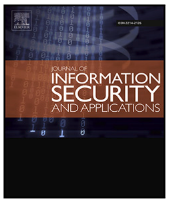
Journal of Information Security and Applications 60 (2021) 102853

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa



FUPE: A security driven task scheduling approach for SDN-based IoT–Fog networks

Saeed Javanmardi [a], Mohammad Shojafar [b], Reza Mohammadi [c], Amin Nazari [d], Valerio Persico [a,*], Antonio Pescapè [a]

[a] Department of Electrical Engineering and Information Technologies (DIETI), University of Napoli "Federico II", Italy
[b] 5GIC & 6GIC, Institute for Communication Systems (ICS), University of Surrey, Guildford, UK
[c] Bu-Ali Sina university, computer engineering department, Iran
[d] Arak university, computer engineering department, Iran

ARTICLE INFO

Keywords:
Internet of Things (IoT)
Fog computing
Software defined networking (SDN)
Resource management
Multi-objective particle swarm optimization (MOPSO)
Fuzzy logic

ABSTRACT

Fog computing is a paradigm to overcome the cloud computing limitations which provides low latency to the users' applications for the Internet of Things (IoT). Software-defined networking (SDN) is a practical networking infrastructure that provides a great capability in managing network flows. SDN switches are powerful devices, which can be used as fog devices/fog gateways simultaneously. Hence, fog devices are more vulnerable to several attacks. TCP SYN flood attack is one of the most common denial of service attacks, in which a malicious node produces many half-open TCP connections on the targeted computational nodes so as to break them down. Motivated by this, in this paper, we apply SDN concepts to address TCP SYN flood attacks in IoT–fog networks . We propose FUPE, a security-aware task scheduler in IoT–fog networks. FUPE puts forward a fuzzy-based multi-objective particle swarm Optimization approach to aggregate optimal computing resources and providing a proper level of security protection into one synthetic objective to find a single proper answer. We perform extensive simulations on IoT-based scenario to show that the FUPE algorithm significantly outperforms state-of-the-art algorithms. The simulation results indicate that, by varying the attack rates, the number of fog devices, and the number of jobs, the average response time of FUPE improved by 11% and 17%, and the network utilization of FUPE improved by 10% and 22% in comparison with Genetic and Particle Swarm Optimization algorithms, respectively.

1. Introduction

Recent advances in smart devices and communication technologies are fueling the Internet of Things (IoT) paradigm [1], which is characterized by the pervasive presence around people of (interconnected and uniquely addressable) things, able to measure and modify the environment and communicate with each other. Accordingly, technology leaders, governments, and researchers are putting serious efforts to develop solutions enabling wide IoT deployment in order to support a variety of applications impacting a number of different scenarios (e.g. healthcare [2], industry 4.0 [3], smart home [4]). Often, resource-constrained things are required to interact with service platforms in order to benefit from their computation, storage, and networking capability on request. While on the one hand referring to cloud services is the natural choice [5], on the other hand, some classes of applications may suffer from the performance figures that cloud platforms may guarantee in terms of provided QoS (e.g. due to latency to reach far-away cloud data centers). To address this issue, fog computing has been proposed [6], which provide the tools to mitigate cloud QoS shortcomings at the expenses of not having available virtually infinite resources of cloud data centers. In fact, fog computing is a cutting edge solution, which leverages near-user (possibly cooperating) edge devices (fog devices) rather than a single (far-away) cloud data center to supply computing services to IoT applications. [7]. Hence, fog infrastructures allow for supporting IoT application requirements to reduce both delay and network utilization [8]. Usually, cloud data centers still take part in composing the overall fog architectures, but are accessed only in case of need (i.e. when the fog nodes capabilities are not enough). In the context of fog computing, resource management is a challenging issue to be considered. Task scheduling is the major part of a resource management unit that aims to assign a set of tasks to fog devices [9].

* Corresponding author.
E-mail addresses: saeed.javanmardi@unina.it (S. Javanmardi), m.shojafar@surrey.ac.uk (M. Shojafar), r.mohammadi@basu.ac.ir (R. Mohammadi), aminnazari91@gmail.com (A. Nazari), valerio.persico@unina.it (V. Persico), pescape@unina.it (A. Pescapè).

https://doi.org/10.1016/j.jisa.2021.102853

Available online 19 May 2021

To this end, the task scheduler —possibly located at different places in a fog architecture, such as fog gateways, fog devices, cloud gateway or cloud data center— decompresses the jobs into a set of (independent) tasks, and then it assigns them to the fog devices [10].

Besides their requirements in terms of QoS, IoT and related fog infrastructures are prone to security and privacy concerns due to the critical nature of the contexts the applications are deployed in and the generated data (e.g., smart home, healthcare etc.). These concerns potentially derive from low-cost hardware and software design choices of IoT devices (e.g. unsafe update mechanism, outdated component, etc.) or lack of adequate security protection [11]. Thus, malicious users have shifted the main target of daily-released malware, now pointing to infect IoT services and make them unavailable, leveraging the manifold and significant weaknesses generated by the IoT context. IoT and fog devices are both susceptible to be hacked by malicious users [12,13]. These devices may be even incorporated in botnets, thus unwillingly participating to the attack after they have been compromised [14] (e.g., taking part to Distributed Denial of Service DDoS attacks which aim at disrupting targeted server/service by overwhelming the target with a flood of malicious Internet traffic [15,16]). As the Software-defined networking (SDN) [17] provides flexible network programmability and logically centralized control (through a global view of the network), this paradigm is able to provide the tools for effectively detecting and containing network security problems that recently is used in IoT–fog networks [18–20]. Indeed, SDN represents a good fit for the fog environment as fog devices and gateways have the capabilities to implement this paradigm [21].

A possible solution to detect and mitigate TCP SYN flood attack is using firewalls deployed at fog gateways. With the firewall filtering out malicious requests, the scheduler can put away the attacker nodes. Firewalls filter the traffic by using some predefined rules and act like a checkpoint. Firewalls detect flood attacks in IoT–fog networks by using different mechanisms such as a predefined threshold, specifying port addresses, or defining rules to filter protocols, ports, IP addresses, or network traffic. For instance, in a threshold-based firewall if the number of TCP SYN exceeds 10 (the justification can be applied), the firewall detects the node as an attacker. The major limitation of threshold-based approaches is that they do not provide the sensitivity and specificity required for precise classification. A threshold-based approach cannot distinguish the flash traffic (i.e., sudden traffic from a legitimate source node to a particular destination node) from malicious traffic. Moreover, the difference between 9 and 10 is very smidgen; but a threshold-based firewall detects 9 as a benign request, and 10 as a malicious one. Threshold realization is a classic technique for firewall malicious behavior detection. However, different firewalls nowadays utilizes some cutting edge methods. Recent works on cutting-edge firewall use hybrid techniques to mitigate malicious traffic using pattern of machine learning models [22]. On the other side, they have several gaps especially for real-time stream traffic applications. A typical method is to use a regular scheduler plus a firewall. With the firewall filtering out malicious requests, a regular scheduling algorithm can schedule the remaining benign ones. While this approach is legitimate, in this work we investigate the feasibility and the performance of an integrated solution to meet the requirements of real-time applications. In detail, we make use of fuzzy logic capabilities and consider security in the scheduler deployed at fog gateways. FUPE can be implemented in firewalls as well.

In accordance with the context above, intelligent and efficient solutions are required to detect and protect against DDoS attacks in their IoT–fog networks [23,24]. However considering task scheduling efficiency and security poses a non-trivial challenge to task scheduling algorithms [25,26] that are required to strike a balance between these two distinct objectives. As fog environments have dynamic features in which the characteristics of elements change continuously, techniques to jointly preserve trust and security in such dynamic environments are required [27]. Since IoT–fog network provides a shared and distributed infrastructure for the users, establishing security for the fog devices must be considered during the scheduling process. Existing IoT scheduling algorithms consider efficiency and disregard security concerns of resources. Recently, more attention has been paid to security-aware IoT–fog task scheduling algorithms [25,26,28]. However, these studies did not touch DDOS attacks issues. Besides, as security and efficiency are considered separately in their frameworks, these studies are not suitable for real-time applications. Motivated by these challenges, in this paper we integrate DDoS attack detection techniques in a scheduling algorithm. To the best of our knowledge, FUPE is the first attempt to perform the security-aware task scheduling in IoT–fog networks considered TCP SYN flood attack through a multi objective optimization algorithm. FUPE finds the best place for users' applications based on security consideration and resource performance.

1.1. Contribution of the paper

In this paper, we propose a security-aware task scheduler algorithm tailored for IoT–fog networks called *FUPE*, which provides a proper security level. This scheduler considers the dynamic behavior of distributed systems and uses two trust degrees obtained from Threshold Random Walk with Credit-Based connection rate-limiting (TRW-CB) and Rate Limiting algorithms— i.e. one of the most prominent source-based attack mitigation strategies available [29,30]—to deal with TCP SYN flood DDoS attacks [31]. Our proposal jointly merges security issues and task scheduling with the aid of multi-objective PSO [32] and multi-criteria decision-making [33] algorithms. FUPE solves multi-objective task scheduling problem to jointly maximize security and efficiency of quality of services (QoS) such as delay in IoT–fog networks. It also leverages SDN programmability and centralized network features to enforce attack mitigation mechanisms against the TCP SYN flood DDoS attacks: in case of detection of requests from devices identified as malicious, the requests are not taken into account. In other words, FUPE addresses security issues inside the scheduler. More specifically, FUPE assigns the most suitable resources to the tasks based on the current status of the resources and incoming tasks. FUPE focuses on assigning applications' tasks among fog devices, considering the trustworthiness of fog devices, and users' devices. Thus, FUPE strikes a balance between efficiency and security objectives.

We evaluate FUPE leveraging Matlab [34], a widely adopted and well-known programming platform. To this end, we test FUPE against two well-known metaheuristic approaches *Genetic algorithm (GA)* and *particle swarm optimization (PSO)* strategies, by varying attack rates, number of fog devices, and number of jobs. The obtained results indicate that our proposal outperforms GA and PSO approaches in terms of Response time, and network utilization. In detail, FUPE improves by ≈14% Average Response time, ≈49% Maximum Response time, while concerning network utilization, FUPE improves this metric in by ≈16%, on average. Thus, the contributions of the paper are listed below.

- The paper presents an architecture which integrates the SDN and Fog technology in the presence of the IoT services and tackles the TCP SYN flood DDoS attacks.
- We design FUPE, a security-aware task scheduler algorithm, dealing with TCP SYN flood DDoS attack.
- FUPE combines fuzzy logic and a multi-objective particle swarm optimization (MOPSO) algorithms supporting secure IoT task demands in the network.
- FUPE uses Mamdani fuzzy inference system helping the scheduler using a relationship between the metrics and priority of the IoT demands [35,36].
- FUPE implemented and tested on various scenarios and compared against the state-of-the-art.

1.2. Organization

The remainder of the paper is as follows. Section 2 presents the related literature on cloud/fog task scheduling and provides their pros and cons. Section 3 overviews the presented architecture and its main components. Moreover, it presents the proposed methodology leveraging Multi-objective PSO and Fuzzy algorithms. Section 4 details the performance evaluation of this work with experimental results. In Section 5, we discuss security aware fog task scheduling challenges which are related to this work. Finally, Section 6 concludes the paper with some future directions.

2. Related work

In this section, first, we list fog task scheduling approaches (Section 2.1). Then, we explain the approaches focusing on TCP SYN flood attack in IoT/SDN networks (Section 2.2), and finally, we explain the security-aware fog task scheduling algorithms (Section 2.3). In the end, Table 1 summarizes the comparison of these categories.

