting that each task has a few qualities as number of directions, the memory required, input document size, and output. Dependent upon the rest of the weight of every request, the plan of the requests may vary for the most part in size. Subsequently, we have used 20 tasks to the 60 tasks were made to check out our proposed NBIHA approach. Every task in was created with

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2924958, IEEE Access

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* $x1, x2, x3, ..., xn$ **do**
  $Resdemand \leftarrow totalresourcesrequiredbyrequests$
  $NeededRes \leftarrow resourcesrequiredbyeachrequest$
  for *Fd* $f1, f2, f3, fj$ **do**
    $sumres \leftarrow sumres + Resdemand$
  end
  if *start* **then**
    $Respool \leftarrow Respool - sumres$
  else
    for *all Cluster size, Cz* $c1, c2, c3, ..., cz$ **do**
      $bestfitres1 \leftarrow firstbestofcz$
      $bestfitres2 \leftarrow secondbestofcz$
      $bestfitres3[] \leftarrow exceptfirstandsecondbestofcz$
    end
    for *all Fd* = $fd1, fd2, fd3, ., fdj$ **do**
      for *all Cluster size, Cz* = $c1, c2, c3, ..., cz$ **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

its characteristics following 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 preparing while others need more memory or data transfer capacity use, etc. There are issues in simulating fog and edge computing environments because 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 picked in light of the fact that it is created dependent on CloudSim, a distributed computing stage that has been broadly utilized and approved in various experiments throughout the years. 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. Hence, our criteria to evaluate the proposed approach are resource utilization and average response time. 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 fog and cloud system after scheduling and managing resources using bio-inspired heuristic algorithms.
- H1: The system with scheduled using NBIHA will have better power utilization and average response time of the resource.

VI. RESULTS

Results of our proposed approach are analyzed with respect to the efficient resource utilization, average response time, energy consumption and execution time. Moreover, the proposed algorithm is compared with benchmark algorithms for scheduling like shortest job first(SJF), first come first serve (FCFS). We have also compared our proposed approach with particle swarm optimization (PSO) metaheuristic approach in 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. From Fig 7, it is shown that the average response time of the proposed MPSO approach is smaller than all of the other. In FCFS, the incoming requests have to wait if the resources are busy because it allocates resources in the manner as it is coming whereas in SJF big incoming requests have to wait more because it is processing shortest job first. In our proposed

