IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 7, 2020, accepted October 10, 2020, date of publication October 15, 2020, date of current version October 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3031472

Scheduling of Scientific Workflows in Multi-Fog Environments Using Markov Models and a Hybrid Salp Swarm Algorithm

OMED HASSAN AHMED[1], (Member, IEEE), JOAN LU[1], ARAM MAHMOOD AHMED[2,3], (Member, IEEE), AMIR MASOUD RAHMANI[4], MEHDI HOSSEINZADEH[5,6], AND MOHAMMAD MASDARI[7]

[1]School of Computing and Engineering, University of Huddersfield, Huddersfield HD1 3DH, U.K.
[2]Department of Information Technology, Sulaimani Polytechnic University, Sulaymaniyah 46001, Iraq
[3]International Academic Office, Kurdistan Institution for Strategic Studies and Scientific Research, Sulaymaniyah 46002, Iraq
[4]Department of Computer Science, Khazar University, Baku AZ1096, Azerbaijan
[5]Institute of Research and Development, Duy Tan University, Da Nang 550000, Vietnam
[6]Mental Health Research Center, Psychosocial Health Research Institute, Iran University of Medical Sciences, Tehran 14496-14535, Iran
[7]Department of Computer Engineering, Urmia Branch, Islamic Azad University, Urmia 57169-63896, Iran

Corresponding author: Mehdi Hosseinzadeh (hosseinzadeh.m@iums.ac.ir)

ABSTRACT Security attacks are a nightmare to many computing environments such as fog computing, and these attacks should be addressed. Fog computing environments are vulnerable to various kinds of DDoS attacks, which can keep fog resources busy. Typically in such attacks, fog environments often have less available resources, which can negatively impact the scheduling of Internet of Things (IoT) submitted workflows. However, most of the existing scheduling schemes do not consider DDoS attacks' effect in the scheduling process, increasing the deadline missed workflows and offloaded tasks on the cloud. For dealing with these issues, a hybrid optimization algorithm is proposed, comprising both Particle Swarm Optimization (PSO) and Salp Swarm algorithm (SSA), to solve the workflow scheduling problem in multiple fog computing environments. Two discrete-time Markov chain models are proposed for each fog computing environment to address DDoS attacks' effects on them. Our first Markov model computes the average available network bandwidth for each fog. The second Markov model finds the average number of available virtual machines (VMs) for each fog; the models address different levels of DDoS attacks. Extensive simulations show that by predicting the effects of DDoS attacks on fog environments, the proposed approach can effectively mitigate the number of offloaded tasks on cloud data centers and can reduce the number of the deadline missed workflows.

INDEX TERMS Fog computing, task, workflow, optimization, makespan, energy.

I. INTRODUCTION

The proliferation of wireless communication technologies and mobile computing has contributed to IoT [1], [2]. However, IoT devices are resource-constrained. When an application needs more resources than the device's capability for further processing of their data, the tasks need to be offloaded to the resource-rich cloud computing data centers [3]. However, offloading to remote cloud data centers is not always possible in delay-sensitive real-time applications and when constraints such as response time and delay are essential [4]. To deal with these problems, ideas such as fog computing [5], [6] and edge computing [7] provide a virtualized layer between IoT devices and cloud data centers [8]–[10].

Like other computing environments, fog computing is vulnerable to various security attacks, primarily to Distributed Denial of Service (DDoS) attacks [11], [12], which try to prevent fogs from providing their services to the IoT. In [13], various DDoS attacks, which were designed and launched against different cloud computing environments, were investigated and classified. Although many security solutions such as intrusion detection systems [14], [15], trust management systems [16], [17], and cryptographic methods such as blockchain [18] are provided to deal with the security

The associate editor coordinating the review of this manuscript and approving it for publication was Navanietha Krishnaraj Krishnaraj Rathinam.

attacks, the computer systems are still vulnerable to a variety of attacks.

Typically, DDoS attacks try to disguise themselves by mimicking themselves as some legal network traffic, flash crowds, or some legal events in the victim's environment. They may also perform obfuscation, for example, by encrypting their transmitted messages. Generally, DDoS attacks benefit from some vulnerabilities of some protocols and systems to launch their attacks. In this context, reflection-based DDoS attacks and amplification DDoS attacks are special cases of the flooding attacks, putting heavy traffic on the victim's site and consuming its bandwidth. Thus, legal requests cannot get a chance to reach the victim's system, and the victim will be unavailable. Besides, many DDoS attacks such as VM migration attacks, cloud-internal DoS attacks, VM sprawling attacks, neighbor attacks, and VM escape attacks are conducted on virtualization systems. Such attacks reduce the number of available VMs and mitigate fog's ability to handle the IoT network's incoming requests.

Considering the limited fog computing resources, effective resource management in fogs under DDoS attacks is highly essential. In this context, task and workflow scheduling methods can be used to manage fog environments by properly allocating the tasks to the best available VMs [19]. Efficient task and workflow scheduling provide several advantages in fog computing. For instance, scheduling methods can reduce VMs and fogs' power consumption, which leads to more green computing solutions. Besides, reducing the amount of allocated VMs, a cost reduction, and an increase in the throughput and scalability can be achieved. Besides, proper scheduling of tasks can reduce offloaded tasks on the cloud and enable cloud data centers to focus on processing other tasks. Generally, task and workflow scheduling are already proven to be NP-hard problems in fog and cloud computing environments [20]. Several scheduling frameworks, such as [21], have been investigated and introduced to schedule IoT tasks on fog computing's virtual resources. However, only a few scheduling schemes are proposed for fog computing environments that have considered security issues and especially DDoS attacks. Thus, security is a critical issue that has been neglected by fog scheduling schemes, and most of them have considered factors such as performance, power consumption, cost, makespan, and reliability.

Typically, the scheduling algorithms can be classified based on their type of algorithm, namely heuristic and meta-heuristic scheduling algorithms [22]. Specifically, metaheuristic algorithms are successfully applied to solve scheduling problems in various computing environments and fog computing. In this context, the SSA or Salp Swarm Algorithm [23] is an optimization algorithm designed to mimic the salps' behaviors. But, it suffers from problems such as local optima and population diversity. For mitigating these problems, this study combines the SSA with the PSO [24], and the combination of two algorithms results in an algorithm with fast convergence speed. The proposed hybrid optimization algorithm is denoted as the SSPSO algorithm. It provides two sub-populations from its main population and gives one subpopulation to SSA and PSO, respectively. Besides increasing exploitation, each algorithm exchanges its best solution, a random solution, or a probabilistically selected solution. This algorithm can balance the exploration and exploitation to avoid local optima problem, and in this study, it is used to optimize workflow scheduling in multi-fog environments.

In this scheme, the effects of DDoS attacks on the bandwidth of fogs and cloud computing data centers are modeled using a discrete-time Markov chain. Then, the model is used in computing the average available bandwidth of fogs for accurately estimating the makespan of the workflow. Also, another discrete-time Markov model is presented for calculating the average number of the available VMs for each fog environment which suffers from DDoS attacks. This model is used in the population initialization of the SSPSO, with consideration of the appropriate number of VMs as the maximum available VMs. Using the proposed Markov models, the makespan of the workflow will be estimated more accurately, and tasks will be allocated to the fogs regarding their security situation. As a result, fewer tasks will be offloaded from the fog environments to the cloud data centers because of the lack of the required resource to run the IoT tasks. The experiments conducted in the iFogSim simulator [25] indicate the effectiveness of the SSPSO algorithm and the proposed Markov models in terms of the number of offloaded tasks on cloud data centers and the number of the deadline missed scientific workflows.

This article's remainder is structured as follows: Section 2 studies the existing scheduling frameworks; Section 3 introduces the applied optimization algorithm. Section 4 presents the proposed hybrid optimization algorithm; Section 5 introduces the designed scheduling approach based on the proposed hybrid optimization algorithm; Section 6 presents the experiments carried out on scientific workflows. Finally, section 7 provides the concluding issues and future studies directions.

II. RELATED WORKS

Many task scheduling schemes, such as [16], [26]–[32], have been proposed for fog computing; these schemes are presented briefly in this section.

In [33], the authors developed ROSA, an uncertainty-based online scheduling method, to schedule multiple workflows with deadline constraints. This scheduling approach can control the number of tasks waiting on each VM. The authors provided five sets of experiments to compare their scheme with five other algorithms. The comparison results reveal that ROSA performs better than the five compared algorithms concerning costs, deviation, resource utilization, and fairness.

In [34], the authors provided a scheduling algorithm for minimizing the power consumption of IoT workflows on fog's heterogeneous resources. They modeled the problem using integer linear programming to mitigate power consumption. They also introduced EMS, a scheduling algorithm to combine different policies and find near-optimum

scheduling. They analyzed their ILP model and scheduling algorithm and showed that EMS can achieve near-optimal power consumption and can reduce the makespan.

The scheduling method provided in [35] introduces a cross-layer analytical method to balance the service delay and power consumption during dynamic task scheduling in fog networks. The authors specifically combined their analysis with the Lyapunov optimization method, provided a delay and power-aware task scheduling method, and reported that their scheme improves delay-energy performance in task scheduling. As an advantage, they provided a complete simulation of their system and analyzed metrics such as energy consumption, queue length, delay, delay jitter, service delay. However, they have not considered the security factors in scheduling and offloading processes.

In [36], a task-scheduling model was presented using containers to ensure that the tasks are completed on time and considered the number of concurrent tasks for the fog node. Furthermore, the authors proposed a reallocation mechanism to reduce task delays. The results showed that this task-scheduling algorithm could reduce task delays and improve the concurrency number of fog nodes' tasks.

The scheduling scheme provided in [37] introduces a decentralized and scalable scheduling scheme denoted as DATS, for minimizing service delay in heterogeneous fogs. It uses the processing efficiency-based computing resources competition and a QoE-based task scheduling. Based on the conducted experiments, the authors exhibited that DATS can balance computing resources and communication links usage, aiming for reducing service delay. However, this scheme only focused on delay minimization and neglected security issues.

In [38], the authors demonstrated DOTS's application, which is a delay-aware task scheduling algorithm to optimize delay in offloading. They indicated that DOTS could provide the optimal set of helper nodes, subtask sizes, and the transmission power to minimize the task processing delay by conducting the required experiments. Moreover, compared with the command-mode offloading, the voluntary-mode achieves more balanced offloading and a higher fairness level among the fog nodes. The authors have evaluated metrics such as fairness index, energy consumption, delay, and the number of nodes, but no evaluations regarding security issues have been conducted.

In [39], Folo, which is a scheduling algorithm, was proposed for optimizing the latency and quality of task allocation in vehicular fogs. In this study, the authors formulated the task allocation to fog nodes as a bi-objective problem and considered quality loss and service latency. This scheme uses linear programming-based optimization and binary PSO. The study showed that their scheme could achieve higher performance, mitigate the service latency, and reduce the quality loss compared to random and naive node allocation in fog. As an advantage, this scheme is evaluated using the SUMO tool using real-world taxi traces, but again, no security-related factor was incorporated in this scheme.

The authors in [40] proposed a heuristic for the dynamic scheduling of multiple IoT workflows. It schedules computational tasks in the cloud and the fog's communication-intensive tasks by considering the communication cost from IoT devices to fog. The authors compared the performance of this scheme with a cloud-unaware system under different workloads. They indicated that their method provides a lower deadline miss ratio at a higher monetary cost. However, this scheme is simulated using C++, while better simulator environments are available, and also only the time factor is considered in the scheduling process.

In [41], the authors introduced an efficient scheduling approach by using the Max-Min ant system to handle the workflow scheduling problem in multiprocessor environments, used as mini-servers in fog computing. This approach manipulates the priority of tasks to achieve the most optimal task-order. The authors used various random workflows to evaluate the performance of their scheme. However, this scheme is evaluated using the visual basic 6.0 programming environment, and further evaluations on the special-purpose fog simulators are necessary for it.