2.1. Fog task scheduling approaches

In this subsection, we review some of the fog task scheduling algorithms. Bittencourt et al. [37] put forward *three* scheduling approaches, namely concurrent, first-input first-output (FIFO) and delay priority in IoT–fog network. In all of these methods, the scheduling algorithms run in fog devices (FD) such that when a new request arrives at the FD, the scheduler decides either runs it on the FD or send it to the cloud data center. These algorithms utilize resource availability and CPU capacity for decision-making. They validate their strategies in terms of application loop delay, network usage, and the number of applications moved to the cloud data center. Afterward, Mahmud et al. [38] introduced a latency-aware application scheduling algorithm in FD to assure deadline-satisfied service delivery for users' demands. They use deployment time, deadline, and some fog devices as the nature of observation and reduces the applications' service delivery latency. Similarly, Bitam et al. [39] performed a bio-inspired method using the Bee life scheduling algorithm to discover an optimal trade-off between CPU execution time and allocated memory of the users' requests in an FD and provides low execution time.

Gill et al. [4] designed a task scheduling in FDs using particle swarm optimization (PSO) algorithm to mutually reduce network bandwidth, response time, latency, and energy consumption. Most recently, the same authors of this paper (i.e., FUPE) [40] proposed FPFTS, a meta-heuristic method merging PSO and fuzzy methods to tackle the fog task scheduling problem. They implement and test FPFTS on different scenarios like on various mobile devices and FD characteristics when faced with various link delays. Nevertheless, none of these approaches except FPFTS, mutually consider the FD and application task characteristics to find efficient fog scheduling methods. The proposed FUPE comparing with FPFTS has *three* main differences. First, FUPE uses the remaining amount of RAM and CPU capacity for task scheduling. Second, FUPE uses the trust degree of FDs and users' devices for the scheduling that enables it to select trustworthy devices in the exact time of utilizing the devices. Third, FUPE imposes a balance between FDs' trustworthiness and efficiency with the aid of multi-objective PSO (MOPSO) features. Sun et al. [41] devised a two-level application scheduling approach in fog–IoT networks called RSS-IN. RSS-IN schedules the applications among the fog regions and then performs scheduling within the same fog region. This approach achieved proper stability and end-to-end delay through a non-dominated sorting genetic algorithm (NSGA-II).

2.2. TCP SYN flood attack in IoT/SDN approaches

In this part, several surveys summarize recent attack and defense techniques targeting distributed denial of service (DDoS) attack, especially TCP SYN flood attack in IoT and SDN [23,24,42]. In particular, Evmorfos et al. [43] designed a lightweight technique to detect TCP SYN flood attack in IoT network. To this end, the authors apply the long-short-term-memory and the random neural network to develop the predictive model mitigating SYN TCP attack detection. In another work, Kolias et al. [44] designed Mirai-like bots as the IoT-based DDoS attacks real examples. The Mirai is a malware that causes severe disruptions in the IoT network services owing to its sophisticated spreading mechanism. Mirai uses bot instances that attack the target computational resources with TCP SYN flood attack.

Some other related works mainly focus on TCP SYN flood attack/defense in the SDN network. For example, Yan et al. [45] proposed a TCP SYN flood mitigation multi-level framework with the aid of SDN to control the Industrial IoT network. To accomplish this, their method monitors the SDN gateways, gather the network traffic data and detect the DDoS attack. To confirm their method, they evaluate it based on time-delay of normal users in the context of no defense mechanisms is used, and in the context of the presence, SDN to mitigate the attacks. Alternatively, Kumar et a. [46] proposed, SAFETY, an approach to detect and mitigate TCP SYN flood attack in SDN networks. SAFETY considers destination IP and a few attributes of TCP flags. The authors implement SAFETY as a module in the Floodlight OpenFlow controller. Mohammadi et al. [31] presented an approach to mitigate TCP flood attack in SDN called *SLICOTS*. SLICOTS supervises the ongoing TCP connection attempts and detects malicious hosts. It utilizes the programmability features of the SDN to tackle the TCP SYN flooding attack. Later on, they [47] proposed, SYN-Guard, a countermeasure approach to detect and prevent TCP SYN flood attack in an SDN network. They implement SYN-Guard as an extension module on the SDN controller to monitor the incoming TCP connection attempts. SYN-Guard is evaluated based on average response time. Recently, Zhou et al. [48] developed a three-way approach to mitigate DDOS attacks in industrial IoT–fog networks. This approach considers the prevailing edge devices' capacities, such as local proxy servers, firewalls, IDS, etc. They take TCP SYN flood and Modbus flood attacks into consideration in the implementation and show that their method has a low detection time. In this paper, unlike the solutions above in this category, FUPE aims to detect and mitigate TCP SYN flood attack by mutually considering malicious users' devices and compromised fog devices in task scheduling. It enhances the balanced task scheduling on available and benign FDs.

2.3. Security-aware fog task scheduling approaches

This part is devoted to summarize the fog task scheduling approaches considering security challenges. Sujana et al. [25] designed a platform that helps to achieve convincing secure scheduling relying on a stochastic behavior of workflow in cloud/fog environment. They emphasize direct trust and indirect reputation metrics for task scheduling to preserve security in the cloud/fog. This scheme assures that the assigned resource only retain the trusted VMs. In their implementation, they consider a distinct security guaranteed level to specify the percentage of security ratio for each VM. Also, they address various related attacks, such as VM theft, VM escape, hyper jacking, data leakage attacks. Besides, Daoud et al. [28] offered a security model for fog resource management and task scheduling. Their security model integrates access control to ensure secure cooperation between diverse resources and tasks. The scheduler assigns user tasks to the resources based on the trust degree of each users' devices and the availability of resources. They compute the trust degree of users' devices based on the behaviors of each user, and the access control considers these degrees of trust to permit trustworthy users to access the FDs.

Lately, Auluck et al. [26] introduced a secure fog task scheduling algorithm considering deadline and security constraints of IoT applications. This algorithm assigns *three* security labels to the tasks as follow, namely private, semi-private, and public. Similarly, it assigns *three* security labels to the resources. The scheduler assigns tasks to the resources based on these labels and the application deadline. It uses jobs deadlines, spare capacity available on the resources and job privacy for assigning tasks to them. Unlike these methods, in this study, FUPE uses the available amount of CPU and RAM space of FDs, tasks CPU requirements, and also trust degree of devices altogether as the nature of observations. In this way, the scheduler is aware of the current state of resources' trust degrees.

2.4. Evaluating of the background methods

FUPE, differently from the works by Gill et al. [4] and Li et al. [49], makes use of fuzzy logic in the MOPSO fitness function. These works utilize the weighted sum approach to prioritize the objectives in the PSO fitness function. RSS-IN [41] takes the distance between the requester and the fog regions into account. Unlike this work, FUPE considers link bandwidth to reduce response time. In this paper, unlike our former work [40] in which we only considered efficiency objective, we take security and efficiency objectives into consideration. The works by Sujana et al. [25], Daoud et al. [28], Rjoub et al. [50], Li et al. [49], and Auluck et al. [26] did not propose multi-objective optimization solutions. These works proposed single objective optimization algorithms for response time minimization with security constraints. They utilized optimization models to define security levels for tasks and resources. Based on the tasks' security requirements, these algorithms assign resources to applications' tasks. The work by Gill et al. [51] has presented a single-objective resource management approach that offers self-protection against security attacks. Different from these works, FUPE utilizes a multi-objective algorithm. The work by Bittencourt et al. [37] did not consider the processing capacities of the other fog devices. Different from this work, FUPE considers the processing capacities of the other fog devices in the same fog region. Unlike the work by Mahmud et al. [38] which focused on reducing latency, FUPE main aim is reducing latency and network congestion. Finally unlike the work by Bitam et al. [39] that provides low scalability, FUPE makes use of the P2P structure for nodal collaboration to provide high scalability.

The evaluated background methods in the mentioned works focus on various IoT scenarios except [49–51] which use cloud task schedulers. All the mentioned works considered security issues except [4,37–41], which relied on the security unit. Although the works [25,26,28] have touched the security challenges, they are susceptible to be downed by TCP SYN flood attack. Unlike all the mentioned works, FUPE integrates DDoS attack detection technique in a scheduling algorithm. This scheduler is more reliable for Scheduling real-time applications. Considering security issues in the implementation of task scheduler has two important advantages: first, it prevents the computation of malicious requests which may lead to overload the fog resources and as a consequence cause the starvation for benign requests. Second, it allows giving priority to the requests from users exposing legitimate behavior. Moreover, as FUPE utilizes a multi objective optimization algorithm, it prioritizes efficiency and security as the two different objectives in the scheduler.

3. Proposed approach

In this section, we detail the proposed FUPE. To this end, first we describe the reference architecture in Section 3.1. Then, we introduce the security-aware task scheduling problem we address and the resulting organization in modules we adopt for our proposal in Section 3.2. In Section 3.3, we give an account of TRW-CB and Rate Limiting techniques. Finally, we detail FUPE in Section 3.4.

3.1. Reference architecture

In this paper, we present a three-layer structure (i.e. consisting of cloud, fog, and IoT device layers) as that adopted in previous works [52,53]. The device layer consists of users' devices that send requests to fog devices in order to benefit from their computing resources. The fog layer is composed by a set of fog devices which are placed at the edge of the access network. Fog devices are grouped in fog regions: when a fog device becomes overloaded, it can offload requests to other fog devices *fog-to-fog* offloading) inside the same fog region. Finally, at the cloud layer virtually unlimited computing resources are organized in cloud data centers and are available to execute tasks which are passed from by the fog layer (*fog-to-cloud* offloading). Fig. 1 presents the architecture of the proposed approach. In our proposed architecture, the fog devices are clustered into virtual organizations (with each of them representing a fog region), and they are interconnected by SDN switches. The central controller of our scheme is located on the Cloud gateway. In this work, we consider the fog gateway and the cloud gateway as SDN switch and SDN controller, respectively.

Fog gateways play a central role as the instances of FUPE are implemented on them. The FUPE instances in fog gateways act as broker [54,55] between users' devices and fog devices. In fact, they are the interface between users' applications in IoT device layer and resources in fog layer. In a scenario where some of the devices/nodes may take part to attacks against the fog infrastructure, the fog gateways collaborate with the cloud gateway to calculate the trust degree of the nodes for task scheduling. Indeed, flooding attacks (such as TCP SYN flood attacks) generate a waste of resources, which may lead to denial of service. Accordingly, hereinafter we refer to these attacks as *malicious requests*.

3.2. Problem statement and proposed solution

Security-aware task scheduling is a critical part of IoT–fog networks and significantly impacts the performance of the overall system. In fact, it is a challenging NP-hard problem owing to the large-scale, dynamic, and heterogeneous architecture it relates to [56]. Indeed, users' devices and applications can join and leave the system in a dynamic manner and are both susceptible to be hacked.

The problem we address in this paper consists in efficiently and effectively assigning applications' tasks to fog devices, such that the security level constraints are met. Since various security threats are a big concern of IoT–fog networks, it is mandatory to deploy security mechanisms to protect security-critical users' applications executing on fog devices from being attacked. The aim of security-driven task scheduling is assigning tasks generated by trustful users' applications to the computing resources available at the trustworthy fog devices. In other words, applications which have an adequate level of trustworthiness are decomposed into a set of tasks to be assigned to the fog devices that have proper trust values and adequate computational capacity.