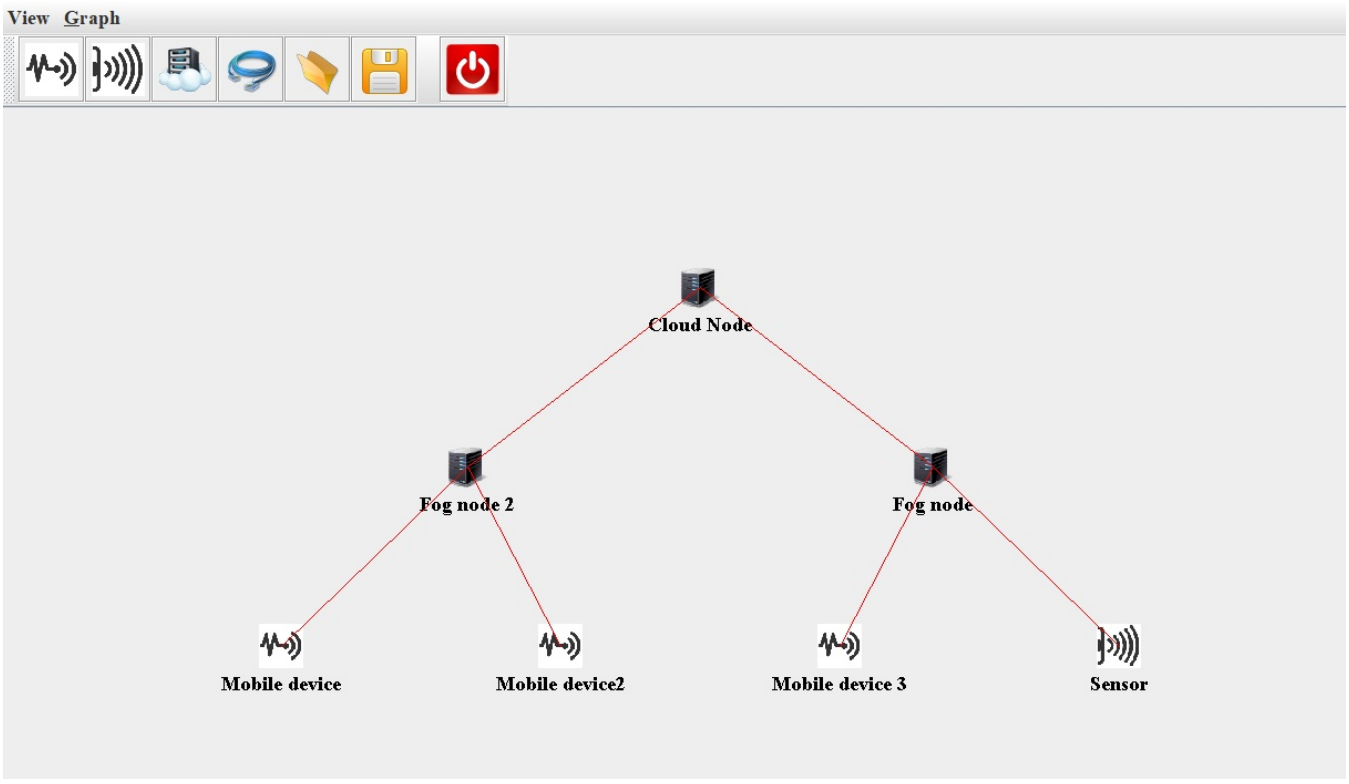

FIGURE 6: Fog computing environment topology designed in iFogSim.

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. In Fig 7, it has shown that MPSO uses all fog nodes equally whereas the other two algorithms use a smaller number of resources which increases the load. We have not considered MCSO and Hybrid approach for scheduling because it takes more time like this it uses both modes of the MCSO seeking mode takes more time that is why for scheduling only MPSO is used. We have evaluated our task scheduling and load balancing by considering resource utilization when we use the MPSO which efficiently schedules the tasks it increased the maximum and approximately equal utilization of the all fog nodes whereas in case of FCFS and SJF we can see that if it does not find match on fog node it send the task to cloud node which in return increases the usage cost.

The Fig 7 shows that there are fluctuation in the resource utilization while using different approaches. Since FCFS allocates resource on the basis of arrival of the demands of incoming requests. There is uneven utilization of resources in case of FCFS and SJF where as in MPSO it finds the best fit using Global best and balancing the load that is why resource utilization is evenly divided on all the fog nodes.

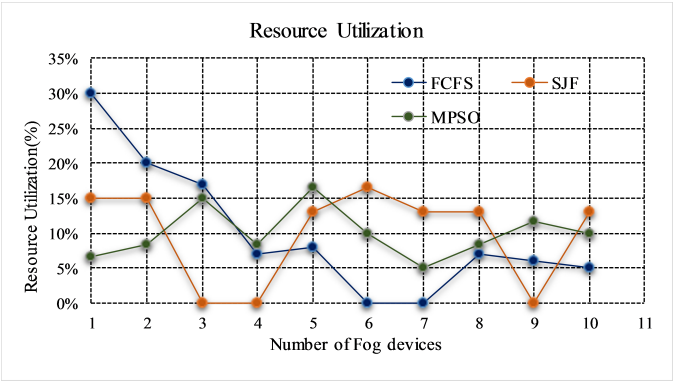

FIGURE 7: Resource utilization in terms of tasks performed by each fog device.