A fully decentralized hybrid of edge and fog commuting denoted as Edge-Fog cloud was presented in [42]. In this study, the authors provided the LPCF algorithm, which assigns tasks to the Edge-Fog cloud's available nodes. They showed that LPCF achieves near-optimal networking costs in polynomial time as opposed to exponential time complexity. This scheme considers networking costs and processing time, but it does not consider security-related factors.

A deadline-aware task scheduling method was presented in [43], in which fog nodes collaborate and use cloud resources to execute the tasks. The authors formulated the task scheduling problem as a multiple dimensional 0-1 knapsack and proposed an ACO-based algorithm to solve it. Also, they tried to increase the profit of fog computing while meeting the tasks' deadlines. However, this scheme only considers cost and deadline factors and do not consider security issues.

Another task scheduling algorithm was proposed in [44], which prioritizes tasks based on their delay tolerance levels, to increase throughput while reducing the response time and cost. This scheme assigns each incoming request on the nearest fog resources, and in the proper priority queue. Then, it processes all the requests in the three priority queues in a fog server and reassigns some requests to other fog servers when there are insufficient resources to handle the request in its deadline. In the end, the request should be sent to cloud data centers. As an advantage, this scheme is evaluated using Cloudanalyst, which is a CloudSim-based simulation tool. Table 1 provides a comparison of fog scheduling schemes.

III. BACKGROUND KNOWLEDGE

This section mainly focuses on SSA and PSO optimization algorithms that are used in this article.

TABLE 1. Comparison of fog scheduling schemes.

TABLE 1. Comparison of fog scheduling schemes				
Scheme	Task/Workflow	Scheduling Factors	Simulator/ Environment	Evaluation Factors
[34]	Task Scheduling	Energy		Energy Consumption
[35]	Task Scheduling	Energy Consumption, Service Delay, Delay Jitter		Accumulated Energy Consumption, Accumulated Queue Length, Delay, Delay Jitter, Service Delay, Average Energy Consumption
[36]	Task Scheduling	Resources	Docker, QEMU, Libvirt, Linpack	Total Number of Accepted Tasks, Average Reducing Time
[37]	Task Scheduling	Delay		Latency Reduction Ratio, Latency
[38]	Task Scheduling	Delay		Fairness Index, Energy Consumption, Delay, Number of Nodes
[39]	Task Scheduling	Service Latency, Quality Loss	SUMO	Latency
[40]	Workflow Scheduling	Deadline	C++	Deadline Miss Ratio, Monetary Cost, Percentage of Tasks Executed on Cloud, Deadline Miss Ratio
[41]	Workflow Scheduling	Makespan	Visual Basic 6.0	Normalized Schedule Length
[42]		Cost, Network Cost	Python	Network Cost, Processing Cost
[43]	Task Scheduling	Profit		Guarantee Ratio, Profit
[44]	Task and Workflow Scheduling	Estimated Service Time, Delay, Request Arrival Rate	Cloudanalyst	Cost, Response Time

```
Initialize the initial salps
For(round=1; round < max iterations; round++)
begin
    determine the fitness value of each salp
     determine the leader
    Update the position of the leader salp
    Update the follower position
end
return the leader
```

FIGURE 1. Pseudo-code of the SSA algorithm.

A. SSA ALGORITHM

This algorithm is inspired by salps' swarming behavior in the deep oceans to form a chain known as the salp chain. The SSA algorithm organizes its population into followers and leaders in which the leader represents the chain front. In contrast, the followers are other salps in the chain, and they try to follow each other. Accordingly, Figure 1 indicates the pseudo-code of the SSA algorithm.

B. PSO ALGORITHM

PSO is a stochastic and population-based optimization algorithm introduced by Eberhart and Kennedy. This algorithm is inspired by fish schooling or bird flocking, and at first, it is initialized by several randomly generated solutions for a specific problem, also called particles. Then, it searches the problem space for optima by updating the particles using the current best particle, the best position of each particle, and the current position of the particle. The best solution or particle has the lowest fitness value, which has been achieved so far.

IV. THE PROPOSED HYBRID OPTIMIZATION ALGORITHM

As outlined before, SSA and PSO algorithms use different methods for searching for the problem space and have different capabilities. To take advantage of these algorithms, they are combined to have a more compelling hybrid optimization algorithm. Figure 2 depicts the flowchart of the proposed hybrid optimization algorithm. As shown in this flowchart, at first, a random solution is created. Then, the population is divided between these two algorithms. However, each population's size is not fixed and can vary based on the improvements achieved by each of them.

The population combination and division are made once in every seven rounds, which can reduce the overheads of population division. Then, each algorithm works on its subpopulation and searches the problem space using its solutions. After each algorithm is executed for one round, the PSO algorithm sends its best solution to the SSA. If this solution is better than SSA's best solution, it becomes the leader solution; otherwise, it will be dropped. Also, SSA can send the best solution, a randomly selected solution, or a chosen solution using the Rolette wheel to the PSO algorithm. This exchange of solutions helps the applied algorithms to benefit from the results achieved by each other and improves the final result. Also, to provide different populations for SSA and PSO, a population dividing procedure is proposed, as shown in Figure 3. In this regard, two low overhead dividing methods are presented, whereby in the first one, population solutions are divided randomly. In the second one, the solutions are divided using the roulette wheel method. This procedure, at first, assigns an equal number of solutions for SSA and PSO algorithms. Then, in other rounds, some statistics about the algorithm's results are collected and used in dividing

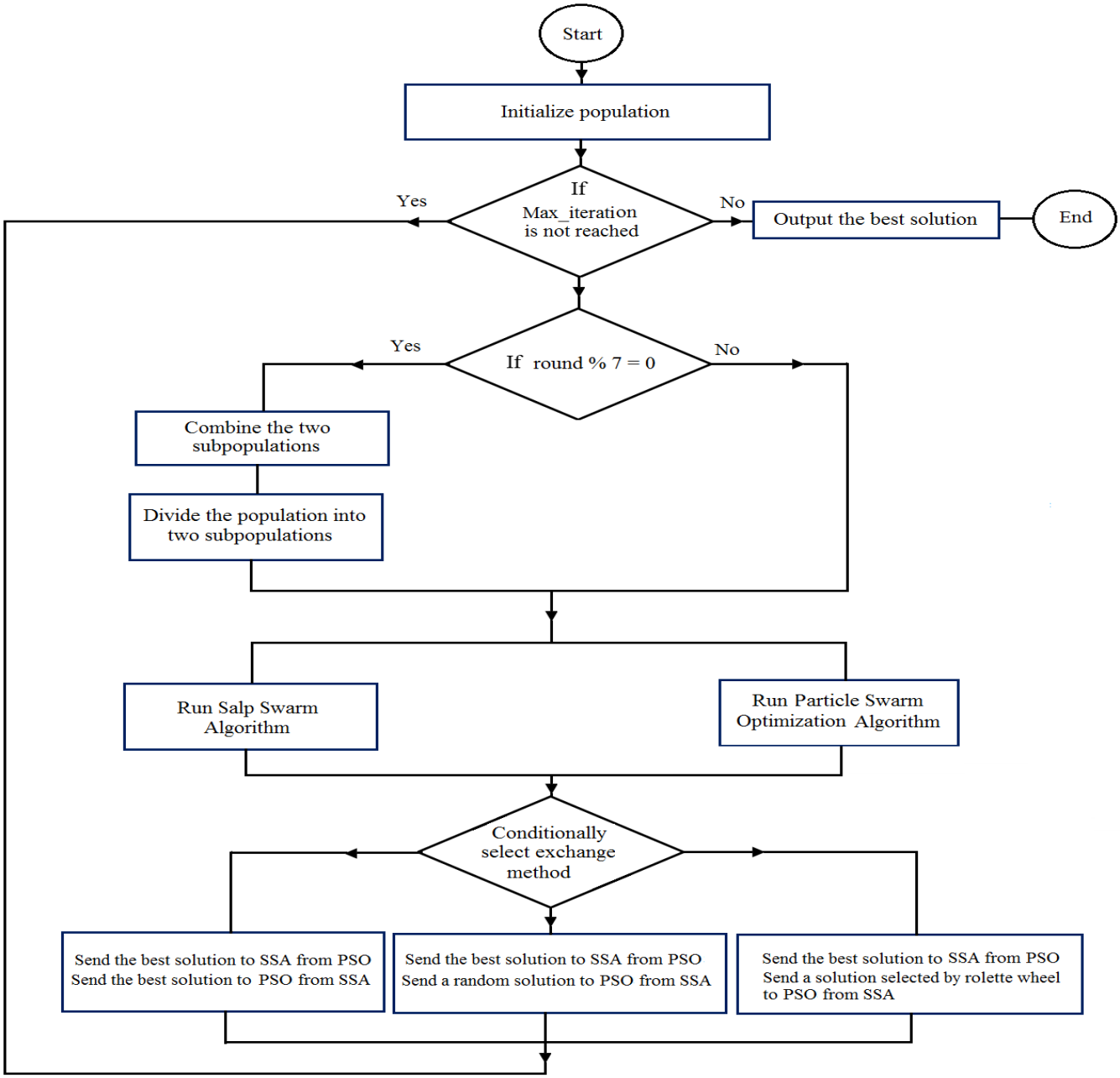


FIGURE 2. Flowchart of the SSPSO hybrid optimization algorithm.

the population according to each algorithm's achievements. Thus, the algorithm which can achieve better results gets more solutions.

V. THE PROPOSED WORKFLOW SCHEDULING APPROACH

This section presents this study's proposed workflow scheduling algorithm designed for multi-fog environments. Accordingly, Figure 4 depicts the architecture of the proposed scheme. On the left side of this figure, there are IoT networks that consist of various home appliances and user devices. Each of these IoT networks is aided by a broker node that possesses fog computing environments' resources information. In this scheme, the presumption is that several fogs have a data repository and a different set of virtual resources. The IoT broker is responsible for scheduling the IoT submitted workflows on the exiting fog environments. Besides, each fog is responsible for executing the tasks delivered to it, and when it does not have the required resources to run the tasks, it submits them to cloud computing. The scheduling scheme proposed in this study is mainly trying to achieve the following objectives:

```
Function Dividing_population()
  Input: Population
         Divisition_type
  Output: Two sub-population
Begin
    If  round = 0  or  round mod 49 = 0   Then
        SSA_population_size= population_size/2
        PSO_population_size= population_size/2
    Else
        If  SSA_improvement > PSO_improvement  Then
            Increase SSA_population_size
            Decrease PSO_population_size
        Else
            Decrease SSA_population_size
            Increase PSO_population_size
        End
    End
    If  Division_type =1  Then    //  Divide the population randomly
            For  i=1 to  PSO_population_size
                 Select a random solution from the main population
                 Remove the solution from the main population
                 Add the solution to PSO's sub-population
            End
             Add the rest of the population to the SSA's sub-population
     Else    //  Divide the population using the roulette wheel
          For  i=1 to  PSO_population_size
              Select a solution from the main population using the roulette wheel, based on the fitness values of the solutions
              Remove the solution from the main population
              Add the solution to PSO's sub-population
           End
           Add the rest of the population to the SSA's sub-population
     End
End
```

FIGURE 3. Dividing the population between SSA and PSO algorithms.

- To optimize workflow scheduling in a multi-fog environment.
- To reduce the number of tasks that should be offloaded on cloud computing data centers by correctly estimating the workflow makespan.
- To consider the impact of DDoS attacks on the network bandwidth of fog and cloud environments.
- To consider the impact of DDoS attacks on the number of available VMs in fog computing.

In this scheme, the proposed SSPSO optimization algorithm is applied to optimize the workflow scheduling in a multi-fog environment for achieving these objectives.