Our solution implements security-aware task scheduling at fog gateways via FUPE and benefits from SDN advantages. SDN infrastructure provides an efficient solution to mitigate security challenges in IoT–fog networks thanks to advanced network programmability, dynamic flow control, and network monitoring through centralized visibility. It enables administrators to automatically manage the entire IoT–fog network flexibly and dynamically (e.g. instantly block network traffic anomaly whenever malicious activity is detected) [57–59]. As shown in Fig. 2, our proposal consists of *five modules*: *Connection Logger*, *Request Validator*, *Job Decomposer*, *Task Assigner*, and *Forwarder*. Each module provides the functionalities as detailed in the following.

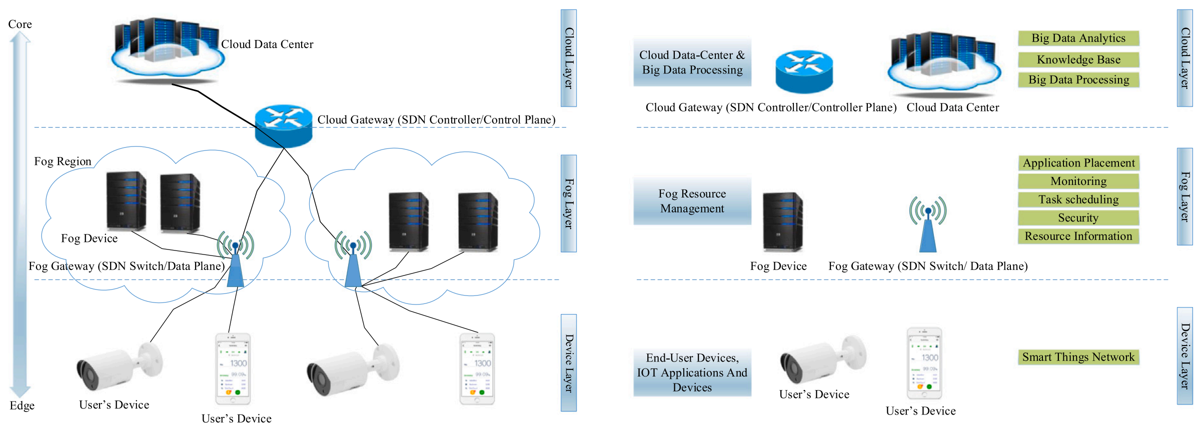
- *Connection Logger*: This module monitors the connection attempts from users' devices/fog devices, logs them, and provides the cloud gateway with this logged information. Based on this information, the cloud gateway updates the *Credit* and *Rate* values for each

Table 1
Comparison of existing scheduling approaches against FUPE.

Refs.	Algorithms	Tool	Scenario	Observations	Security	Advantages	Disadvantages
[37]	Concurrent/ FIFO/Delay-priority	iFogSim	IoT	Resources availability and CPU capacity	No	+ Movement based scheduling at fog device level	– High time complexity – Relying on security unit
[38]	Heuristic	iFogSim	IoT	Deployment time, deadline, Number of fog devices	No	+ latency-aware IoT application + Reducing the amount of deployment time + Deals with varying application	– No Real case study – Relying on security unit
[39]	Bees swarm	C++	IoT	CPU execution time, Allocated memory	No	+ Managing allocated memory + Low CPU execution time	– Only fog devices are used – Static scheduling – Relying on security unit
[4]	PSO	iFogSim	IoT	Response time, energy, latency, and network bandwidth	No	+ Energy + Latency and response time + Network bandwidth	– No reliability assurance – Relying on security unit
[41]	NSGA-II	MATLAB	IoT	Requester distance to the fog region Service completion time Fog device's capability Fog device's reliability	No	+ Low latency + Improved stability	– Not suitable for complex topology – Relying on security unit
[51]	Machine learning	Java	Cloud	Anomaly-based/Signature-based detector Execution time, Execution cost	Yes	+ delivers secure cloud services	– Centralized feature and not suitable for IoT
[49]	PSO	CloudSim	Cloud	Execution cost, deadline and risk rate	Yes	+ Risk rate constraint of workflow in scheduling	– Centralized feature and not suitable for IoT
[50]	Multi-criteria task priority	CloudSim	Cloud	Resources degree of trust Tasks priority level	Yes	+ Applying resources degree of trust in scheduling	– Centralized feature and not suitable for IoT
[25]	Stochastic	CloudSim	Cloud/Fog	Service Level Agreement (SLA), The earliest execution start time of tasks	Yes	+ Different levels of trust	– Lack of defense mechanisms to deal with attacks – Relying on feedback collected from users
[26]	Multi-criteria task priority	iFogSim	IoT	Job deadlines Spare capacity available on the resources Job privacy	Yes	+ Assigning security tag to the resources.	– Lack of defense mechanisms to deal with attacks
[28]	Multi-criteria task priority	iFogSim	IoT	Jobs trust level Jobs arrival time Resources availability	Yes	+ Assigning security tag to the resources.	– Lack of defense mechanisms to deal with attacks – Relying on feedback collected from users
[40]	Hybrid (PSO-Fuzzy)	iFogSim	IoT	Total amount of fog devices' ram size, Total amount of fog devices' CPU capacity, Total amount of fog-devices' link bandwidth, Applications' Tasks CPU need	No	+ Mobility-aware scenario + Considering total computing capacity of resources + Low application loop delay/network utilization	– No fog gateways fault tolerance
FUPE	Hybrid (MOPSO-Fuzzy)	Matlab	IoT	Available fog-devices' processing capacity, Available fog-devices' RAM size, Available fog-devices' link bandwidth Applications' Tasks CPU need Trust degree of nodes	Yes	+ Security-aware scenario + Considering available computing capacity of resources + Low average response time/maximum response time + Low network utilization	– No SDN switches fault tolerance

user's device/fog device, i.e. the counters related to TRW-CB and Rate Limiting techniques, respectively, which indicate the trustworthiness likelihood ratio of the connection attempt requesters. Finally, these values are sent on request to both Request Validator module and Task Assigner module of the fog gateways.

- *Request Validator*: Authentication is one of the major subjects in the security field. It is the process of verifying the identity of a node (fog device/user's device) that is a prerequisite to allowing access to the fog layer/cloud layer resources. In this work, we assume the nodes have already been authenticated, and the authentication is not in the area of this research paper. Request Validator module is in charge of validating only users' devices. Upon the fog gateway receives a request from an user's device, this module first checks the validity of the requester. To this end, it asks the Cloud gateway to obtain the *Credit* and *Rate* values for the requester. Then, the module leverages the returned information and implements a Mamdani fuzzy inference system as shown in Fig. 5, and Table 3. In detail, this module uses the outcome of *security* objective to determine whether the requester is benign or it is an attacker. If the requester is benign then this module passes the job to the Decomposer module. Otherwise, this module rejects the request.
- *Job Decomposer*: This module receives the application jobs sent by the Request Validator module and then decomposes it into multiple separate tasks. After decomposition, the tasks are independent of each other and are ready to process by the assigner module.
- *Task Assigner*: This module assigns the application's tasks originated from the validated users' devices to the fog devices, with the aim of increasing performance securely. To this end, upon receiving application's tasks from the decomposer module, it first



(a) Three-tier IoT-Fog-Cloud architecture.

(b) Task-wise architecture.

Fig. 1. The considered FUPE architecture.

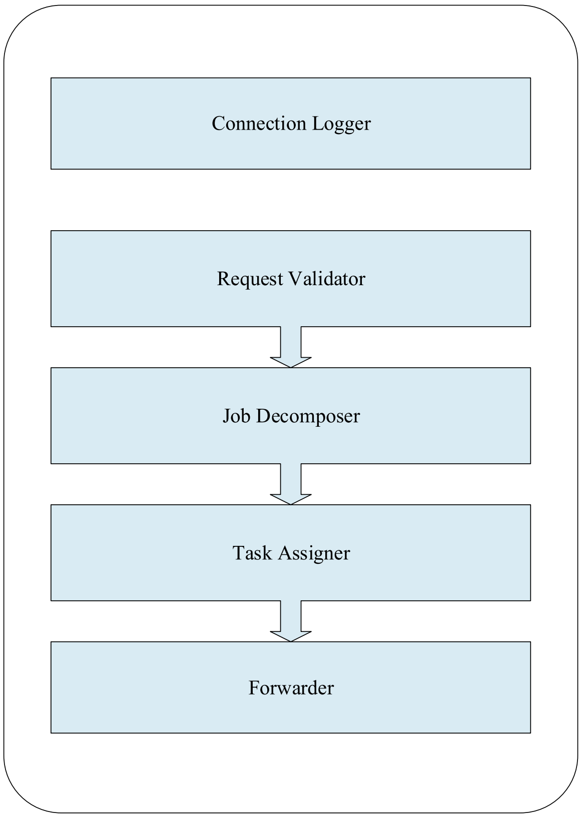


Fig. 2. Modules composing the proposed solution.

gathers required information regarding fog devices computational features together with their security conditions. Therefore, it sends a request to the Cloud gateway and asks the information about the *Credit* and *Rate* values for all fog devices in the fog region. The fog gateways have the information about the fog devices located inside the fog regions. Then, it performs a secure scheduling as described in Section 3.4. This module uses the outcome of both *security* and *efficiency* objectives to determine a secure and proper fog device to execute the validated application's tasks of user's device. This module provides the application's tasks and designated fog device's ID to the forwarder module.

- *Forwarder*: The forwarder module implements the features of the data plane of the SDN paradigm and, accordingly, provides a set of flow tables that are managed by the controller (Cloud gateway) [60]. These flow tables include flow entries containing a header to match the incoming packets and a set of actions to apply to match packets. Because preventing an attack is done at the fog gateways, the controller proactively installs flows on the forwarder modules. If a request is recognized as a legitimate by the Request Validator (and the Task Assigner), then the forwarder module forwards it toward designated fog device concerning the flows which the controller has installed on the forwarder module. This ensures more efficiency, because limits the required intervention of the controller. For example, in our proposed solution, when a fog device/user's device establishes a connection to another fog device successfully, the controller installs a flow rule as it is indicated in Table 2 to handle the packet in the session. Therefore, the controller fills six fields of the mentioned rule by some values. After that, the Forwarder can route the request toward the fog device based on its flow table.

3.3. Implementation of TRW-CB and rate limiting

Based on TRW-CB, the probability of a successful connection attempt is much higher for a trustworthy node than a malicious node. It considers a fog device/ an user's device as the malicious node whenever the number of TCP SYN packets sent from them to the fog gateway, is much higher, compared to the number of TCP SYNACK packets sent from the fog gateway to them. Rate Limiting is based on the idea that a malicious node tries to connect to (the) various nodes in a short period. In other words, it assumes that a malicious node is likely to make many connection-initiations in a short amount of time, compared to the benign node, which is unlikely to do so. Whenever the SDN switch receives a TCP packet, it checks the packet against a list of recently contacted fog devices called the *working set*. If the TCP packet request belongs to a fog device presented in the *working set*, then it forwards the request normally. Otherwise, it enqueues the request in another data structure called the *delay queue*. FUPE separates pairs of working sets and delay queues for every fog device. These connection-based techniques are scalable and can support a large number of rules which are suitable to be used in rule-based fuzzy techniques. It is illustrated in the literature that these techniques, on the one hand, process only a small fraction of the total network traffic; and on the other hand, they attain the same accuracy by inspecting every packet in SDN networks [59–61]. Rate limiting and TRW-CB do not create

Table 2
A flow rule in SDN switch example.