B. RESOURCE ALLOCATION AND MANAGEMENT

As explained earlier, we have formulated a problem in which incoming jobs request for the fog nodes for processing. A number of tasks are coming in sequential form. The first fog node is considered as a scheduler. In the start, fog nodes are given on the basis of the need using the MPSO algorithm. After that, we have created clusters of fog nodes. These resources will be managed using our proposed hybrid algorithm. Experiments are performed to evaluate the average response time (ART) in the sequential form. In Fig 8, we have utilized set of 10 and 20 fog devices in groups with 60 incoming tasks. For task scheduling, it uses global best (GB) method of MPSO. After that, it will use the MCSO approach to manage the resources. In this bestfit res3 will be compared for the future demand of the tasks. In Fig 8, it has been shown that among the MPSO, SJF, FCFS, and hybrid the hybrid

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2924958, IEEE Access

approach provides reasonable results with respect to average response time. To analyze the resource management part of our work, we have used the parameter of average response time. Average response time is considered as the time taken between the start and end of the processing of the task. We have analyzed it with different sets of tasks and fog devices. It produces consistent results in both cases. In first case, we have taken 10 fog devices whereas in second case we have taken 20 fog devices. The results are depicted in Fig 8.

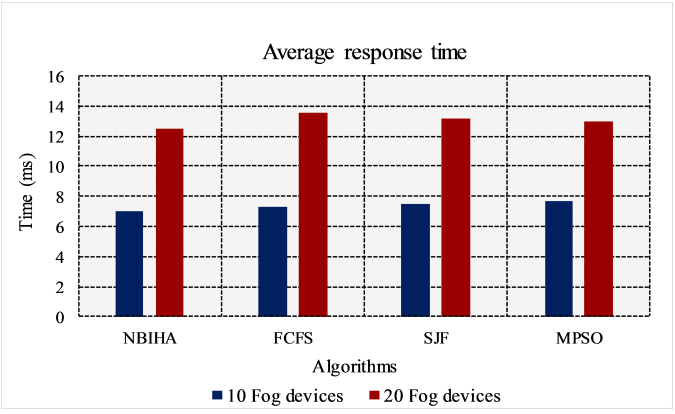

FIGURE 8: Average response time of algorithms on the basis of task completion time.

C. ENERGY CONSUMPTION

In Fig 9, we have evaluated our approach NBIHA with FCFS and SJF with respect to the energy consumption of fog devices. Energy consumption in case of NBIHA is less as compared to the other two approaches. In Fig 9, we have considered 24 fog devices to evaluate the energy consumption of resources by using NBIHA, FCFS and SJF. It can be seen in the Fig 9 that due to load balancing all nodes are being approximately equal that is why energy consumption in case of our proposed approach is less as compared to other two approaches. In FCFS, the placement of jobs is made on the basis of sequence of incoming requests whereas in SJF the shortest jobs are entertained earlier irrespective of the sequence. In our approach, the fitness value is determined on the basis of which the resource is allocated to the job for processing.

In Fig 10, we have used set of 5, 10, 15, and 20 fog devices and evaluated the performance of our proposed approach with respect to energy consumption. It is seen in Fig 10 that when we increase number of fog devices the usage of energy increase in all cases whereas in comparison with FCFS and SJF our proposed NBIHA has less energy consumption in all sets of fog devices.

D. EXECUTION TIME ANALYSIS

In Fig 11, three cases of all the proposed algorithms are considered best, average and worst. We explored different combinations regarding the tasks and fog devices with state-of-the-art conventional scheduling algorithms and NBIHA.

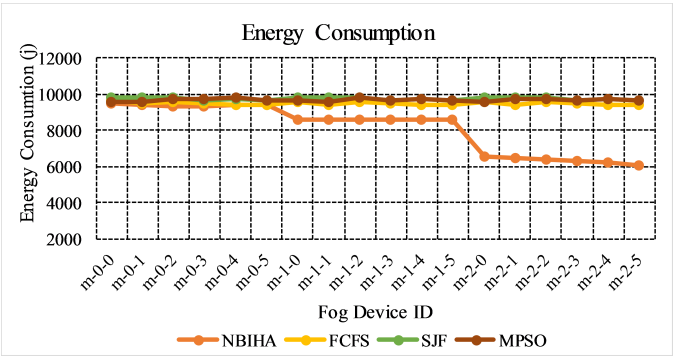

FIGURE 9: Energy consumption of algorithms with respect to each fog device.