A. FORMULATING THE WORKFLOW SCHEDULING

This section provides a formal definition of the considered workflow model. Table 2 specifies the abbreviations applied in the rest of this section. In this scheme, the assumption is that there are several fog computing environments denoted as, $Fog = \{fog_1, fog_2, fog_3, \dots\}$ and each of these fogs contains a set of virtual machines or VMs, denoted by $VM_i = \{VM_{i1}, VM_{i2}, VM_{i3}, \dots\}$, in which VM_i denotes the VMs of fog_i. Several brokers are used in this scheme, and brokers are denoted as, $Broker = \{broker_1, broker_2, broker_3, \dots\}$. Also, cloud computing environments considered in the scheme are denoted as, $Cloud = \{cloud_1, cloud_2, cloud_3, \dots\}$. Besides, each workflow contains some tasks expressed as, $W_i = \{T_1, T_2, T_3, \dots\}$. Moreover, each workflow is considered as a DAG, in which each node represents a task, and the edges

TABLE 2. Abbreviations and Acronyms.

Abbreviation	Description
VM_i	i^{th} VM
Nvm	Number of VMs
W_i	i^{th} workflow
T_i	i^{th} task
$FT(T_i, VMj)$	Finish time of i^{th} task on the j^{th} VM
$avail(VM_j)$	The time that j^{th} VM will be available to execute the tasks
$Exection\_Time(T_i, VM_j)$	The execution time of the task T_i on the VM_j
$STime(T_i, VM_j)$	The start time of the task T_i on the VMj
$Ave(Exection\_Time(Ti))$	The average execution time of the task T_i
$Com\_Time(T_i, Tj)$	Communication time of the data transfer between the T_i and T_j
$Predecessor(T_i)$	Predecessor set of task Ti
$Successor(T_i)$	Successor set of task Ti

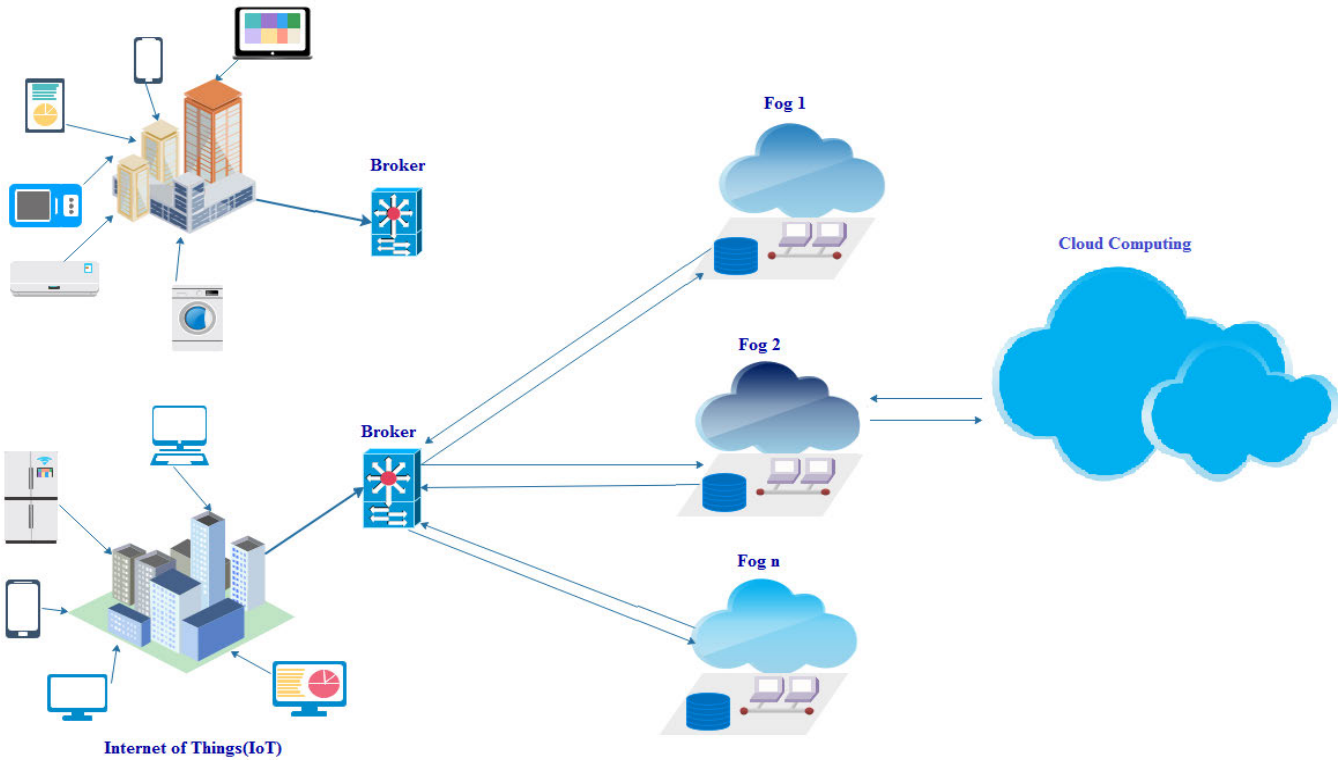


FIGURE 4. Fog computing architecture.

specify the data or control dependencies between the tasks, in which E_{ij} defines the edge between the T_i and T_j, when $T_i \neq T_j$. This indicates that child tasks can be executed after all its parent tasks have been fully executed, and their output data have been sent to it. Meanwhile, control dependencies only transfer the configuration parameters needed to execute the child task and transfer fewer data. However, the transferred data in the data dependencies are used as input data to the child process.

B. RANKING WORKFLOW TASKS

This scheme benefits from the task prioritization method proposed by HEFT algorithm for finding the ranks of tasks in the scientific workflows execution process. HEFT is a heuristic list scheduling algorithm that attempts to schedule a set of inter-dependent tasks on a set of VMs by taking the communication time into account. Equation 1 indicates how the rank should be computed for each workflow task, as follows:

$$\begin{aligned} Rank(T_i) = {} & Ave\,(Exection\_Time(T_i)) \\ & + \{\max(Com\_Time(T_i, T_j) \\ & + Rank(T_i))|T_j \in \text{Successor}(T_i)\} \end{aligned} \tag{1}$$

where T_i is the i^{th} task in the workflow and $Ave(Exection\_Time(T_i))$ is the average execution cost of the i^{th} task. Also, as outlined before, Successor(T_i) specifies the successor tasks of the T_i and $Com\_Time(T_i, T_j)$ specifies the communication cost between the T_i and T_j. After the priority of tasks is determined, the tasks should be allocated to the VMs. In this process, the highest priority task in which all its parent tasks are executed should be scheduled on the VM that leads to the earliest finish time. In this scheme, the set of all direct predecessors of each workflow task can be computed, as follows:

$$Predecessor(T_i) = \{T_j|(T_j, T_i) \in E\} \tag{2}$$

Thus, their predecessor set should be empty regarding the entry task or tasks, as follows: $Predecessor(T_{entry}) = \{\}$. Furthermore, the set of all direct successors of each task can be computed, as follows:

$$Successor(T_i) = T_j|(T_i, T_j) \in E \tag{3}$$

Additionally, for the exit task or tasks, their successor set will be empty, $Successor(T_{exit}) = \{\}$.

To compute the average computation time of the T_i on VM_j, Equation 4 should be used:

$$Exection\_Time(T_i, VM_j) = \frac{Task\_len(T_i)}{VM_j} \tag{4}$$

Accordingly, the average execution time of the task T_i on all VMs, can be computed as follows:

$$Ave\,(Exection\_Time(T_i)) = \frac{1}{Nvm}\sum_{j=1}^{Nvm} Exection\_Time(T_i, VM_j), \tag{5}$$

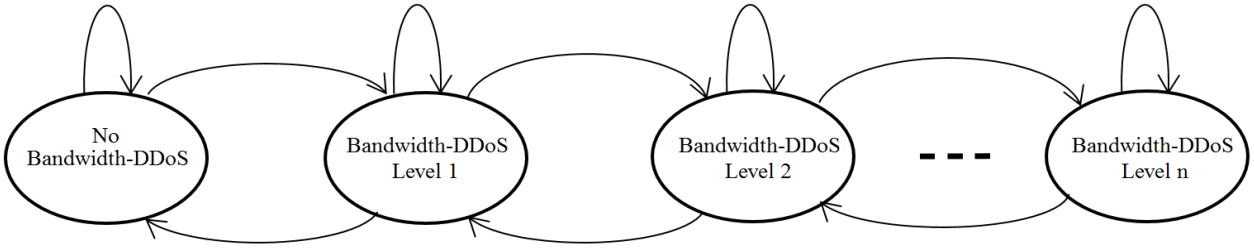


FIGURE 5. Bandwidth Markov chain model.

In this scheme, the earliest start time of each task can be computed by using Equation 6, as shown at the bottom of the page. Here, $avail(VM_j)$ is when jth VM becomes available to execute the requested task.

Also, in this scheme, the finish time of each task can be computed as Equation 7, as shown at the bottom of the page. Here, $deadline_{w_i}$ denotes the deadline of w_i workflow. Furthermore, the communication time of the data transfer between T_i and T_j can be computed as in Equation 8, as shown at the bottom of the page. Here, $Bandwidth(VM\ (T_i)\ , VM\ (T_j))$ is the bandwidth between two VMs which should execute the T_i and T_j tasks, while $Data\ (T_i, T_j)$ denotes the amount of data which should be transferred between these tasks. Typically a table is required to store the maximum available bandwidth among different VMs of the first fog computing environment. Also, a similar data structure for storing the bandwidth of VMs in other fogs is considered.

C. COMPUTING BANDWIDTH REGARDING DDOS ATTACKS

In this scheme, the maximum available bandwidth between different fogs and brokers is stored in another table. In most scheduling schemes, these tables only store the maximum available bandwidth, which in practice, these values are much less due to problems such as DDoS attacks.

However, to be more practical, various factors affecting network bandwidth should be considered in computing the true average bandwidth. It is important to note that not having the average bandwidth calculated may increase the bandwidth's reliance, leading to incorrect selection of proper virtual resources to run the workflow, which results in SLA violations. In this scheduling approach, the effects of DDoS attacks on the network bandwidth are taken into account. A discrete Markov chain model (Figure 5) for computing the average available bandwidth of brokers, fogs, and clouds, is proposed. Using this method, the overestimation of the available bandwidth for brokers, fogs, and clouds, which can lead to a more accurate estimation of the workflow makespan, can be prevented.