In_Port	Src_Mac	Dst_Mac	Layer2_Prot	Src_IP	Dst_IP	TCP_Src_Port	TCP_Dst_Port	Output
*	*	*	0X8000	X	Y	Z	T	3

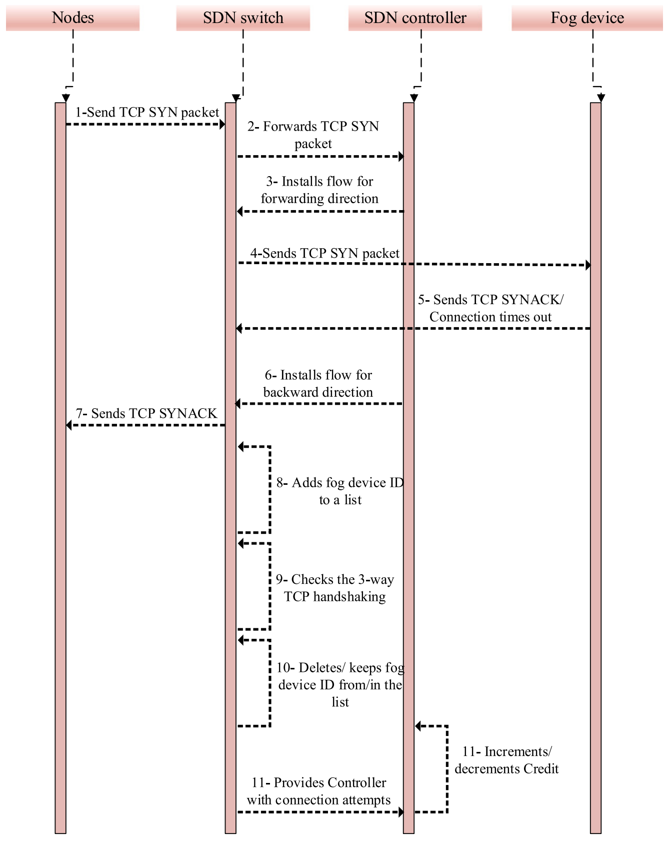

Nodes
SDN switch
SDN controller
Fog device
1-Send TCP SYN packet
2- Forwards TCP SYN packet
3- Installs flow for forwarding direction
4-Sends TCP SYN packet
5- Sends TCP SYNACK/ Connection times out
6- Installs flow for backward direction
7- Sends TCP SYNACK
8- Adds fog device ID to a list
9- Checks the 3-way TCP handshaking
10- Deletes/ keeps fog device ID from/in the list
11- Increments/ decrements Credit
11- Provides Controller with connection attempts

Fig. 3. The sequence diagram of TRW-CB algorithm.

additional overhead at the network level. So, our approach does not add a packet or header to the network. It only monitors the network traffic to evaluate the received packets; then decides whether to block them or let them pass.

TRW-CB is implemented as follows: (I), Suppose fog device/user's device *A* sends a TCP SYN packet to the fog device *B*. Since there are no flows in the fog gateway *C* matching this packet, the fog gateway forwards the packet to the cloud gateway (SDN controller). The controller Installs flow for forwarding direction in the SDN switch. (II), The FUPE instance on the cloud gateway forwards this packet, through the forwarder module on the fog gateway *C*, to fog device *B*. (III), the logger module adds fog device *B's ID* to the list of targets that fog device/user's device *A* attempted to contact. Based on the answer receiving from fog device B, two scenarios can happen as follow: (One) If the fog device/user's device *A* completes a 3-way TCP handshaking process with *B*, then the Logger module deletes the record from fog device/user's device *A* hosted queue and increments its balance. The controller Installs flow for backward direction in the switch as well. (Two) Otherwise, the controller performs counters processing for connection failure (i.e. it decrements fog device/user's device *A* balance). In this case, the controller does not set any flows in the SDN switch flow table.

Rate limiting is implemented as follows: (I), Suppose fog device/user's device *A* sends a TCP SYN packet to the fog device *B*, and fog device/user's device *A* is in the working set; Then the cloud gateway Installs flow for forwarding direction in the fog gateway. (II), *Working set* is a list of recently contacted fog devices by node *A*. If node B receives a new connection request from node *A*, and the node *B* is not in the working set, FUPE adds it into a queue named the delay queue. In this case, it does not install any flows in the fog gateway *C*. (III), Every *d* seconds, FUPE moves a new connection request from the delay queue to the working set. FUPE forwards the connection request through the fog gateway without installing any flows. The Logger module in fog gateway records the number of connection attempts during a specific period which can be placed in a range of fuzzy sets. (IV), Whenever node *B* sends a TCP SYNACK to the fog gateway, cloud gateway Installs flow for backward direction in the fog gateway.

Figs. 3 and 4 indicate the process diagram of TRW-CB and rate limiting approaches, respectively. Alg. 1 and Alg. 2 show TRW-CB and Rate limiting algorithms as well.

Algorithm 1 TRW-CB

```
Output: $Credit$
1:  List:= [];
2:  if packet.type!= TCP then
3:      return;
4:  end if
5:  if packet.flags != SYN or SYNACK then
6:      return;
7:  end if
8:  if packet.flags = SYN and not ACK then
9:      if packet.source not in List then
10:         Install flow for forwarding direction;
11:         List := packet;
12:         Else
13:          $Malicious\_Counter++$;
14:     end if
15: end if
16: if packet is SYNACK and packet source in list then
17:     Install flow for backward direction;
18:     $Safe\_Counter++$;
19: end if
20: $percentage$:= The percentage of normal connections;
21: Map the percentage to the credit fuzzy set;
22: return the credit value to the fuzzy inference engine
```

Algorithm 2 Rate limiting

```
Output: $Rate$
1:  if packet.type != TCP then then
2:      return;
3:  end if
4:  if packet.flags = SYN then
5:      Install flow for forwarding direction;
6:      $Counter++$;
7:  end if
8:  if packet.flags = SYNACK then
9:      Install flow for backward direction;
10: end if
11: if request not in working set then
12:     $delayqueue$ := request;
13:     Move the new request from the delay queue to the working set;
14: end if
15: $percentage$:= The percentage of connection attempts;
16: Map the percentage to the Rate fuzzy set;
17:  return the rate value to the fuzzy inference engine;
```

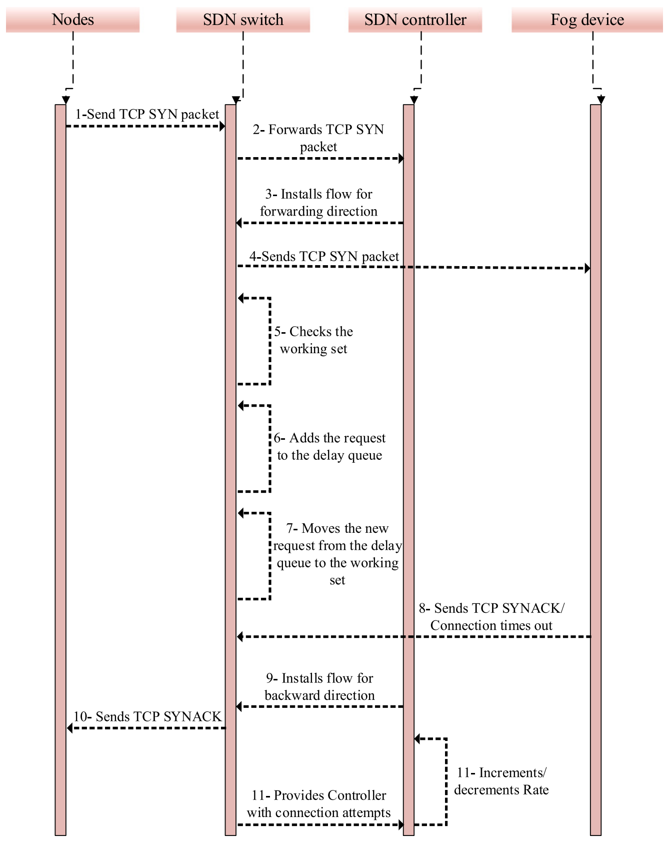

Nodes
SDN switch
SDN controller
Fog device
1-Send TCP SYN packet
2- Forwards TCP SYN packet
3- Installs flow for forwarding direction
4-Sends TCP SYN packet
5- Checks the working set
6- Adds the request to the delay queue
7- Moves the new request from the delay queue to the working set
8- Sends TCP SYNACK/ Connection times out
9- Installs flow for backward direction
10- Sends TCP SYNACK
11- Increments/ decrements Rate
11- Provides Controller with connection attempts

Fig. 4. The sequence diagram of Rate limiting algorithm.

3.4. FUPE: proposed security-aware scheduler

As aforementioned, the role of the assigner module is to find the most proper fog device for processing a requested task that has been previously validated by *request validator* module. The designated fog device to the application's tasks should be secure and also has a suitable computing capacity. In this work, we assume both the users' devices and compromised fog devices can perform DDOS attacks against other fog devices. Moreover, we assume the behavior of a benign node can change to become an attacker. We apply TRW-CB in FUPE due to its continuous monitoring ability to detect malicious network traffic. FUPE can instantly detect and block anomalies in network traffic when malicious activity happens. To achieve this, FUPE uses a multi-objective PSO algorithm to increase the efficiency together with security. Multi-objective optimization solutions [62] fall into two main categories: *(i)* those, defining the objective functions separately and combine them by using the weighted sum of them; *(ii)* those, using a single objective function and aggregate the distinct objectives into one synthetic objective by using Pareto Optimality. In FUPE, we adopt the former strategy.

Fuzzy Based Fitness Function. As Fuzzy logic is widely used to consider multiple criteria simultaneously, we use it in both fitness functions. Each of the fitness functions determines the rank of the two objectives. Our presented approach has two objectives as detailed in the following.

First Objective (Security): For considering security objective, we use *TRW-CB* and *Rate Limiting* techniques. The cloud gateway gathers the information regarding the amount of exchanged traffic between fog devices/users' devices and the number of successful connections between fog devices/users' devices from all fog gateways. Then, it calculates the *Credit* and the *Rate* values for each fog device/user's device. Finally, it sends these values to fog gateways. We use them as the input parameters of the fuzzy based fitness function for security objective. For instance, *Low Credit* and *High Rate* indicate a malicious fog device/user's device, respectively. According to previous researches such as [59], an anomaly occurs when the value of Credit and Rate exceeds a predefined threshold. This work considers High Credit and High Rate as the malicious activity. In other work [61], the authors use Low Credit and High Rate to indicate malicious nodes. In this paper, we consider Low Credit and High Rate as the anomaly behavior as well. Defining the fuzzy sets and fuzzy rules are based on using either learning methods [63–65] or former experience /predefined assumptions [36,55,61]. We define the range of Credit and Rate fuzzy sets based on the predefined assumptions (i.e., the fuzzy sets and fuzzy rules set are predefined and static). A suitable approach to define the fuzzy sets is in such a way that the end-point of the first fuzzy set is the beginning point of the third fuzzy set; the second fuzzy set has overlap with both the other fuzzy sets. The benefit of this approach is that it increases both of the fuzzy features of the fuzzy sets and the overlapping between the fuzzy sets.

Second Objective (efficiency): It is illustrated in the literature that simultaneous consideration of the computing features of the resources and application's tasks CPU need, is suitable for task scheduling [40, 55,66,67]. So, we use application's tasks CPU need, and the features of fog devices (i.e., available processing capacity, available RAM size, and available fog-device link bandwidth) as the input parameters of the fuzzy based fitness function for efficiency objective. FUPE implements a task merging mechanism where the tasks of the same application are assigned to the same fog device. Besides, we use the total amount of tasks CPU needs of an application as the *application's tasks CPU need* parameter.