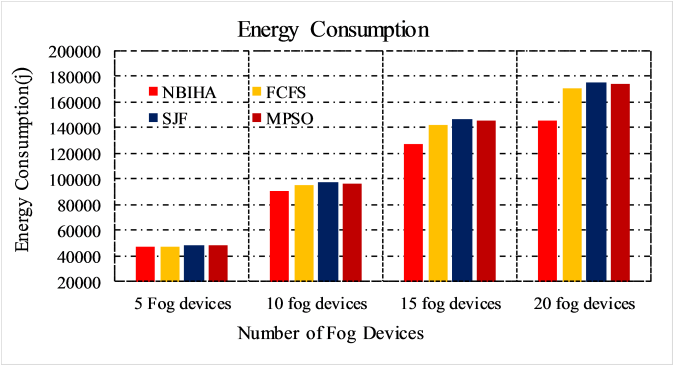

FIGURE 10: Energy consumption analysis of algorithms on the basis of 5,10,15 and 20 fog devices.

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 incoming request. These resources are fetched from buffer of cloud resources for more often than not, consequently the execution time of FCFS will proceed as before for all the depicted cases. In SJF, the resources requirement has been matched with the fog devices which have processing power to process the shortest job first. Since there would not be any ideal resource match for more often than due to incoming request demand or due to waiting of long jobs and hence SJF takes approximately the same execution time for all cases to execute tasks. The worst case scenario of execution time for MPSO, and NBIHA approach is continuously increasing because of the delay expanding a direct result of the deferral in the confound examination of assets from the fog devices to the future incoming request. In the MPSO, whenever the best two matches from each group of resources exactly match with the upcoming demand of tasks is considered best-case. In normal-case, the resource match is changing so that MPSO works in better and efficient way when compared to FCFS and SJF. Further, proposed NBIHA approach has less execution time in comparison with other approaches in both cases. However, if all the resource are matched with demands in case of NBIHA approach, it is considered more efficient

than, when MPSO considered independently. Hence, NBIHA approach, hybrid of MPSO and MCSO, is more efficient as far as of resource distribution and management in the fog environment is considered.

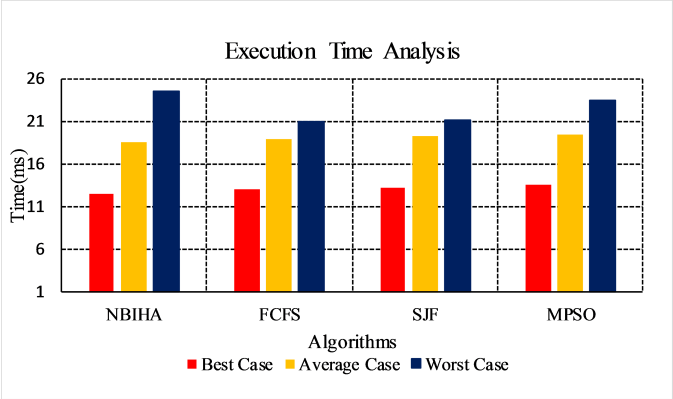

FIGURE 11: Comparison of execution time of algorithms in best, average and worst scenario.

Finally, we have evaluated NBIHA and other benchmark algorithm with respect to cost of processing in fog environment. As observed in Fig 12, the cost of our proposed approach is lesser as compared to other algorithms. In case of NBIHA, the load is balanced by scheduling tasks which in result reduces the computational overhead and queuing time and cost is reduced whereas in case of FCFS most of the tasks are sent to cloud which increases cost of processing because cost of processing in cloud is more than that of in fog computing environment.

E. PROCESSING COST ANALYSIS

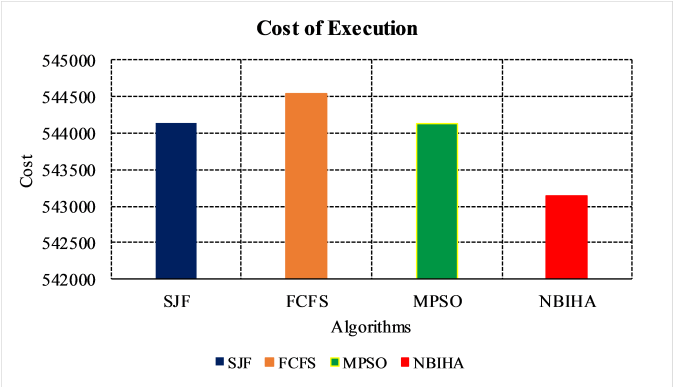

FIGURE 12: Cost analysis of algorithms.