Generally, n states are considered for this discrete Markov chain model. The first state indicates the No bandwidth-DDoS attack state, in which the maximum bandwidth is available. Meanwhile, other states denote different levels of DDoS attacks, whereby each of them suffers from some degree of DDoS attacks. As a result, in the state of the DDoS Level 1, the network bandwidth is a little bit reduced, and in the DDoS Level 2, the network bandwidth is reduced further, and this reduction of bandwidth increases in the other states of the Markov model. In this scheme, we compute the average available bandwidth between fogs and brokers by using Equation 9, as shown at the bottom of the next page. Here, $Fog\_ave\_bandwidth_i$ is the average bandwidth of the ith fog and $Broker\_ave\_bandwidth_j$ is the average bandwidth of the jth broker. For computing these average bandwidth, the bandwidth Markov model for them should be solved. By solving the bandwidth Markov model for each fog, we can compute the average available bandwidth for a fog environment by Equations 10 and 11, as shown at the bottom of the next page. In which, $bandwidth_k$ indicates the bandwidth of the kth state and is a portion of $Fog\_bandwidth_i$, which denotes the maximum bandwidth of the ith fog. Also, in this equation, P_k is the probability of the ith state of the proposed Markov model.

$$ESTime(T_i, VM_j) = \begin{cases} 0 & If\ T_1\ is\ an\ entry\ task \\ \max\{avail\ (VM_j)\ , \max\ \{FT\ (T_j) + Com\_Tim\ (T_j, T_i)\} & otherwise\ For\ each\ T_j \in Predecessor\ (T_i) \end{cases} \quad (6)$$

$$FT\ (T_i, VM_j) = \begin{cases} deadline(w_i) & if\ T_i\ is\ an\ exit\ task \\ ESTime(T_i, VM_j) + Ave\ (Exection\_Time(T_i)) & otherwise \end{cases} \quad (7)$$

$$Com\_Time(T_i, T_j) = \begin{cases} 0 & IF\ VM\ (T_i) = VM\ (T_j) \\ \dfrac{Data\ (T_i, T_j)}{Bandwidth(VM\ (T_i)\ , VM\ (T_j))} & otherwise \end{cases} \quad (8)$$

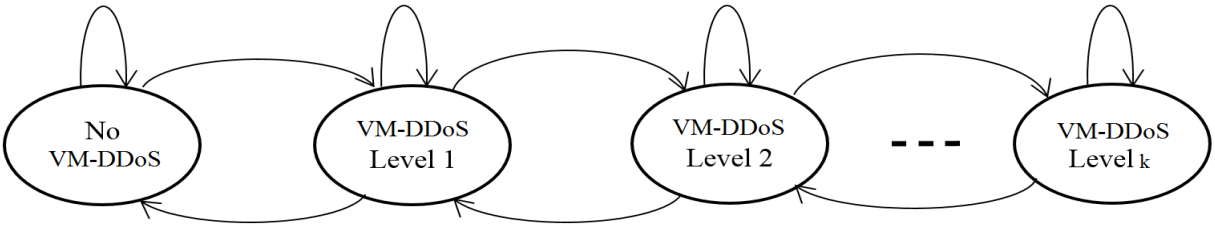


FIGURE 6. VM Markov model for a fog computing environment.

By solving the bandwidth Markov model for each broker, we can compute the average available bandwidth for a broker by Equations 12 and 13, as shown at the bottom of the page. In these equations, $bandwidth_k$ denotes the bandwidth of the kth state and is a portion of $Broker\_bandwidth_i$, which denotes the maximum bandwidth of the ith broker node. In this scheme, when a fog environment does not possess the required resources to run the tasks, it will offload them to cloud computing data centers. To compute the average available bandwidth between clouds and fogs, we use Equations 14–16, as shown at the bottom of the page. Here, $Fog\_ave\_bandwidth_i$ is the average bandwidth of the ith fog and $Cloud\_ave\_bandwidth_j$ is the average bandwidth of the jth cloud environment. Also, $bandwidth_k$ indicates the bandwidth of kth state and P_k is the probability of the ith state in steady state distribution.

D. INITIAL POPULATION AND ENCODING

For encoding the considered scheduling problem, a two-dimensional array is used, in which each row indicates a fog environment, and each column indicates a task of the workflow. Each cell of this table specifies a VM that should execute the workflow in the ith fog environment. One of the main objectives of the scheduling algorithm performed in this study is to minimize the number of VMs allocated to execute each workflow in fog environments. Generally, most of the existing scheduling schemes assume that all VMs are available for fog computing scheduling. However, the assumption is inaccurate, and the availability of VMs can oscillate regarding problems such as DDoS attacks and VM failures.

This scheme considers the effect of such problems by using a discrete Markov chain model (Figure 6), in which each state indicates the attacks faced by fog and the corresponding number of the available VMs. In this Markov model, each state indicates the severity of the attacks. For simplicity, the same level of severity of all DDoS attacks is assumed.

For solving the proposed Markov models, a transition table is used for each of these models. This table indicates the probability of transition from one state to another. Generally, in the transition tables, the sum of each row should be 1. For instance, $\sum_{j=1}^{n} P_{ij} = 1$ indicates that the sum of ith row in matrix P is 1, and in this equation, $0 \leq P_{ij} \leq 1$ and P_{ij} is the transition probability of transition from ith state to jth state. The required data for building the transition table can be achieved from fog computing servers' security logs.

To solve this Markov chain model, we can simply compute the P matrix's powers until it converges, in which all row will

$$Fog\_Broker\_BW_{ij} = min(Fog\_ave\_bandwidth_i, Broker\_ave\_bandwidth_j) \tag{9}$$

$$Fog\_ave\_bandwidth_i = \sum_{k=1}^{Sn} bandwidth_k * P_k \quad \text{where} \quad \sum_{k=1}^{Sn} P_k = 1 \tag{10}$$

$$bandwidth_k = k_i * Fog\_bandwidth_i \tag{11}$$

$$0 < k_i \leq 1 \sum_{i=1} k_i = 1$$

$$Broker\_ave\_bandwidth_j = \sum_{k=1}^{Sn} bandwidth_k * P_k \quad \text{where} \quad \sum_{k=1}^{Sn} P_k = 1 \tag{12}$$

$$bandwidth_k = k_i * Broker\_bandwidth_i \tag{13}$$

$$0 < k_i \leq 1 \sum_{i=1} k_i = 1$$

$$Fog\_Cloud\_BW_{ij} = min(Fog\_ave\_bandwidth_i, Cloud\_ave\_bandwidth_j) \tag{14}$$

$$Cloud\_ave\_bandwidth_i = \sum_{k=1}^{Sn} bandwidth_k * P_k \quad \text{where} \quad \sum_{k=1}^{Sn} P_k = 1 \tag{15}$$

$$bandwidth_k = k_i * Cloud\_bandwidth_i$$

$$0 < k_i \leq 1 \sum_{i=1} k_i = 1 \tag{16}$$

```
For i=1 to  Nfog
    Compute Ave_VMi by solving the Markov model
    for the ith fog
End
For i=1 to population_size
   For j=1 to  Nfog
      For  k=1 to Ntasks
          Pop(i,j,k) = random (0 to Ave_VMi)
      End
   End
End
```

FIGURE 7. Encoding initialization.

be the same. By solving this Markov model, we can achieve the steady-state of the Markov model. Then, we can compute ave_VM_i or the average available VMs in the i^{th} fog by using Equation 17:

$$ave\_VMi = \sum_{j=1}^{Sn} N_{vmj} * P_j \tag{17}$$

In which, N_{vmi} indicates the number of available VMs in i^{th} state and P_i is the probability of i^{th} state, achieved from the steady-state distribution.

To provide effective workflow scheduling, we use the result of solving the Markov model in the encoding of the solutions (Figure 7). For this purpose, first, the VMs Markov model for each fog should be solved, and then the average number of VMs achieved by these Markov models should be used as the upper bound of each dimension of the solutions.

E. FITNESS FUNCTION

Equation 18, as shown at the bottom of the next page, indicates the fitness function applied in this fog scheduling scheme, in which α and β coefficients determine the importance of each objective, and their sum should be 1. Moreover, the makespan of w_i workflow can be computed by Equation 19, as shown at the bottom of the next page. Here, $Nlevel_i$ is the number of levels of the i^{th} workflow submitted to the broker, and $inputdelay_j$ determines the delay of sending the input data of the j^{th} level of the workflow from the broker to the designated fog. Also, $processingdelay_j$ is the processing delay of the j^{th} level's tasks, and $outputdelay_j$ is the delay of sending the output of the j^{th} level's tasks to the broker. Also, $inputdelay_j$ can be computed by Equations 20–24, as shown at the bottom of the next page. Here, $Ntasklevel_j$ is the number of tasks in the j^{th} level of w_i workflow while $inputdelay_{jk}$ is the input delay of the k^{th} task of the jth level. In which, $Ntasklocal_j$ indicates the number of tasks executed locally on the fog and $Ntaskcloud_j$ exhibits the number of tasks offloaded to a remote cloud datacenter, in the j^{th} level of the workflow. Also, $Nvmlevel_j$ is the number of the VMs needed in the j^{th} level, and $Nlevel_i$ is the number of levels in the i^{th} workflow. Also, $outputdelay_{jk}$ is the output of the k^{th} task in the i^{th} level of the w_i workflow.

The pseudo-code of the proposed SSPSO algorithm is shown in Figure 8.

```
Procedure Fog_scheduling()
Input: Workflow
       Deadline
       Fogs and their VMs
Determine the number of VMs in the fog environments
Determine the workflow deadline
Read workflow data from its DAX file
Solve the VMs Markov chain models to get ave_VM for each
fog
Max_VM=floor(ave_VM)
Run HEFT
Compute the workflow tasks' rank based on the HEFT
ranking method
Sort tasks in increasing order of their rank
Use this task order in population initialization
For i=1 to population_size
   For j=1 to  Nfog
      For  k=1 to Ntasks
          Pop(i,j,k)=rand(1 to  Max_VM)
      End
   End
End
Set the fitness function
Use the SSPSO optimization algorithm to obtain the best
solution
For each  Ti task in workflow Wi
      Find the Fogi and VMj from the best solution
      Find the predecessor set of Ti

      While( VMj in the Fogi is not idle and or all Ti's
predecessor are not executed )
          Wait
      End
      Send the task Ti and its required input data to execute Ti
on the VMi
      Send the results of the Ti to its descendants
      Wait until receiving the response from the fogs
       Check the result for their integrity
End
```

FIGURE 8. Workflow scheduling using SSPSO algorithm in broker nodes.

```
Receive the tasks to be executed and their input data
Check the availability of the resources for running the specified
part of the workflow
If the required number of VMs are available  Then
   for each input tasks Ti
      While( the required VMj in the fog is not idle)
          Wait
      End
      Run the tasks
   End
Else
   Send the input data and tasks to the designated cloud
computing data centers
End
Send the output of tasks execution to the broker which has issued
the workflow
```

FIGURE 9. Execution of tasks in each fog.

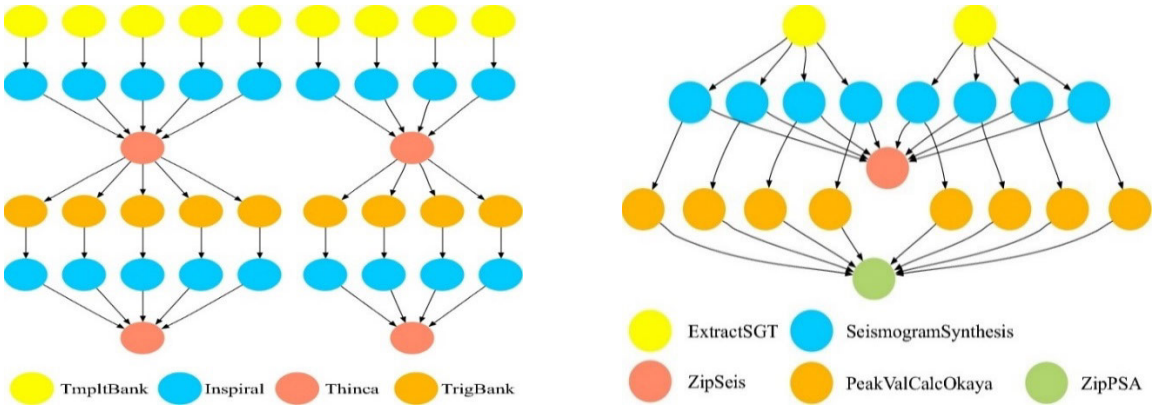


FIGURE 10. Scientific workflows: (a) LIGO, (b) CyberShake.

Figure 9 exhibits the execution of tasks in different fogs. As shown in this figure, when a fog receives the input tasks and their required data, it checks the availability of the required number of VMs, specified by the broker. Then, when the needed resources by the tasks become available, fog should run the tasks; otherwise, when it cannot allocate the required number of VMs, it offloads the tasks and their data to the cloud computing data center. Then, the tasks assigned to fog are executed, and their results should be forwarded to the broker node, which should further process them and send them to other fogs to run other tasks.

VI. EXPERIMENTAL RESULTS

This section presents the results of extensive simulations conducted on the proposed workflow scheduling approach. We simulated our proposed scientific workflow scheduling scheme in the IFogSim simulator and compared it with other workflow scheduling schemes such as MOWO [45] and Hybrid-EDF [40], which are designed for fog computing environments, regarding metrics such as the percentage of the average number of tasks offloaded to cloud data centers and the percentage of the average number of workflows which have missed their deadline.