Mamdani fuzzy inference engine is one of the most common fuzzy inference engines which uses fuzzy sets and fuzzy rules. These rules are based on the former experience or predefined assumptions. We define three overlapping fuzzy sets that make the input values possibly locate in several sets simultaneously. Accordingly, each of the input values belongs to several fuzzy sets with different membership degrees. For instance, consider the fuzzy sets in Figs. 3 and 4. Low and Medium fuzzy sets are overlapped, but there is no overlap between Low and High fuzzy sets. In this way, an input value which is in the Low fuzzy set interval is considered as both Low and Medium at the same time with different degrees of membership (confidence). Besides, if *B* is the degree of membership of the input value in the Low fuzzy set, then its membership in the Medium fuzzy set is *1-B*. For the sake of clarity, suppose Credit/Rate fuzzy sets in Fig. 3. If the input value is 0.2, then the degree of membership for Low and Medium fuzzy sets are 0.8 and 0.2, respectively. In this example, *B* and *1-B* are 0.8 and 0.2, respectively.

As a consequence, each of the input values can trigger several fuzzy rules (i.e. rule firing). Then the fuzzy inference engine aggregates rules and combines the fuzzy sets that represent the outputs of each fuzzy rule into a single fuzzy set. Finally, it interprets the membership degrees of the fuzzy set into a numeric non-fuzzy value which is the output of each of the objectives. The obtained values are the output of the fitness functions (i.e. security and efficiency objective). The fuzzy rules are indicated in Table 3 for the first objective and Table 4 for the second objective. Figs. 5 and 6 show the fuzzy sets for the two objectives as well. The rows in Tables 3 and 4 are the fuzzy rules. The rules consist of the fuzzy sets, and the values of the fuzzy sets are indicated in Figs. 3 and 4. The cells Low, Medium, High, Safe, Potential, and Attack are the fuzzy sets that are used in the fuzzy rules. For instance, the third row in Table 3 means: If Credit is Low and Rate is High then Result is Attack. As it is shown in Fig. 3, the values of Credit Low, Rate High, and Result Attack fuzzy sets are 0.0 to 0.5, 0.5 to 1, and 0.0 to 0.5, respectively. To take another example, the first rule in Table 4 means: If Task Length is Low and Bandwidth is Low and CPU is Low and Ram (memory) is Low then Result is Inappropriate. Fig. 4 shows the values of the Task length Low, Bandwidth low, CPU low, Memory low, and

Table 3
Fuzzy rules for security objective.

Credit	Rate	Result
Low	Low	Potential
Low	Medium	Attack
Low	High	Attack
Medium	Low	Safe
Medium	Medium	Potential
Medium	High	Attack
High	Low	Safe
High	Medium	Safe
High	High	Potential

Inappropriate fuzzy sets that are 500 to 2750, 10 000 to 15 000, 10 000 to 25 000, 0 to 12 300, and 0.0 to 0.5, respectively. Result fuzzy set is the output of each of the rules. Mamdani Fuzzy inference engine utilizes the mentioned fuzzy sets in the aggregation method to obtain the output of each of the objectives.

MOPSO based IoT–fog security driven task scheduling algorithm. The PSO algorithm is used to find an optimal solution for optimization problems which is based on the behavior of a flock of birds to find their way to a specific destination. To start this algorithm, a population of particles is generated randomly in which their positions are considered as a potential solution. Each particle has a position vector and a velocity vector which determine the direction of movements in search space. In the PSO algorithm, the particles are initialized randomly. In this work, we consider a request as a particle and the state of the particle as the particle's position. We also consider *pbest* and *gbest* as the best position of a job request attains among the job request list, and the best position of job request as job request attains, respectively. The *gbest* is the global optimal state (i.e., the best fog device for each request). For example, if there are 8 fog devices and 3 requests in the fog region, FUPE assigns the fog device to the requests based on the *gbest* value for that request. A possible solution could resemble $\{2, 3, 2, 1, 2, 3, 2, 1\}$. For instance, the first element of this combination means the first fog device is the best optimal venue for the second request, and the second fog device is the best optimal venue for the third request.

In each iteration, fuzzy-based fitness functions are used to assess each particle's values so as to replace the position of particle with *pbest* variable. FUPE rejects a new solution if its *pbest* value is less than the current solution. In fact, it improves the fitness value of the applications' tasks in every iteration. *pbest* of all particles are used to update the *Gbest*. The current *pbest* is compared to *Gbest*, and if its satisfaction is more than *Gbest*, it will be replaced. The final result is the best *pbest* value (i.e. *gbest*). General PSO is defines as follows:

$$X^i_{k+1} = X^i_k + V^i_{k+1} \tag{1}$$

In this equation, i, X, and V are the particles, the position of the particles, and the velocity of the particles, respectively. The formula for updating velocity vector is:

$$V^{i+1}_k = W_k V^i_k + C_1 r_1 (Pbest^i_k - X^i_k) + C_2 r_2 (Gbest - X^k_i) \tag{2}$$

where C_1 and C_2 are the acceleration coefficient which is considered as learning factors, r_1 and r_2 are used to control the randomness in the movements of particles in the search space, and W is inertia weight which indicates the balance between local and global search. This parameter determines the repeat rate for finding the best solution. To aggregate the outcome of the two objectives into one synthetic objective to obtain a single proper answer, we use Eq. (3).

$$Aggregation = ((Security * \alpha) + (Efficiency * \beta)) * (-1) \tag{3}$$

where $Security$ and $Efficiency$ are the first and second objective outcomes respectively. We assign priority to the objectives by α and β which are predefined weights. We set these weights in a way that the sum of them equals 1. In this work, we assume the priority of the two objectives is the same. So, we assign 0.5 to both α and β. PSO is a minimization problem and FUPE is a maximization problem. As the goal of the PSO algorithm is to minimize the objective and we aim to maximize the outcome of FUPE, we multiply the outcome by −1. It is worth highlighting that we use the two fuzzy-based fitness functions to update the values of *security* and *efficiency* variables in Eq. (3) to calculate *pbest*. Algorithm 3 indicates the proposed algorithm which leverages the benefits of the MOPSO algorithm in order to find an optimal solution regarding the two aforementioned objective functions.

Algorithm 3 FUPE-Proposed Algorithm

```
Output: $gbest$
1:  for $i = 1$ $toM = Iterationnumber$ do
    Initialize $P[i]$ randomly; $P$ is population of particles
2:    Initialize $v[i] = 0$;
    $v$=speed of each particle
3:    $F = ComputeFitness(p[i])$;
4:    $gbest$ = best particle found in $F$;
5:    for $i = 1$ $toM$ do
6:        $pbest[i] = F$;
7:    end for
8:    repeat
9:        for $i = 1$ $toM$ do
10:           Set $W$, $C_1 >= 0$ and $C_2 >= 0$;
11:           $v[i] = W \times v[i] = C_1 \times rand1(pbest[i] - p[i] + C_2 \times rand2(gbest[i] - P[i])$;
12:           update speed of each particle
13:           $P[i] = P[i] + v[i]$;
14:           if a particle goes outside the predefined hypercube then
15:               it is reintegrated to its boundaries;
16:           end if
17:           $F = Computefitness(p[i])$;
18:           if new population is better then
19:               $pbest[i] = F$;
20:           end if
21:           $gbest$=Update it by the best particle found in $p[i]$;
22:       end for
23:   until stopping criterion is satisfied
24: end for
25: return $gbest$
```

Alg. 3 Description. In Alg. 3, first, FUPE defines the initial position and velocity vector for each particle (tasks) in the search area randomly (line 1). We save the current solution in *pbest* (line 2). In lines 3 to 4, FUPE computes the two fitness functions for the particles. In lines 5 to 7, after assigning the fitness value for all of the particles, FUPE compares them, and if the fitness value of the current solution is higher than *pbest*, it replaces it with the *pbest*. In lines 9 to 23, after each iteration, FUPE updates *gbest* value by comparing the *pbest* value of the current iteration to the value which is stored in the *gbest*. FPFTS calculates the request's position and request's velocity by using Eqs. (1) and (2). In each iteration, FUPE calculates *pbest* for the particles with the aid of Eq. (3). FUPE repeats these steps to reach the maximum number of iterations. Finally, FUPE sets the best *pbest* value as the *gbest* value. *Gbest* is the output of this algorithm which indicates the most proper fog device for the application's tasks. Each fog device has a numerical value as the *gbest* for the incoming job tasks. FUPE assigns a fog device which has the greatest *gbest* value for the job tasks.

Computational Complexity of FUPE: Computational complexity shows the number of resources that an algorithm needs to be executed. We make reference to the work by Arora et al. [68] in which the authors set out a clear mathematical illustration of computational complexity. In FUPE, as we use computing features of resources and tasks and the parameters of the TRW-CB and Rate Limiting simultaneously, we consider the two objectives' computational complexity and then multiply

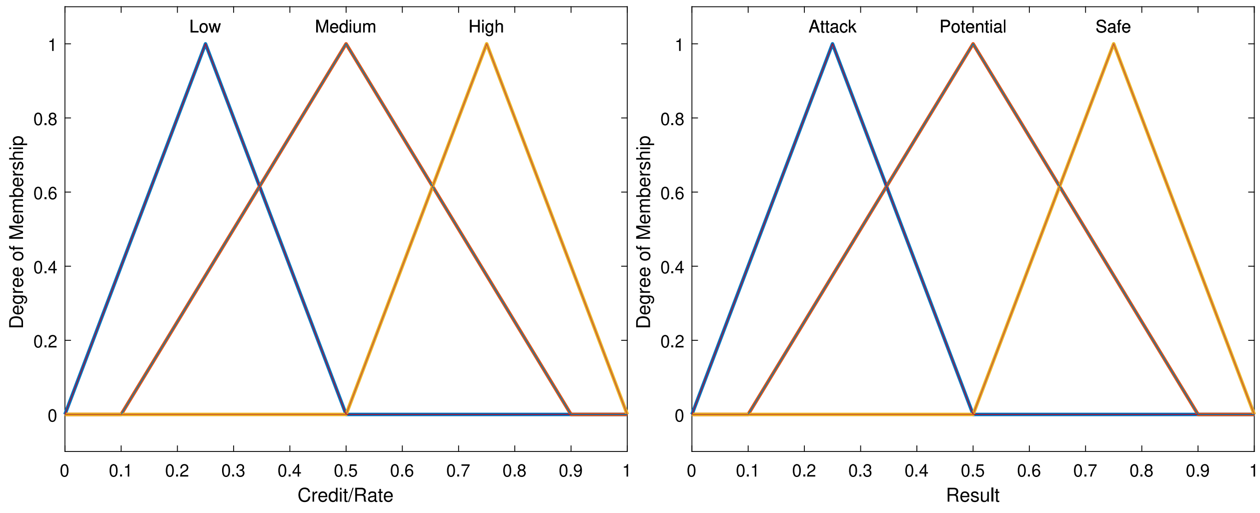


Fig. 5. Fuzzy sets for security objective.