Same is in the case of SJF because it increases time due to mismatch of resource. In case of MPSO, it increases cost due to exact match of the resources to the demand of the tasks which 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 which can also be seen in Fig 12.

VII. CONCLUSION

Fog computing is intended to convey the possibility that the benefits of cloud computing can be provided at the very edge of the network. This paper aims to provide the effective resource utilization in fog-IoT paradigm by proposing a novel hybrid resource management approach named NBIHA. The proposed scheme balances the load among fog nodes and manages the 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 which in result balances the load among fog nodes. The hybrid of bio-inspired algorithms is used to achieve the efficient resource allocation. The simulation results show that the merger of above-mentioned two approaches optimizes the resources utilization and reduces the average response time when compared with the state-of-the-art benchmark and conventional scheduling algorithms. In the future, we intend to utilize the 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. 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 Networks, 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, 2016.

[4] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, "Fog computing for the internet of things: A survey," ACM Transactions on Internet Technology (TOIT), 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 Journal on Wireless Communications and Networking, 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 Communications Surveys & Tutorials, vol. 20, no. 3, pp. 1826–1857, 2018.

[7] D. Willis, A. Dasgupta, and S. Banerjee, "Paradrop: a multi-tenant platform to dynamically install third party services on wireless gateways," in Proceedings of the 9th ACM workshop on Mobility in the evolving internet architecture. ACM, 2014, pp. 43–48.

[8] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in Proceedings of the 2015 workshop on mobile big data. ACM, 2015, pp. 37–42.

[9] M. A. M. C. I. S. U. Bhatti, Fizzah; Shah, "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 Journal on Wireless Communications and Networking, 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 Transactions on Cloud Computing, vol. 6, no. 1, pp. 46–59, 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 Generation Computer Systems, vol. 61, pp. 118–127, 2016.

[13] S. Bitam, S. Zeadally, and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm," Enterprise Information Systems, 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," Knowledge-Based Systems, vol. 115, pp. 123–132, 2017.

[15] B. Ottenwälder, B. Koldehofe, K. Rothermel, and U. Ramachandran, "Migcep: operator migration for mobility driven distributed complex event

processing," in Proceedings of the 7th ACM international conference on Distributed event-based systems. ACM, 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," Journal of network and computer applications, vol. 98, pp. 27–42, 2017.

[17] H. T. T. Binh, D. B. Son, P. A. Duc, B. M. Nguyen et al., "An evolutionary algorithm for solving task scheduling problem in cloud-fog computing environment," in Proceedings of the Ninth International Symposium on Information and Communication Technology. ACM, 2018, pp. 397–404.

[18] R. M. Guddeti, R. Buyya et al., "A hybrid bio-inspired algorithm for scheduling and resource management in cloud environment," IEEE Transactions on Services Computing, 2017.

[19] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang, "Cost efficient resource management in fog computing supported medical cyber-physical system," IEEE Transactions on Emerging Topics in Computing, vol. 5, no. 1, pp. 108–119, 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 of Things Journal, vol. 3, no. 6, pp. 1171–1181, 2016.

[21] J. Oueis, E. C. Strinati, and S. Barbarossa, "The fog balancing: Load distribution for small cell cloud computing," in 2015 IEEE 81st Vehicular Technology Conference (VTC Spring). IEEE, 2015, pp. 1–6.

[22] S. Agarwal, S. Yadav, and A. K. Yadav, "An efficient architecture and algorithm for resource provisioning in fog computing," International Journal of Information Engineering and Electronic Business, 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 Communications, vol. 13, no. 3, pp. 156–164, 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 Computing, vol. 4, no. 2, pp. 26–35, 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 Communications and Mobile Computing, vol. 2019, 2019.