A. EXPERIMENTS

In this section, two sets of experiments are presented in two different scenarios. The first set of scenarios is defined on LIGO and CyberShake workflows, depicted in Figure 10. In these scenarios, two sets of workflows with 100 and 1000 tasks are considered. Also, in these experiments, four fog

$$Fitness = \begin{cases} 1 & if\ makespan(w_i) > deadline(w_i) \\ \alpha * \dfrac{makespan(w_i)}{deadline(w_i)} + \beta * \dfrac{Nneededvm(w_i)}{NVM} & if\ makespan(w_i) < deadline(w_i) \end{cases}$$

$$\alpha + \beta = 1 \tag{18}$$

$$makespan(w_i) = \sum_{j=1}^{Nlevel_i} (inputdelay_j + processingdelay_j + outputdelay_j) \tag{19}$$

$$inputdelay_j = \begin{cases} 0 & if\ j = 0 \\ \sum_{k=1}^{Ntasklevel_j} inputdelay_{jk} & if\ j \neq 0 \end{cases} \tag{20}$$

$$processingdelay_j = \sum_{k=1}^{Ntasklocal_j} processingdelay_{jk} + \sum_{j=1}^{Ntaskcloud_i} (cinputdelay_j + cprocessingdelay_j + coutputdelay_j) \tag{21}$$

$$Ntasklevel_j = Ntasklocal_j + Ntaskcloud_j \tag{22}$$

$$outputdelay_j = \begin{cases} 0 & if\ j = 0 \\ \sum_{k=1}^{Ntasklevel_j} outputdelay_{jk} & if\ j \neq 0 \end{cases} \tag{23}$$

$$Nneededvm(w_i) = \sum_{j=1}^{Nlevel_i} Nvmlevel_j \tag{24}$$

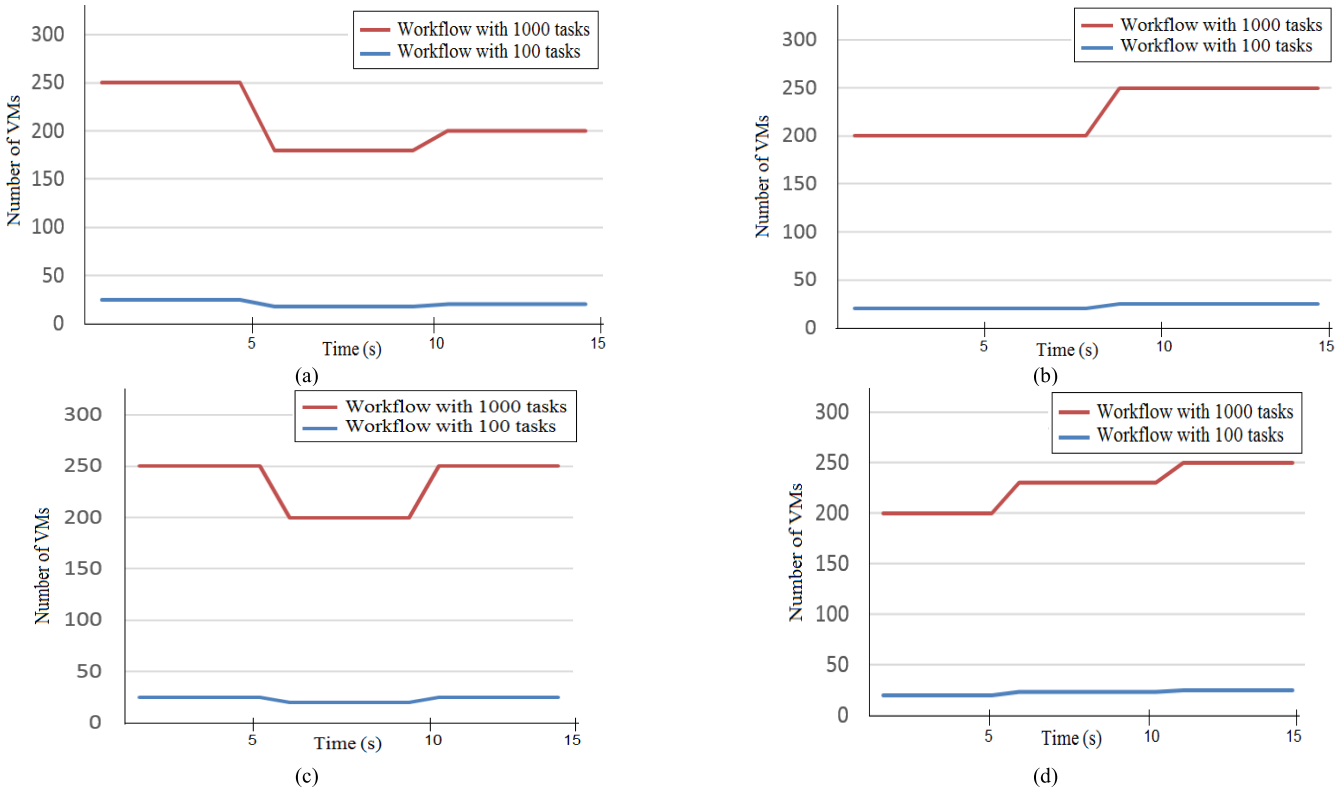


FIGURE 11. The impact of the DDOS attacks on the fogs' VMs.

computing environments are considered in two scenarios. In the first scenario, 300 VMs are considered for executing workflows with 1000 tasks, and in the second scenario, 30 VMs are used for running workflows with 100 tasks.

In this study, the effects of DDoS attacks on each fog have been considered, and the number of VMs available to run the workflows in 15 seconds simulation time period was depicted. As shown in Figure 11, the impacts of DDoS attacks are dynamic and temporal, and these attacks cause an oscillation in the number of available VMs. Consequently, when fogs are under security attacks, they may have fewer VMs than their maximum number. Typically, this issue can negatively impact the execution of the workflows, whereby some tasks may be offloaded to cloud computing environments, while others may miss their deadlines. Figures 12 indicates the percentage of the offloaded tasks on the cloud in four considered fogs, in which the results of the proposed scheme are compared against those of MOWO and Hybrid-EDF workflow scheduling schemes. In this scenario, LIGO workflows with 100 and 1000 tasks are executed. As shown in this figure, the proposed method offloads fewer tasks on the cloud and is more effective than the other two workflow scheduling schemes. This improvement results from the applied Markov models used for considering the impacts of DDoS attacks on VMs of fog environments. As outlined in the previous section, this study computes the average number of VMs for each fog, and therefore, the scheme proposed in this study suffers less from DDoS attacks. Also, the availability of fewer VMs in fogs makes the makespan much longer, and finally, the deadline of workflows may be missed.

Figure 13 exhibits how the proposed scheme computes the percentage of the average number of deadline missed workflows. Meanwhile, Figure 14 indicates the percentage of the average number of the deadline missed workflows in the experiments conducted with LIGO workflows on the four considered fog environments. As shown, the scheme proposed in this study computes the average number of available VMs. As such, the scheme can better tolerate the negative impact of DDoS attacks, and as a result, fewer workflows missed their deadlines. However, since other workflow scheduling schemes rely on the maximum number of the available VMs, they cannot tolerate the impact of DDoS attacks on fog's VMs. Finally, some of their workflows missed their deadline. In addition to the LIGO workflow, CyberShake workflow was analyzed with the settings specified to evaluate LIGO workflows' scheduling. For this purpose, the impact of DDoS attacks on fogs was considered as well.

Table 3 determines the percentage of the average number of offloaded tasks in cloud data centers. As shown in this figure, the scheme proposed in this study provides better results than other scheduling approaches and mitigates the number of tasks that should be offloaded to the cloud. By comparing the results of the scheduling of CyberShake workflows with

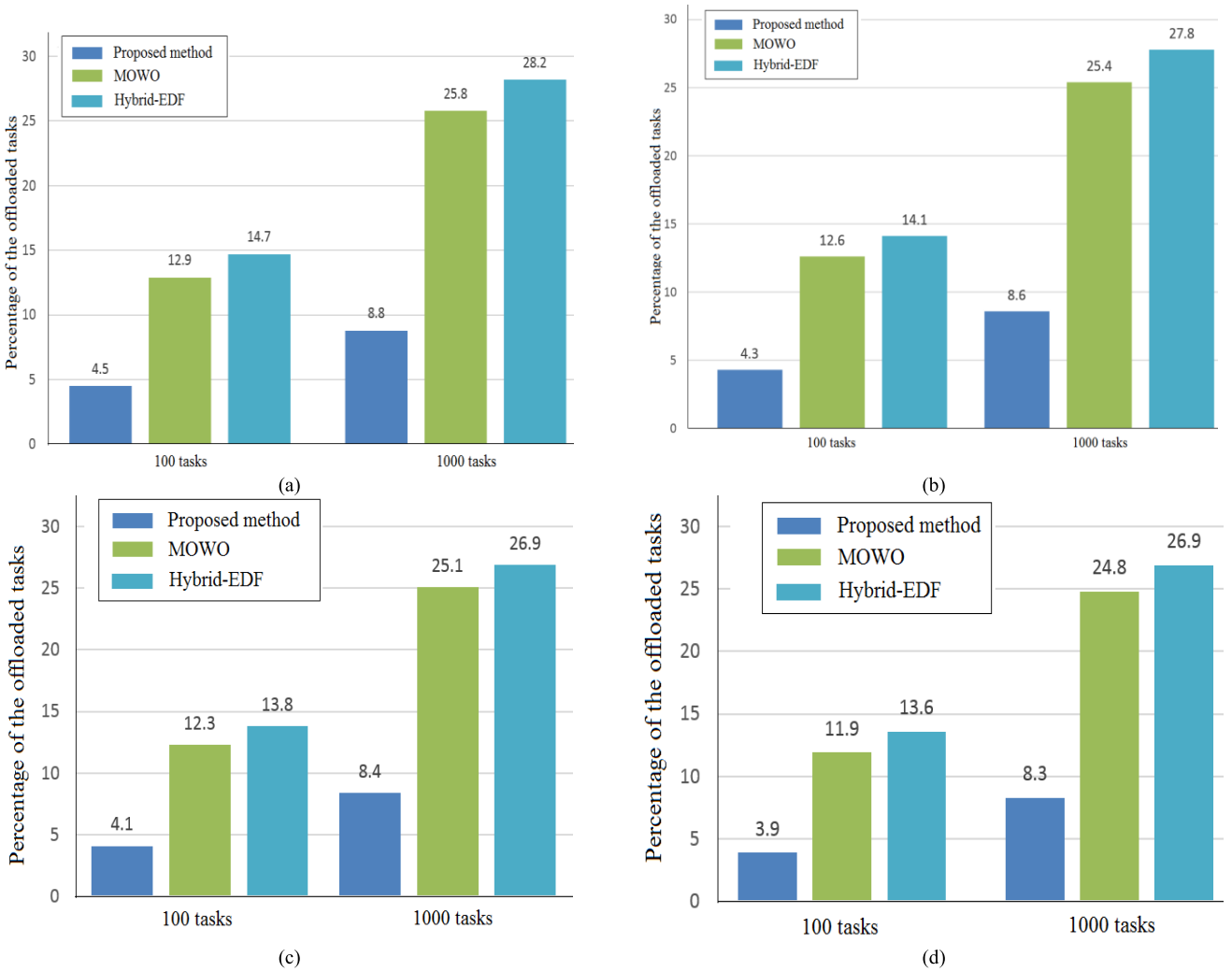


FIGURE 12. The impact of the DDOS attacks on the fogs' VMs in the LIGO workflows.

```
Deadline_missed=0;
Number_of_workflows=70;
For i=1 to Number_of_workflows
    Call Fog_scheduling(wi)
    If makespan(Wi)> Deadline(Wi) Then
        Deadline_missed++;
    End
End
Print 100* Deadline_missed/ Number_of_workflows;
```

FIGURE 13. Computing the percentage of deadline missed workflows.