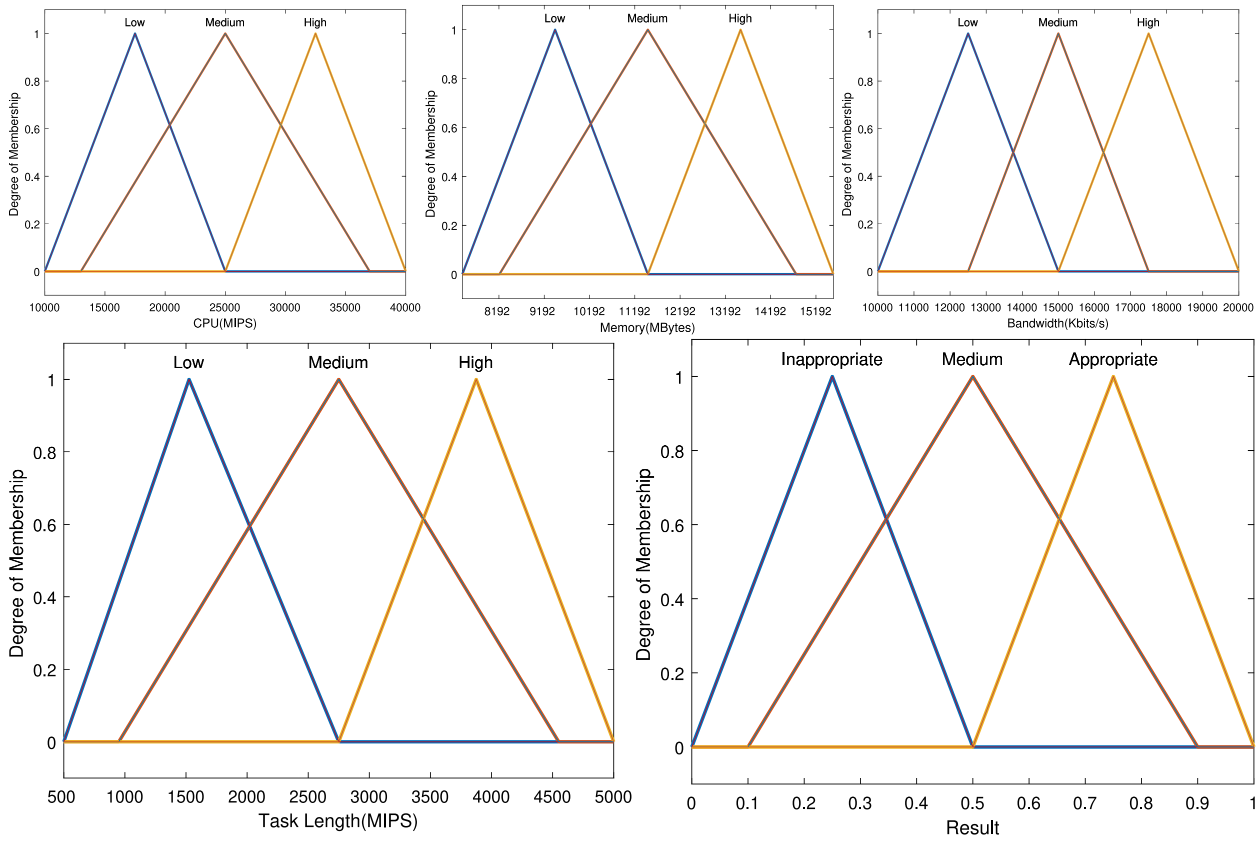


Fig. 6. Fuzzy sets for efficiency objective.

them as follow: Firstly, the computational complexity of efficiency objective is in the order of $\mathcal{O}(I \times R \times P)$ where I is the number of iterations, R is the number of resources (i.e. fog devices), and P is the number of particles. Besides, both of the Rate Limiting and TRW-CB techniques take the same amount of time to obtain the result irrespective of the input size. They take the exact time to process one item as well as for example ten items. So, they have a constant growth rate run time. As we apply both TRW-CB and Rate Limiting techniques at the same time for the same nodes, we can say the computational complexity of security objective is in order of $\mathcal{O}(1)$. Finally, FUPE uses Mamdani fuzzy inference engine, so the complexity of the fitness function is in order of $\mathcal{O}(1)$. As Mamdani fuzzy inference engine searches, fires and aggregates the adequate rules in a parallel manner, the complexity of it is in order of $\mathcal{O}(1)$ [69]. As a result, the total computational complexity of the FUPE is in order of $\mathcal{O}((I \times R \times P))$.

Space Complexity of FUPE: For calculating the space complexity of FUPE, we take the spaces that FUPE needs to be executed into consideration. As the instances of FUPE are located in fog gateways, the number of fog regions plays a central role in the algorithm space requirements. So to obtain the space complexity of FUPE, we use the number of fog regions and we name this parameter S. As a result, the total space complexity of the FUPE is in order of $\mathcal{O}(S \times (I \times R \times P))$.

An example of scenario for FUPE complexity: As FUPE uses MOPSO algorithm, more particles, fog devices, and iterations increase

Table 4
Fuzzy rules for efficiency Objective. TL = Task length. C/R = CPU/RAM. I= Inappropriate. A = Appropriate.

TL = Low , BW = Low				TL = Medium , BW = Low				TL = High , BW = Low			
H	M	L	C/R	H	M	L	C/R	H	M	L	C/R
M	I	I	L	I	I	I	L	I	I	I	L
M	M	I	M	M	I	I	M	I	I	I	M
A	M	M	H	M	M	I	H	M	I	I	H
TL = Low , BW = Medium				TL = Medium , BW = Medium				TL = High , BW = Medium			
H	M	L	C/R	H	M	L	C/R	H	M	L	C/R
A	M	I	L	M	I	I	L	I	I	I	L
A	A	A	M	A	M	I	M	M	I	I	M
A	A	A	H	A	A	M	H	A	M	I	H
TL = Low , BW = High				TL = Medium , BW = High				TL = High , BW = High			
H	M	L	C/R	H	M	L	C/R	H	M	L	C/R
A	A	M	L	M	M	I	L	M	I	I	L
A	A	A	M	A	M	M	M	M	M	I	M
A	A	A	H	A	A	M	H	A	M	M	H

complexity. Defining the number of iterations depends on the environment. For instance, we can define the number of fog devices or applications as the iteration number. Suppose a scenario in which the number of particles is 10, the number of resources is 5, and we define the number of particles as the iteration. Moreover, we have 2 fog regions in the network. In this way, the Computational Complexity is 500 $(10 \times 5 \times 10)$, and the space complexity is 1000 $(2 \times 10 \times 5 \times 10)$.

4. Performance evaluation

In this section, we conduct a comprehensive simulation study and analyze the results to compare the performance of FUPE in a fog environment to a single-objective PSO algorithm [70,71] and a Genetic algorithm [41] that do not take security into account. This comparison allows to show the overhead of security mechanisms in FUPE.

For the evaluation, for each scenario, we run the algorithms ten times. Application scheduling in distributed systems is an NP-complete problem [72]. Evolutionary optimization algorithms are proper approaches to find the optimized solutions that generate several solutions. The final solution is the best answer among the generated solutions. So, in FUPE we have a set of applications, and FUPE runs while an adequate amount of applications are on the scheduling list. We performed the simulation study in MATLAB R2018a [73]. Besides, we used Xfuzzy 3.5, which is a fuzzy logic design tool to implement fuzzy-inference-based systems [74]. Since Matlab does not have inbuilt features to support the OpenFlow protocol directly, we extended the Matlab classes to add the OpenFlow protocol specification. We added routing or decision making functionality to Matlab classes. In detail, we defined a node (SDN switch) with a data structure similar to the flow tables. Then, another node (SDN controller), fills the flow tables.

iFogSim [75] is a widely accepted IoT–Fog simulator which we also used to implement our former work, FPFTS [40]. This simulator supports edge processing, edge communication, IoT devices, and also cloud processing. The big disadvantage of this tool is the lack of SDN support, SDN-WAN support, network communication, also network protocols and the missing possibility to implement the proposed security objective (e.g., 3-way handshaking). However, standard network simulators (e.g., ns [76], and OMNeT [77]) or SDN-aware network simulators (e.g., mininet [78], and SDN-Sim [79]) are not suitable to implement the presented approach at all. For instance, NS-3 and OMNeT++ support network communication and network protocols. Although they provide the users with proper programming capabilities, SDN and SDN-WAN features, edge processing, edge communication, also fog computing facilities and IoT devices are not natively supported by them. To take another example, Mininet quantifies SDN performance within different network structures and routing protocols. It only supports SDN-WAN facilities, network communication, and network protocols. The focus of SDN-Sim is to deploy SDN-based policies, such as channel modeling, traffic shaping, and QoS demands. The disadvantage of these tools is the lack of edge processing, edge communication, also fog computing facilities and IoT devices.

Table 5
Configuration of fog devices.

Configuration	Config 1	Config 2
RAM	8 GB	16 GB
CPU	5000 MIPS	10 000 MIPS
Core in CPU	2	4
Bandwidth	10 Mbps	20 Mbps

FUPE does not consider the actual medium and stack of protocols. Hence, we do not change or modify the TCP/IP layers to observe the effect of FUPE on them. We did not use actual network parameters in Matlab. We can define propagation delay as distance, and not consider Queue delay, transfer delay, and processing delay. Attacks can occur in both wired or wireless ways. We are concentrating on the behavior of node requests. It makes no difference whether they are transmitted via a wired or wireless link. We basically study the behavior of malicious nodes. In any network environment setting, we expect the malicious nodes to behave differently than the benign nodes. Our method catches that abnormality in task scheduling phase. Therefore, using any network environment for packet request generation does not show any significant difference in results than those reported. Finally, it is illustrated in the work [10] that about 15% of the resource management approaches in fog computing have utilized Matlab simulator. There are also several works in SDN networks and DDOS attack detection systems that are implemented by Matlab [80–93]. As there is no corresponding simulator that covers the minutiae of FUPE, we simplify it into a trade-off problem, and use MATLAB for evaluation. We implement a scalable IoT–fog framework in MATLAB in order to simulate FUPE as much as possible in a real scenario. This framework is available along with the paper, and we can use it in other similar works.[1] We can generate other scenarios and reproduce the framework for other SDN based IoT–fog works.

4.1. Simulation setup

To simulate fog devices, we assume each fog device's configuration can be one of the items specified in Table 5. Upon creating fog devices, one of these configurations is selected at random. We considered the same random generated scenarios for each algorithm under test.

[1] https://github.com/mshojafar/sourcecodes/tree/master/Saeed2021FUPE_Sourcecode.

Table 6
Task requirements.

Requirements	REQ 1	REQ 2	REQ 3	REQ 4
RAM	256 MB	512 MB	512 MB	1 GB
CPU	500 MIPS	1000 MIPS	2000 MIPS	5000 MIPS
Bandwidth	5 Mbps	10 Mbps	10 Mbps	10 Mbps

To simulate task requirements, we assume that each separate task needs resources in terms of RAM, CPU, and Bandwidth. Once a task is generated, it randomly selects one of the four requirement configurations specified in Table 6.