[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 Communications, vol. 16, no. 3, pp. 32–41, 2019.

[29] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," Egyptian informatics journal, 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. Springer, 2014, pp. 329–336.

[31] A. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," Future Generation Computer Systems, vol. 91, pp. 407–415, 2019.

[32] L. Zhang, Q. Fu, J. Chen, H. Bai, and X. Zhou, "A modified particle swarm optimization algorithmâĂŤcpsode," in 2017 29th Chinese Control And Decision Conference (CCDC). IEEE, 2017, pp. 6659–6663.

[33] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in 1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360). IEEE, 1998, pp. 69–73.

[34] B. Santosa and M. K. Ningrum, "Cat swarm optimization for clustering," in 2009 International Conference of Soft Computing and Pattern Recognition. IEEE, 2009, pp. 54–59.

[35] S.-C. Chu, P.-W. Tsai, and J.-S. Pan, "Cat swarm optimization," in Pacific Rim international conference on artificial intelligence. Springer, 2006, pp. 854–858.

[36] S. Svorobej, P. Takako 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. Vahid 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," Software: Practice and Experience, 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," Fog and Edge Computing: Principles and Paradigms, pp. 1–35, 2019.

HINA RAFIQUE received her B.Sc. degree in Software Engineering from COMSATS University Islamabad, Pakistan, in 2013-2017. Currently, she is pursuing her Masters in Software Engineering from Department of Computer Science, COMSATS University Islamabad, Pakistan. She is working at Virtual University of Pakistan as Instructor since 2018. Her Research interests lie in the domain of Cloud Computing, Fog Computing and Software Process Improvement.

MUNAM ALI SHAH received B.Sc. and M.Sc. degrees, both in Computer Science from University of Peshawar, Pakistan, in 2001 and 2003 respectively. He completed his M.S. degree in Security Technologies and Applications from University of Surrey, UK, in 2010, and has passed his Ph.D. from University of Bedfordshire, UK in 2013. Since July 2004, he has been a Lecturer, Department of Computer Science, COMSATS Institute of Information Technology, Islamabad, Pakistan. His research interests include MAC protocol design, QoS and security issues in wireless communication systems. Dr. Shah received the Best Paper Award of the International Conference on Automation and Computing in 2012. Dr. Shah is the author of more than 50 research articles published in international conferences and journals.

SAIF UL ISLAM received his Ph.D. in Computer Science at the University Toulouse III Paul Sabatier, France in 2015. He is Assistant Professor at the Department of Computer Science, Dr. A. Q. Khan Institute of Computer Science and Information Technology, Rawalpindi, Pakistan. Previously, he served as Assistant Professor for three years at the COMSATS University, Islamabad, Pakistan. He has been 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 Network (CDN)) and the Internet of Things (IoT).

TAHIR MAQSOOD received M.Sc. degree in Computer Networks from Northumbria University, UK, in 2007 and Ph.D. in Computer Science from COMSATS Institute of Information Technology, Pakistan in 2017. Currently, he is Assistant Professor at 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 is a Faculty Member with the School of Information Technology, Monash University Malaysia Campus. He has published more than 45 high-impact research articles in reputed international journals, including the IEEE Communications Surveys and Tutorials, ACM Computing Surveys, 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.

CASTREN MAPLE is Professor of Cyber Systems Engineering at WMG's Cyber Security Centre (CSC), University of Warwick. He is the director of research in Cyber Security working with organisations in key sectors such as manufacturing, healthcare, financial services and the broader public sector to address the challenges presented by today's global cyber environment. He is a member of several professional societies including the Council of Professors and Heads of Computing (CPHC) whose remit is to promote public education in Computing and its applications and to provide a forum for those responsible for management and research in university computing departments. He is an elected member to the Committee of this body. He is an Education Advisor for TIGA âĂŞ the trade association representing the UK's games industry. He is also a Fellow of the British Computer Society, the Chartered Institute for IT and is a Chartered IT professional. He also holds two Professorships in China, including a position at one of the top two control engineering departments in China. His interests include Information Security and Trust and Authentication in Distributed Systems.