LIGO workflows, it can be seen that since CyberShake workflows need fewer VMs in each level, the workflows suffered less from DDoS attacks.

Table 4 indicates the percentage of the average number of the deadline missed workflows. As shown in this table, fewer workflows have missed their deadline by using the proposed workflows scheduling approach.

In the second set of experiments, the required evaluations on Epigenomics and Montage scientific workflows were performed. Accordingly, Figure 15 exhibits the architecture of these workflows.

Also, Figure 16 indicates the availability of VMs for the four considered fog environments. It is worth noting that higher severity of DDoS attacks is considered in this set of experiments, and as a result, fewer VMs are available for scheduling.

In this part of the paper, the impact of DDoS attacks on Epigenomics workflows is evaluated. Table 5 accordingly determines the percentage of the average number of offloaded tasks of Epigenomics workflows on cloud data centers. As shown in this table, the proposed scheduling approach provides better results than other scheduling approaches and reduces the tasks offloaded on cloud computing. Table 6 indicates the percentage of the average number of the deadline missed workflows. As shown in this table, fewer workflows have missed their deadline by using the proposed workflows scheduling approach.

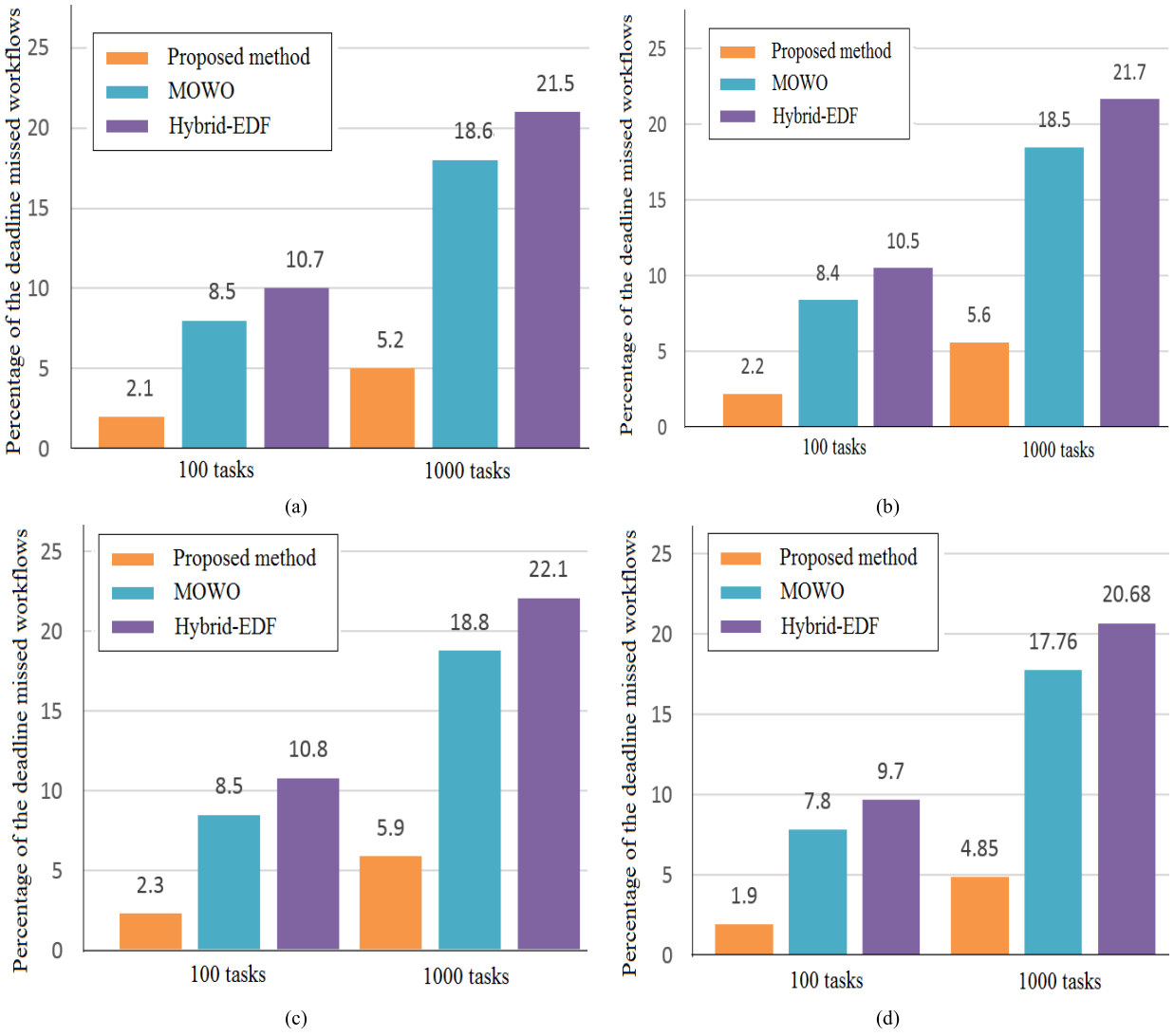


FIGURE 14. The percentage of the average number of the deadline missed LIGO workflows.

TABLE 3. Percentage of the average number of offloaded tasks on cloud computing in CyberShake workflows.

	100 Tasks			1000 Tasks		
Environments	Proposed method	MOWO	Hybrid-EDF	Proposed method	MOWO	Hybrid-EDF
Fog 1	4.1	12.3	13.8	8.3	25.1	27.2
Fog 2	4.3	12.4	14.2	4.5	12.9	14.7
Fog 3	3.4	12.1	13.6	4.1	12.3	13.8
Fog 4	3.6	11.4	13.2	3.7	11.6	13.4

TABLE 4. Percentage of the average number of the deadline missed CyberShake workflows.

	100 Tasks			1000 Tasks		
Environments	Proposed method	MOWO	Hybrid-EDF	Proposed method	MOWO	Hybrid-EDF
Fog 1	2.1	8.3	10.5	2.2	8.4	10.7
Fog 2	2.1	8.2	10.1	5.5	18.2	21.3
Fog 3	2.2	8.2	10.5	5.6	18.4	21.6
Fog 4	1.6	7.5	9.4	4.6	17.4	20.1

Figure 17 shows the percentage of the average number of offloaded Montage workflows tasks with 100 tasks and with 997 tasks on cloud computing. Since montage workflow has not symmetric structure, it needs more VMs in the first two

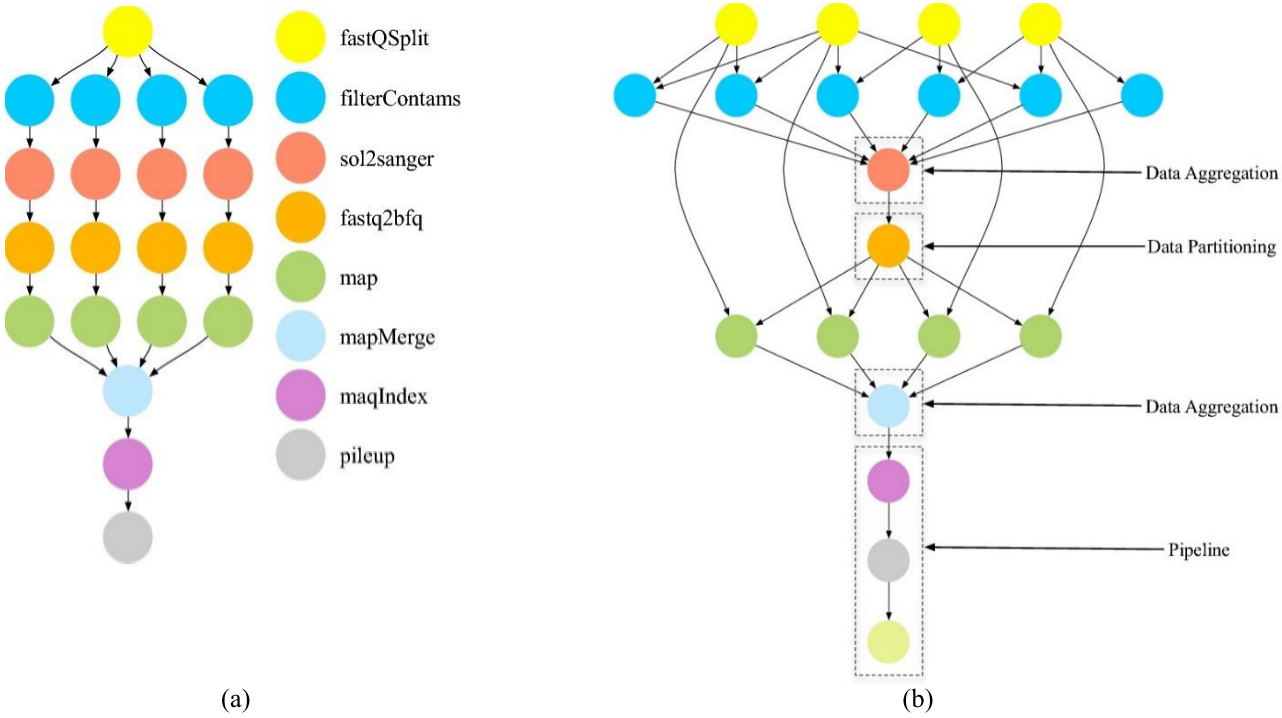


FIGURE 15. Scientific workflows: (a) Epigenomics, (b) Montage.

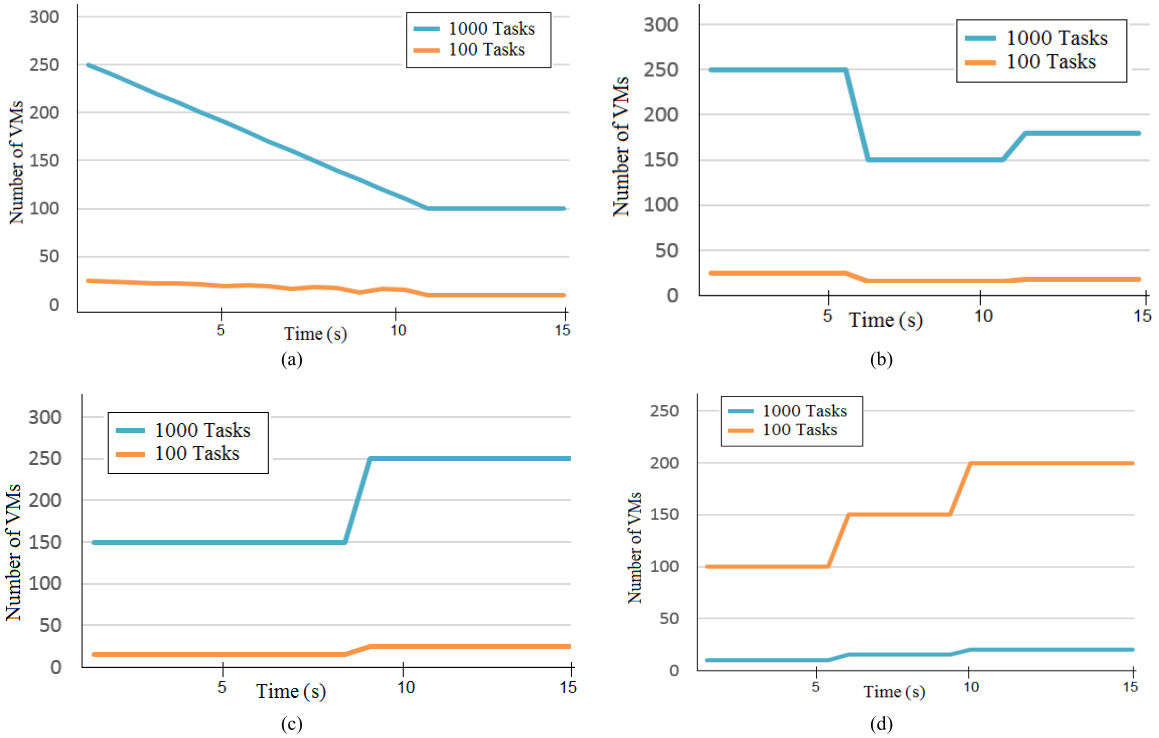


FIGURE 16. The impact of the DDOS attacks on the Fogs' VMs.

levels, causing more tasks to be offloaded on cloud computing. The proposed scheduling scheme uses the proposed SSPSO algorithm to find the minimum number of available VMs to run the workflows while keeping the makespan as low as possible.