4.1.1. Simulation metrics

To evaluate the proposed solution against the selected baselines we consider the set of metrics in the following:

- *Response time:* the amount of time elapsed between the time a task is sent by a user to fog device and the time the related result is delivered to the user. It is worth noting that, this time includes network delay, processing delay, and task scheduling mechanism delay. We focus on both average and maximum performance figures for this metric.
- *Confidence interval (CI):* denotes a range of values that is determined by the margin of error. A confidence interval for the results depends on sampling the distribution of a corresponding estimator. The goal of this metric is to indicate the variability of the results. For the performance evaluation, we use confidence intervals for plots calculated with 95% confidence level. Focusing on the confidence interval, we refer the readers to Ref. [94].
- *Network Utilization:* It provides a quantification for the data transmitted in the network links between network nodes (i.e. how much link bandwidth is used) during a time interval. We use Eq. (4) to calculate this metric. In this equation, the length of data that are encapsulated in each task is calculated in *KByte*. Moreover, Interval is set to 1000 ms (i.e. we calculate network utilization every second).

$$NetworkUtilization = (\sum_{x=1}^{Requests} Task\_length)/Interval \quad (4)$$

4.1.2. Simulation scenarios

To evaluate all aspects of FUPE in terms of performance and security, we have considered various scenarios with different number of attack rates, different number of fog devices, and different number of jobs. There are listed below

- *Scenario-1:* In this scenario, the number of jobs is 60 per second and the number of fog devices is 10. The attack rate in this scenario varies from 0% to 50% (e.g. 0 to 50 percent of requests are malicious and TCP SYN attack, while the other requests are benign and valid requests). The aim of this scenario is to investigate whether FUPE can detect attacks and does not assign tasks to malicious fog devices, and also does not process the task coming from attacker users. Moreover, this scenario illustrates the overhead of the FUPE itself.
- *Scenario-2:* In this scenario, the number of fog devices varies from 5 to 25, and the number of jobs is 60 per second. The attack rate is fix and equal to 30%. In fact, the goal of this scenario is to investigate the performance of FUPE when increasing the fog devices in the presence of a moderate attack.
- *Scenario-3:* In this scenario, the number of fog devices is 60, and the attack rate is 30%. The number of jobs varies from 20 to 100 per second. The goal of this scenario is to investigate the scalability of FUPE when increasing the number of jobs in the presence of a moderate attack.

In all simulation scenarios, we assume one cloud gateway and two fog gateways.

4.2. Experimental results

In this section, we discuss and analyze the simulation results for the FUPE and other approaches in different scenarios. First of all, we investigate the results for *Scenario-1*. Figs. 7a and 7b show the average and maximum response time against different attack rates for the methods, respectively. In these two figures, it is obvious that by increasing the attack rate, the response time grows linearly for Genetic and PSO algorithms. The reason is that in these two methods, there is no mechanism to detect and mitigate the effect of the attacks. Therefore, the response time considerably grows. On the other hand, in FUPE, increasing attack rate does not affect the response time. This is because of considering *Credit* and *Rate* for each request in fog gateways. Applying these security parameters leads to FUPE significantly outperforms other approaches and improve better response time for users' requests. In addition to security consideration, as aforementioned, FUPE uses a multi-objective model in order to make efficient use of the network resources by assigning tasks to suitable fog devices. This behavior also leads to less response time.

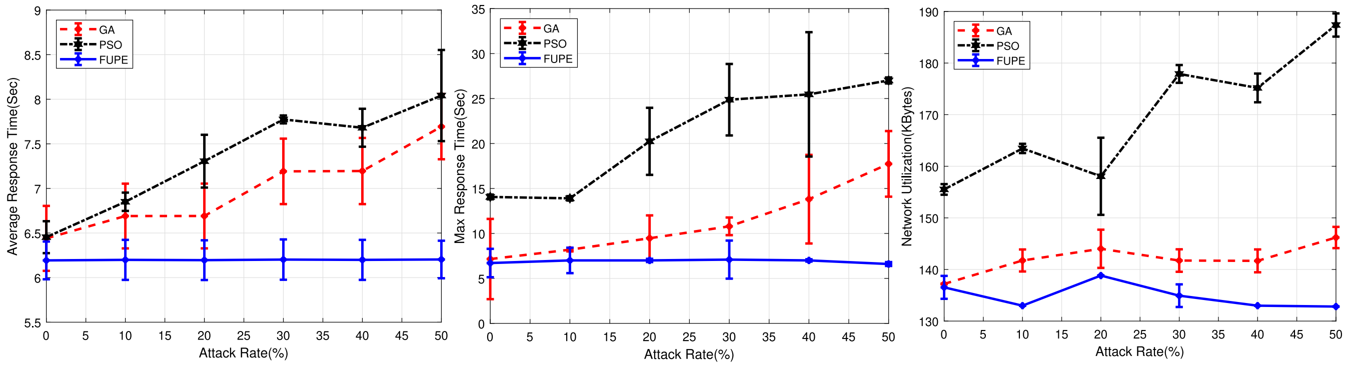
In addition to these achievements, another important benefit of using FUPE is to reduce resource network bandwidth exhaustion. Fig. 7c proves that using security parameters in FUPE leads to a significant prohibition of the adversary effect of different rates of TCP SYN flooding attack. In the other two methods, there is no security mechanism to prevent network utilization by malicious behaviors, and as a result the network utilization has been linearly increased.

Fig. 7 also indicates the CI for plots as indicators over the results. As all the methods use randomness features, they have almost the same tolerance in the results. However, the tolerance of the results for FUPE is a bit less. More specifically, it is more obvious in the 7c. The reason behind this is FUPE uses security parameters in scheduling the tasks to fog devices. Hence, it does not assign the tasks to malicious fog devices. Consequently, the volume of assigned tasks for fog devices is not equal. The quota of compromised fog devices is zero and the quota of non-compromised fog devices is high. This behavior causes more tolerance in the results.

Figs. 8a and 8b illustrate the average and maximum response time with different number of fog devices in Scenario-2, respectively. From these two figures, it can be concluded that, in the presence of a 30% attack rate, increasing the number of fog devices (i.e. computational capacity) decreases the response time in Genetic and PSO algorithms. This result is obvious because the computational capacity of the network has been increased. Moreover, in comparison with 7b, the maximum response time for FUPE also has been decreased. These two figures prove that, unlike Genetic and PSO algorithms, FUPE has not undergone significant changes in terms of response time. The main conclusion from this result is that since FUPE uses the security mechanism together with the multi-objective model to assign tasks optimally, it can provide reasonable response time even in the presence of attacks, and there is no need to add extra fog devices. In nutshell, FUPE utilizes network and computational resources optimally and also prohibits TCP SYN flooding attack to exhaust the resources.

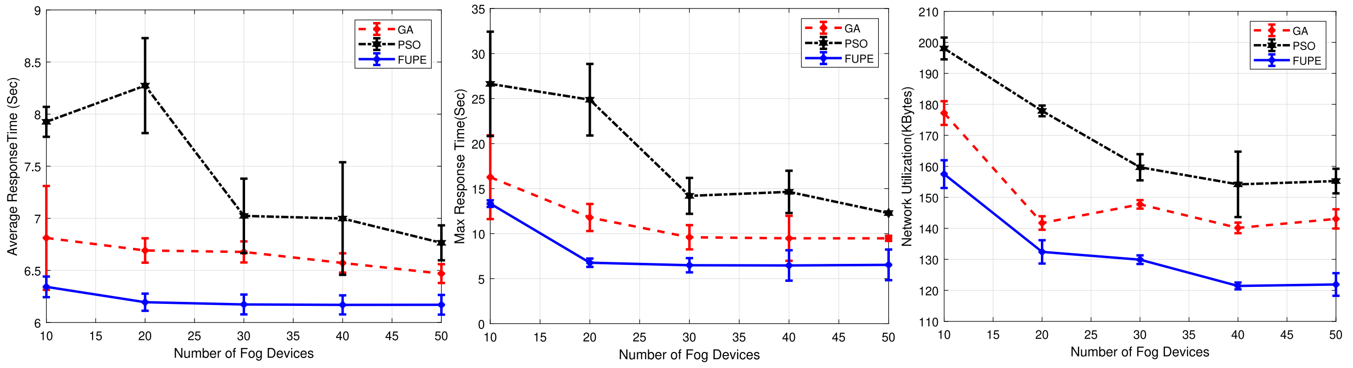
Fig. 8c shows that as the number of fog devices increases the network utilization in all three methods decreases. The reason behind this is that, increasing the size of the network while the load of the network is constant, leads to low network utilization. It can be concluded from Fig. 8c that, because of applying performance parameters in task scheduling in FUPE, it assigns the task in a fair manner in compared to the other methods.

Fig. 8 also shows that the Tolerance of FUPE results in task assignment for FUPE improves when the number of fog devices increases. In fact, using a large number of fog devices leads to spreading tasks to more devices, and as a result the difference between the upper and lower values for CI decreases. The reason behind this is FUPE uses fog devices features and jobs' requirements in the efficiency objective.



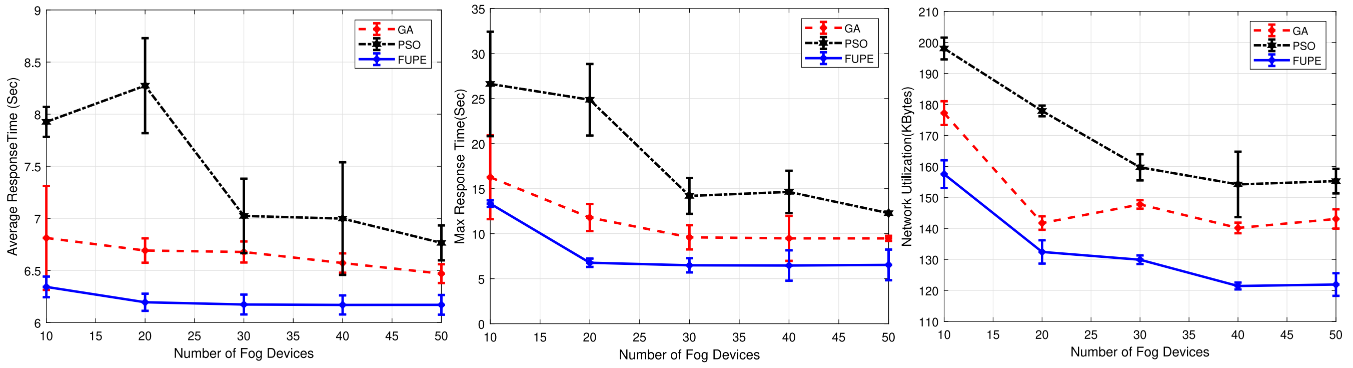
(a) Average response time. (b) Maximum response time. (c) Network utilization.

Fig. 7. The comparison results for the first scenario in the presence of different attack rates in (%).



(a) Average response time. (b) Maximum response time. (c) Network utilization.

Fig. 8. The comparison results for the second scenario in the presence of different number of fog devices.



(a) Average response time. (b) Maximum response time. (c) Network utilization.

Fig. 9. The comparison results for the third scenario in the presence of different number of jobs.

Figs. 9a and 9b depict average and maximum response time with different number of jobs in Scenario-3, respectively. From these two figures, it can be concluded that, in the presence of a 30% attack rate and a fixed number of fog devices (10 devices), increasing the number of jobs considerably increases the response time both Genetic and PSO algorithms. This increase is also true for FUPE. But, in FUPE, the growth of response time is significantly less than the other two methods. Once more, we note that due to applying the security mechanism together with the multi-objective model in FUPE, assigning the tasks is performed optimally. Therefore, network and computational resources have been used efficiently.

Fig. 9c illustrates the impact of increasing the load (number of jobs) in network utilization. As can be seen in this figure, it is obvious that the network utilization is increased by holding the network resources constant. But, this figure clearly shows that FUPE considerably achieves better results in terms of network utilization. This improvement comes from two mechanisms: First, using performance parameters such as CPU, RAM, and network bandwidth in choosing the optimal fog device. Second, FUPE considers the applications' CPU needs. Moreover, it considers the security conditions of requester and fog devices and prohibits malicious requests to be processed. It also blocks future requests from attacker nodes. Unlike FUPE, the other two methods have not any performance and security mechanism to mitigate TCP SYN flooding attacks.