Figure 18 exhibits the percentage of the average number of Montage workflows that have missed their deadlines. These results are achieved by running the proposed scheduling scheme, MOWO, and the Hybrid-EDF with VMs; their availability is shown in Figure 16.

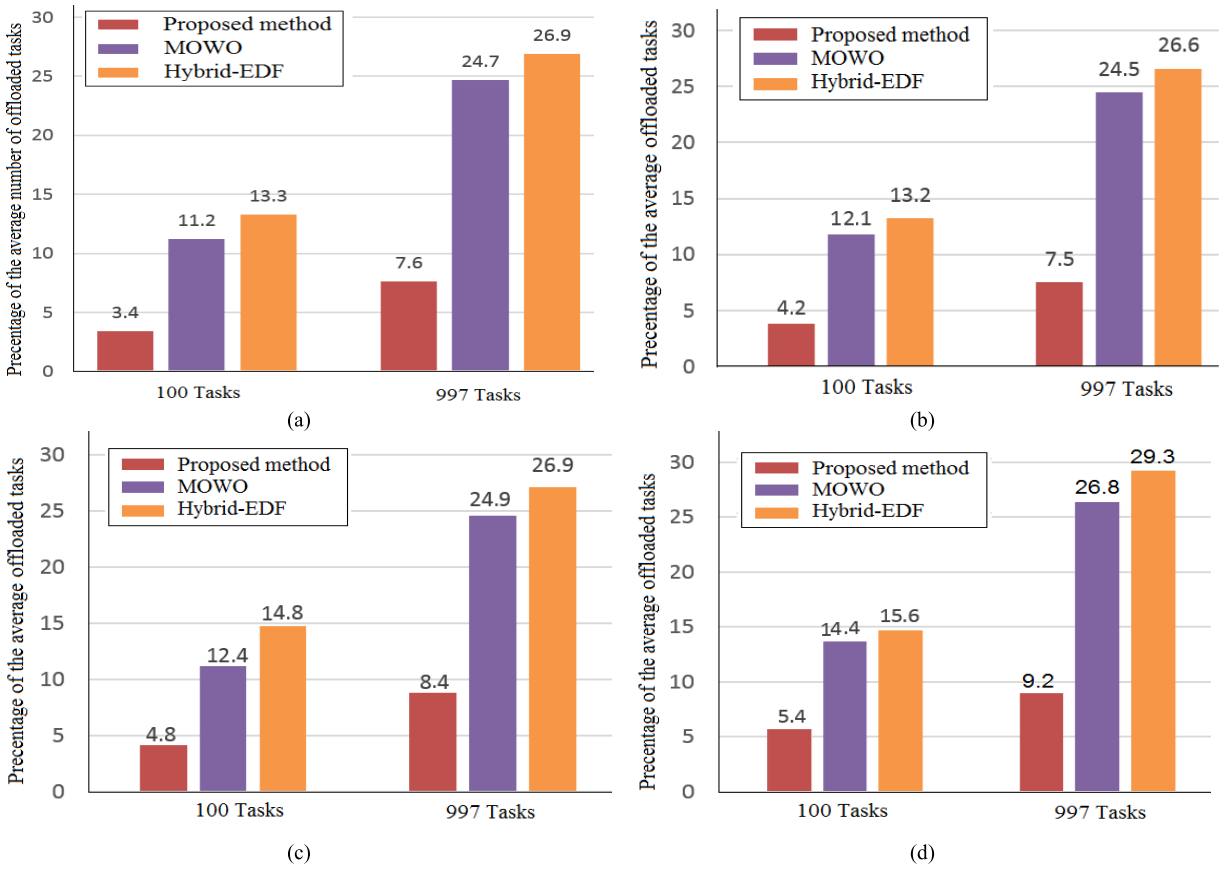


FIGURE 17. The percentage of the average number of offloaded tasks of Montage workflows on cloud computing.

TABLE 5. Percentage of the average number of offloaded tasks of the Epigenomics workflows on the cloud.

	100 Tasks			1000 Tasks		
Environments	Proposed method	MOWO	Hybrid-EDF	Proposed method	MOWO	Hybrid-EDF
Fog 1	4.3	12.6	14.1	8.6	25.4	27.8
Fog 2	4.5	12.9	14.7	4.5	12.9	14.7
Fog 3	4.1	12.3	13.8	4.1	12.3	13.8
Fog 4	3.9	11.9	13.6	3.9	11.9	13.6

TABLE 6. Percentage of the average number of deadline missed workflows.

	100 Tasks			1000 Tasks		
Environments	Proposed method	MOWO	Hybrid-EDF	Proposed method	MOWO	Hybrid-EDF
Fog 1	2.3	8.5	10.8	2.3	8.5	10.8
Fog 2	2.2	8.4	10.5	2.2	8.4	10.5
Fog 3	1.9	7.8	9.7	1.9	7.8	9.9
Fog 4	2	7.9	10.3	2.1	8.1	10.2

As shown in this figure, the proposed scheme suffers less from the deadline miss problem; however, in comparison to the Epigenomics workflow results, more workflows have missed their deadlines. The main reason for this issue is the asymmetric structure of the Montage workflow, which needs more VMs in the first two levels. Nonetheless, the proposed scheme's results are far better than MOWO, and the Hybrid-EDF, even for the Montage workflow.

B. DISCUSSION

From the results of different experiments, it can be concluded that in comparison to MOWO and Hybrid-EDF, the proposed scheme can effectively mitigate the number of deadline-missed workflows and the number of offloaded tasks on cloud computing data centers. These results were achieved because of the following two reasons:

- Two proposed Markov models for computing the average number of VMs and average bandwidth of fogs.
- SSPSO algorithm.

As outlined before, using the proposed Markov models, the proposed scheme can compute the average number of VMs available for each fog. However, since the compared methods have no such capabilities, they cause some

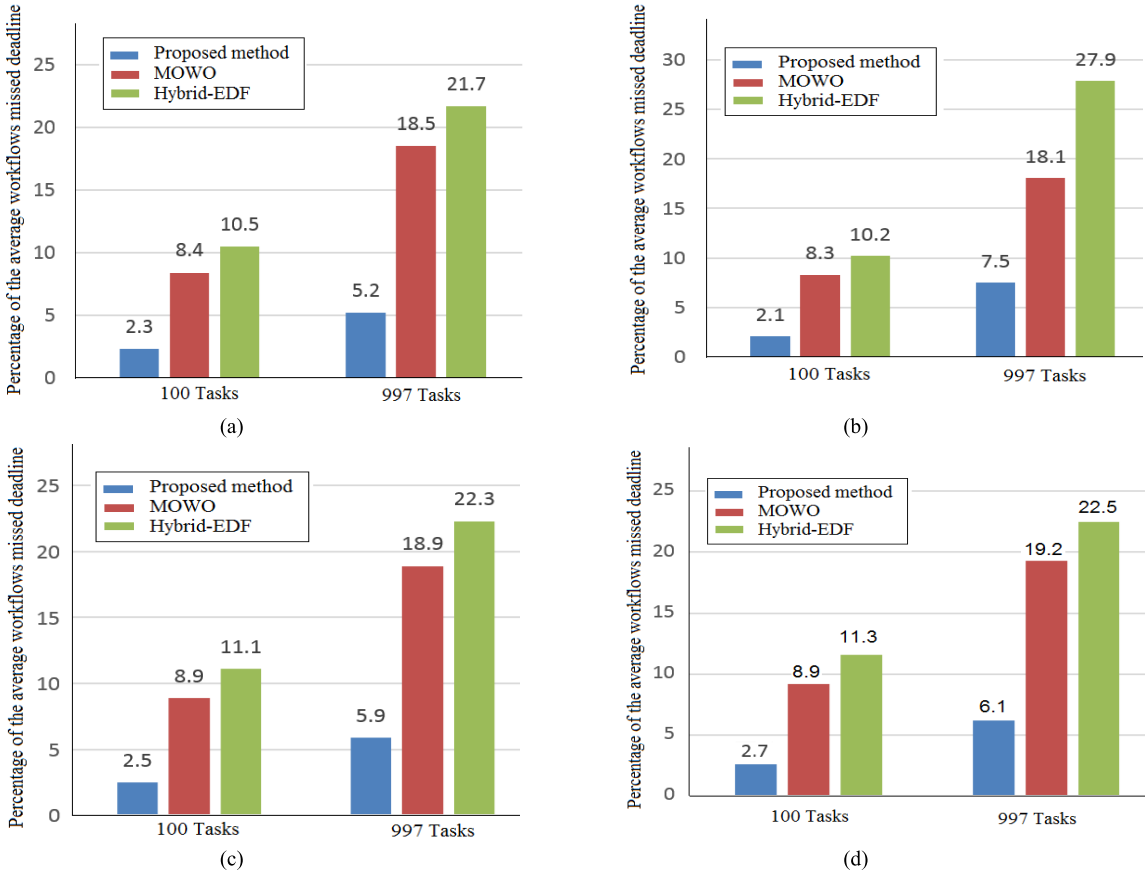


FIGURE 18. The percentage of the average number of deadline missed Montage workflows.

problems, and as a result, some workflows may miss their deadlines. They also suffer from more offloaded tasks, which incurs more network traffic and increases the load of cloud data centers.

VII. CONCLUSION

Fog computing is an interesting architecture designed to support the Internet of Things (IoT)'s increasing demand for more resources and processing power. Fog computing resources should be located near the IoT networks to minimize communication delays and reduce the cloud computing infrastructure's processing load. Furthermore, to provide practical support for the IoT, fog's virtual resources should be appropriately managed by effective scheduling of IoT offloaded tasks.

This paper proposes a new hybrid optimization algorithm by combining two existing state-of-the-art optimization algorithms: PSO and SSA. In our proposed algorithm, each of the incorporated optimization algorithms operates on almost half of the population. Afterward, when an algorithm finds better results, our proposed algorithm assigns more population to it to achieve better results. For providing equal opportunities for each algorithm to search the problem space and work on different solutions, every seven-round, the two sub-populations are combined and divided again. Besides, in each round, both algorithms stochastically exchange one of these solutions: the best solution, a random solution, or a solution achieved by the roulette wheel.

The proposed optimization algorithm is used to schedule IoT-submitted workflows on several fog computing environments that support the IoT network and minimize the workflow makespan and number of VMs applied in fogs. Generally, the performance of fog computing environments may be affected by DDoS attacks. In this scheme, two discrete Markov chain models have been proposed to compute the average number of VMs available for each fog environment to deal with this issue. Using these Markov models, a decent number of resources can be allocated to run the workflow, without over-estimating the number of VMs on fog computing environments. The experiments conducted on the iFogSim simulator show that the proposed workflow scheduling approach could effectively mitigate the makespan of the workflow while minimizing the number of offloaded tasks to cloud data centers and the number of deadlines missed workflows.

Since the task and workflow scheduling research in fog computing is not mature enough, further studies in this context seem to be necessary. For this purpose, in future studies,

the proposed algorithm's multi-objective version needs to be explored, and the best possible Pareto front needs to be extracted. Also, other stochastic models can be useful in dealing with dynamic fog environments.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[2] M. Masdari, S. M. Bazarchi, and M. Bidaki, "Analysis of secure LEACH-based clustering protocols in wireless sensor networks," *J. Netw. Comput. Appl.*, vol. 36, no. 4, pp. 1243–1260, Jul. 2013.

[3] S. Li, L. Da Xu, and S. Zhao, "The Internet of Things: A survey," *Inf. Syst. Frontiers*, vol. 17, no. 2, pp. 243–259, 2015.

[4] M. Masdari and M. Zangakani, "Efficient task and workflow scheduling in inter-cloud environments: Challenges and opportunities," *J. Supercomput.*, vol. 76, no. 1, pp. 499–535, Jan. 2020.

[5] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan, "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE Access*, vol. 6, pp. 47980–48009, 2018.

[6] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 416–464, 1st Quart., 2018.

[7] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[8] M. Masdari and M. Zangakani, "Green cloud computing using proactive virtual machine placement: Challenges and issues," *J. Grid Comput.*, pp. 1–33, Aug. 2019.

[9] M. Masdari, S. S. Nabavi, and V. Ahmadi, "An overview of virtual machine placement schemes in cloud computing," *J. Netw. Comput. Appl.*, vol. 66, pp. 106–127, May 2016.

[10] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities," *Future Gener. Comput. Syst.*, vol. 87, pp. 278–289, Oct. 2018.

[11] B. B. Gupta and O. P. Badve, "Taxonomy of DoS and DDoS attacks and desirable defense mechanism in a cloud computing environment," *Neural Comput. Appl.*, vol. 28, no. 12, pp. 3655–3682, Dec. 2017.

[12] A. Shameli-Sendi, M. Pourzandi, M. Fekih-Ahmed, and M. Cheriet, "Taxonomy of distributed denial of service mitigation approaches for cloud computing," *J. Netw. Comput. Appl.*, vol. 58, pp. 165–179, Dec. 2015.

[13] M. Masdari and M. Jalali, "A survey and taxonomy of DoS attacks in cloud computing," *Secur. Commun. Netw.*, vol. 9, no. 16, pp. 3724–3751, Nov. 2016.

[14] M. Masdari and H. Khezri, "A survey and taxonomy of the fuzzy signature-based intrusion detection systems," *Appl. Soft Comput.*, vol. 92, Jul. 2020, Art. no. 106301.

[15] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, "A survey of intrusion detection techniques in cloud," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 42–57, Jan. 2013.

[16] S. Huang, A. Liu, S. Zhang, T. Wang, and N. Xiong, "BD-VTE: A novel baseline data based verifiable trust evaluation scheme for smart network systems," *IEEE Trans. Netw. Sci. Eng.*, early access, Aug. 7, 2020, doi: 10.1109/TNSE.2020.3014455.

[17] J.-H. Cho, A. Swami, and R. Chen, "A survey on trust management for mobile ad hoc networks," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 4, pp. 562–583, 4th Quart., 2011.

[18] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Gener. Comput. Syst.*, vol. 107, pp. 841–853, Jun. 2020.

[19] M. Masdari, S. ValiKardan, Z. Shahi, and S. I. Azar, "Towards workflow scheduling in cloud computing: A comprehensive analysis," *J. Netw. Comput. Appl.*, vol. 66, pp. 64–82, May 2016.

[20] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, "Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds," *Future Gener. Comput. Syst.*, vol. 29, no. 1, pp. 158–169, Jan. 2013.

[21] P. Varshney and Y. Simmhan, "Characterizing application scheduling on edge, fog, and cloud computing resources," *Softw. Pract. Exper.*, vol. 50, no. 5, pp. 558–595, May 2020.

[22] M. Masdari, F. Salehi, M. Jalali, and M. Bidaki, "A survey of PSO-based scheduling algorithms in cloud computing," *J. Netw. Syst. Manage.*, vol. 25, no. 1, pp. 122–158, Jan. 2017.

[23] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Adv. Eng. Softw.*, vol. 114, pp. 163–191, Dec. 2017.

[24] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw. (ICNN)*, 1995, pp. 1942–1948.

[25] 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," *Softw. Pract. Exper.*, vol. 47, no. 9, pp. 1275–1296, Sep. 2017.

[26] D. P. Abreu, K. Velasquez, M. R. Miranda Assis, L. F. Bittencourt, M. Curado, E. Monteiro, and E. Madeira, "A rank scheduling mechanism for fog environments," in *Proc. IEEE 6th Int. Conf. Future Internet Things Cloud (FiCloud)*, Aug. 2018, pp. 363–369.

[27] V. Cardellini, V. Grassi, F. L. Presti, and M. Nardelli, "On QoS-aware scheduling of data stream applications over fog computing infrastructures," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2015, pp. 271–276.

[28] Y.-C. Chen, Y.-C. Chang, C.-H. Chen, Y.-S. Lin, J.-L. Chen, and Y.-Y. Chang, "Cloud-fog computing for information-centric Internet-of-Things applications," in *Proc. Int. Conf. Appl. Syst. Innov. (ICASI)*, May 2017, pp. 637–640.

[29] S. Kabirzadeh, D. Rahbari, and M. Nickray, "A hyper heuristic algorithm for scheduling of fog networks," in *Proc. 21st Conf. Open Innov. Assoc. (FRUCT)*, 2017, pp. 148–155.

[30] J. Wang and D. Li, "Task scheduling based on a hybrid heuristic algorithm for smart production line with fog computing," *Sensors*, vol. 19, no. 5, p. 1023, Feb. 2019.

[31] J. Ge, B. Liu, T. Wang, Q. Yang, A. Liu, and A. Li, "Q-learning based flexible task scheduling in a global view for the Internet of Things," *Trans. Emerg. Telecommun. Technol.*, p. e4111, Sep. 2020.

[32] X. Liu, H. Song, and A. Liu, "Intelligent UAVs trajectory optimization from space-time for data collection in social networks," *IEEE Trans. Netw. Sci. Eng.*, early access, Aug. 19, 2020, doi: 10.1109/TNSE.2020.3017556.

[33] H. Chen, X. Zhu, G. Liu, and W. Pedrycz, "Uncertainty-aware online scheduling for real-time workflows in cloud service environment," *IEEE Trans. Services Comput.*, early access, Aug. 21, 2019, doi: 10.1109/TSC.2018.2866421.

[34] H.-Y. Wu and C.-R. Lee, "Energy efficient scheduling for heterogeneous fog computing architectures," in *Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Jul. 2018, pp. 555–560.

[35] Y. Yang, S. Zhao, W. Zhang, Y. Chen, X. Luo, and J. Wang, "DEBTS: Delay energy balanced task scheduling in homogeneous fog networks," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2094–2106, Jun. 2018.

[36] L. Yin, J. Luo, and H. Luo, "Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4712–4721, Oct. 2018.

[37] Z. Liu, X. Yang, Y. Yang, K. Wang, and G. Mao, "DATS: Dispersive stable task scheduling in heterogeneous fog networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3423–3436, Apr. 2019.

[38] G. Zhang, F. Shen, N. Chen, P. Zhu, X. Dai, and Y. Yang, "DOTS: Delay-optimal task scheduling among voluntary nodes in fog networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3533–3544, Apr. 2019.

[39] C. Zhu, J. Tao, G. Pastor, Y. Xiao, Y. Ji, Q. Zhou, Y. Li, and A. Ylä-Jääski, "Folo: Latency and quality optimized task allocation in vehicular fog computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4150–4161, Jun. 2019.

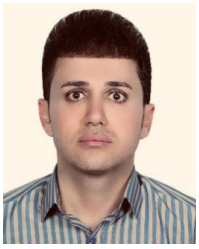
[40] G. L. Stavrinides and H. D. Karatza, "A hybrid approach to scheduling real-time IoT workflows in fog and cloud environments," *Multimedia Tools Appl.*, vol. 78, no. 17, pp. 24639–24655, Sep. 2019.

[41] H. R. Boveiri, R. Khayami, M. Elhoseny, and M. Gunasekaran, "An efficient swarm-intelligence approach for task scheduling in cloud-based Internet of Things applications," *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 9, pp. 3469–3479, Sep. 2019.

[42] N. Mohan and J. Kangasharju, "Edge-fog cloud: A distributed cloud for Internet of Things computations," in *Proc. Cloudification Internet Things (CIoT)*, Nov. 2016, pp. 1–6.

[43] J. Fan, X. Wei, T. Wang, T. Lan, and S. Subramaniam, "Deadline-aware task scheduling in a tiered IoT infrastructure," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–7.

[44] T. Choudhari, M. Moh, and T.-S. Moh, "Prioritized task scheduling in fog computing," in *Proc. ACMSE Conf.*, 2018, pp. 1–8.
[45] V. De Maio and D. Kimovski, "Multi-objective scheduling of extreme data scientific workflows in fog," *Future Gener. Comput. Syst.*, vol. 106, pp. 171–184, May 2020.



OMED HASSAN AHMED (Member, IEEE) received the B.Sc. degree in information technology from Teesside University, U.K., the Higher National Diploma (HND) degree in information technology from Teesside University, and the M.Sc. degree in computer science from Newcastle University, U.K. He is currently the Head of the Information Technology Department, College of Science and Technology, University of Human Development in Northern Iraq, where he has been a Faculty Member, since 2013. He is currently a Ph.D. Researcher with Huddersfield University, U.K.

JOAN LU is currently with the Department of Computer Science and is the Research Group Leader of Information and System Engineering (ISE) with the Centre of High Intelligent Computing (CHIC), having previously been the Team Leader with the IT Department of Charlesworth Group publishing company. She has successfully led and completed two research projects in the area of XML database systems and document processing in collaboration with Beijing University. Both systems were deployed as a part of the company's commercial productions. She has published seven academic books and more than 200 peer-reviewed academic articles. Her research publications have 1388 reads and 185 citations by international colleagues, according to incomplete statistics from the research gate.

ARAM MAHMOOD AHMED (Member, IEEE) received the M.Sc. degree in computer network technology from Northumbria University, U.K. He is currently pursuing the Ph.D. degree in swarm intelligence with KISSR. He is also working with KISSR. He is interested in modeling biological and natural systems into computational techniques.

AMIR MASOUD RAHMANI received the B.S. degree in computer engineering from Amir Kabir University, Tehran, in 1996, the M.S. degree in computer engineering from the Sharif University of Technology, Tehran, in 1998, and the Ph.D. degree in computer engineering from IAU University, Tehran, in 2005. He is currently a Professor with the Department of Computer Engineering, IAU University. He is the author/coauthor of more than 200 publications in technical journals and conferences. His research interests are in the areas of distributed systems, the Internet of Things, and evolutionary computing.

MEHDI HOSSEINZADEH received the B.S. degree in computer hardware engineering from Islamic Azad University, Dezfol Branch, Iran, in 2003, and the M.Sc. and Ph.D. degrees in computer system architecture from the Science and Research Branch, Islamic Azad University, Tehran, Iran, in 2005 and 2008, respectively. He is currently an Associate Professor with the Iran University of Medical Sciences (IUMS), Tehran. He is the author/coauthor of more than 120 publications in technical journals and conferences. His research interests include SDN, information technology, data mining, big data analytics, E-commerce, E-marketing, and social networks.

MOHAMMAD MASDARI received the B.Tech. degree in computer software engineering from Islamic Azad University, Qazvin Branch, Iran, in 2001, the M.Tech. degree in computer software engineering from Islamic Azad University, South Tehran Branch, Tehran, Iran, in 2003, and the Ph.D. degree in computer software engineering from Islamic Azad University, Science and Research Branch, Tehran, in 2014. Since 2003, he has been working as a Faculty Member of Islamic Azad University, Urmia Branch, Iran. He is currently an Assistant Professor with the Department of Computer Engineering, Islamic Azad University, Urmia Branch. His research interests include distributed systems and network security.