Fig. 9 also illustrates that the results tolerance in task assignment for the approaches. In FUPE, assignment of tasks to compromised fog

Table 7
Accuracy evaluation.

	TN	TP	FN	FP	Accuracy	Precision	Recall
Value	688	294	6	12	0.9820	0.9608	0.9800

devices is prohibited and the tasks are assigned to other fog devices. This behavior separates fog devices into two groups and only non-compromised fog devices can process the tasks. As a result, the FUPE results tolerance is less than the other two approaches. AS Genetic algorithm considers resource stability, it improves the stability of the task execution. As the result, it has a better results tolerance compared to the PSO algorithm.

In this scenario, we want to show the overhead to execute the proposed algorithm. We indicate the required CPU and RAM compared to the other solutions. In this paper, we use an approximation algorithm to mimic (approximate) the behavior of the malicious nodes, and also catch them in the scheduling phase. The approximation method gathers information based on the mode status and system capabilities, so we need to access the system characteristics. Consequently, it can take time to gather information unless we ask up it in the model generations. There is no specific hardware (RAM, CPU) requirement for FUPE, and It works for any system without any problem. However, it needs more processing capacity compared to the other solutions. Figs. 10a and 10b indicate the required processing capacities for the algorithms. Both Genetic and PSO methods do not have security assurance. For FUPE, we have some time consumption for the security part (i.e., the security cost). It includes the number of connection requests (i.e. TCP SYN packets) by fog devices/users' devices, the number of TCP SYNACK packets, and the number of flows installed in the fog gateways. Moreover, as MATLAB consumes much CPU and RAM for fuzzy functions, the overhead of FUPE is more than the other two methods.

According to Figs. 7 to 9, it can be concluded that, after FUPE, Genetic algorithm achieves better results in comparison with pure PSO. This is because of, in Genetic approach, the objective function is optimized to consider both latency and stability to assign tasks to the fog devices. Furthermore, Figs. 7 to 9 obviously shows that the only side effect of FUPE is that, in the presence of attacks, it cannot equally assign tasks between fog devices. In fact, this issue is not the weakness of FUPE, because it prohibits compromised fog devices to process tasks. As a result, the burden of workload in compromised fog devices is zero while in non-compromised nodes is high.

Finally, we present the accuracy evaluation of the presented approach. For this evaluation, there are 700 benign requests, and 300 malicious requests (attack). As Genetic and PSO algorithms do not have attack defence mechanisms, for this evaluation, we only evaluate FUPE in terms of accuracy, precision, and recall.

$$Accuracy = (TP + TN)/(TP + TN + FP + FN) \tag{5}$$

$$Precision = TP/(TP + FP) \tag{6}$$

$$Recall = TP/(TP + FN) \tag{7}$$

where TP denotes true positive, and FN denotes false negative, TN denotes true negatives, and FP denotes false positive rates, respectively. Based on Eqs. (5), (6), and (7), *Accuracy*, *Precision*, and *Recall* are 0.9820, 0.9608, and 0.9800, respectively. Table 7 indicates the accuracy detection of FUPE.

5. Discussion

The scheduling problem of users' applications is the major challenging research topic in IoT networks. Due to the distributed feature of IoT–fog resources, the computational resources are more susceptible to come under attack. FUPE investigates the security-driven scheduling policy. As it is illustrated in Section 4, multi-objective PSO method is a proper way to achieve a tradeoff between security and efficiency. We found Mamdani fuzzy inference system, a method to strike a balance between distinctive parameters in the *two* MOPSO fitness functions. Because of randomness feature of PSO, FUPE has more uneven load distribution among fog nodes. However, from security point of view, this indicates that FUPE has the ability to detect malicious fog devices. Although in this work we take TCP SYN flood attack into consideration, we argue that FUPE can be extended to tackle the other DDOS flooding attacks such as UDP flood and ICMP flood attacks.

FUPE can prioritize the distinct objectives based on the organizational policy. It means it can increase/decrease the objectives' priorities. For example, if the priorities of the objectives are the same, we multiply the fitness function outcomes by the same weight value (see Eq. (3) in Section 3.4). Thus, FUPE chooses a computational resource based on the same trustworthiness and computational capability. In another example, suppose the organizational policy is to adjust either the security or efficiency of the computational resources. We can simply increase or decrease the objectives' weight values which are a number between 0 and 1. Suppose the organizational management model is to increase the security level. Thus, we simply adjust the security objective's weight value.

In both TRW-CB and Rate Limiting techniques, FUPE uses the request sender's ID as their identity. This ID can be the IP address or Mac address of the request sender. To mask their identity in a TCP SYN flood attack, compromised fog devices/malicious users' devices use IP spoofing/MAC spoofing. To counteract the aforementioned attacks, we can include IP spoofing and MAC spoofing detector modules in SDN switches. For the former, the IP spoofing detector module includes a table that stores the IP and MAC addresses of fog devices and users' devices. If the detector module detects an IP address associated with more than one MAC address, the fog device/end-user device is considered an attacker [95]. Some works consider the sequence number field in the link-layer header of each flow for the latter. The MAC spoofing detector module computes the difference between the current frame's sequence number and the last frame received from the same fog device/end-device user's. We can use either a threshold-based [96] or a fuzzy-based approach [97] to make decisions.

Improving accuracy metrics is a critical issue in intrusion detection and prevention systems. The work [59] illustrates that by tuning the threshold parameters of TRW-CB and Rate Limiting, they have proper performance in terms of accuracy evaluation metrics. The performance of these algorithms relies on the threshold settings. As FUPE makes use of fuzzy logic for security objective, we eliminate the problem of threshold approaches (i.e., choosing an adequate threshold based on the aim of the approach). On the other hand, it relies on applying an appropriate fuzzy rules set for proper performance. If we use appropriate fuzzy rules in the Mamdani fuzzy rules set, FUPE can provide better performance in terms of accuracy metrics for obtaining security objective. As FUPE utilizes fuzzy logic for obtaining security objective, the results are perceived based on assumptions for defining fuzzy rules set. So, the accuracy of FUPE DDOS attack detection is based on these assumptions that are tunable. It must be customized and fine-tuned to get the desired accuracy.

FUPE is the first security-driven task scheduler that integrates DDoS attack detection techniques in a scheduling algorithm in IoT–fog networks. Response time and network utilization are the two most critical issues in IoT–fog networks which affects on users' satisfaction. To reduce these important metrics we make use of fog computing in IoT networks. That is the reason why they are the two common metrics for evaluating IoT–fog scheduling algorithms. To evaluate the security aspect of FUPE, we varied the attack rates to obtain the mentioned metrics. For evaluating Trust/security aware big data task scheduling approaches, it is common to evaluate the methods by considering the efficiency metrics and varying a security variable. For instance, in the work [50], response time is calculated by varying the number of

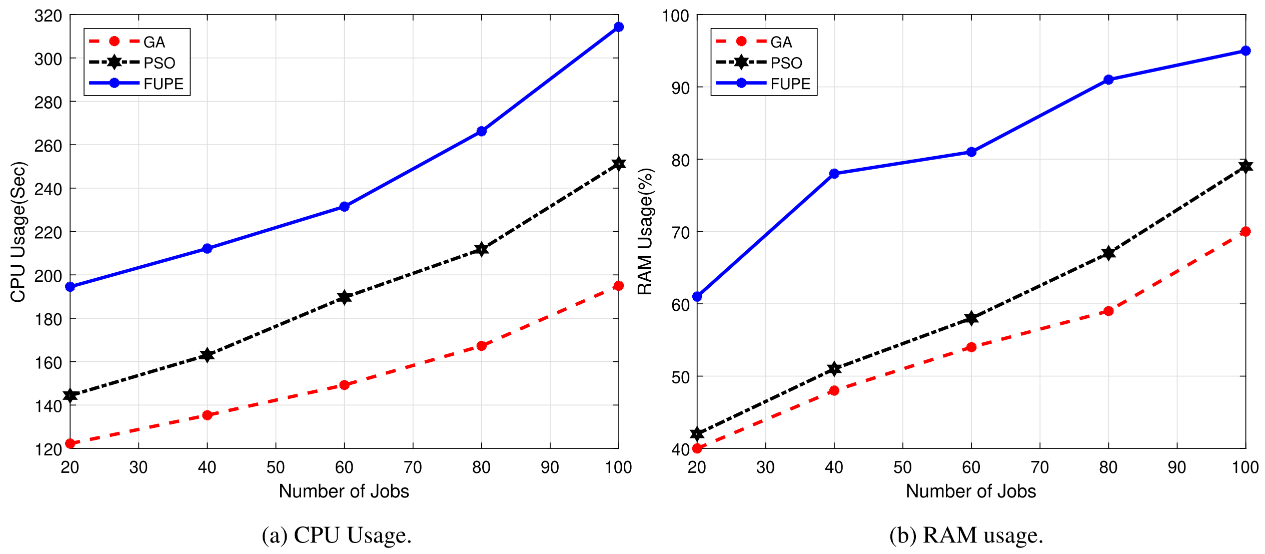


Fig. 10. The comparison results for the required CPU and RAM compared to the other solutions.

untrusted resources. To take another example from [49] execution cost is calculated by varying the number of risk rates. So, as FUPE is a security aware IoT–fog task scheduler, for evaluation we focus on the metrics of time by varying a security variable.

We argue that FUPE is suitable to be implemented in real examples of disturbed systems such as Web Operating Systems [98] with the aid of peer-to-peer (P2P)/semantic P2P Grid architectures [69]. It has *two* substantial benefits. First, the process of the user applications is done at the fog devices which are close to the users' devices. Accordingly, response time is decreased which causes more user satisfaction. Second, as security is a critical concern for users' applications in web OS, FUPE provides a suitable security level for them on fog devices.

6. Conclusions and future directions

This study introduced a security-aware fog task scheduler, called *FUPE*, that considers security issues to effectively assign IoT end-user tasks to fog devices in SDN architecture. FUPE induces the remaining amount of RAM size and CPU capacity, end-user tasks CPU need, and trust degrees as input parameters and implements a multi-objective approach jointly based on PSO and fuzzy theory to assign applications' tasks to fog devices. FUPE tackles the TCP SYN flood attack as the most common denial of service attacks. Our experiments using an IoT-based scenario illustrate the effectiveness of the presented approach. Results show that FUPE outperforms both the metaheuristic methods, i.e., GA and PSO. Compared against the former bio-inspired method, FUPE improves average response time by 11% and network utilization by 10%. It compared against the latter method and improves average response time by 17% and network utilization by 22%. For the future, we will have exhaustive study of a security framework to meet the requirements of 5G mobile edge computing. Moreover, we want to put forward a comprehensive security-driven task scheduling approach in these scenarios using different optimization techniques such as MOPSO and genetic algorithms to tackle the other types of attacks in IoT–fog networks.

CRediT authorship contribution statement

Saeed Javanmardi: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - original draft, Writing - review & editing, Visualization. **Mohammad Shojafar:** Conceptualization, Methodology, Validation, Formal analysis, Investigation, Writing - original draft, Writing - review & editing. **Reza Mohammadi:** Conceptualization, Methodology, Validation, Formal analysis, Investigation, Writing - original draft, Writing - review & editing. **Amin Nazari:** Software, Visualization. **Valerio Persico:** Conceptualization, Methodology, Validation, Formal analysis, Investigation, Writing - original draft, Writing - review & editing. **Antonio Pescapè:** Conceptualization, Methodology, Validation, Writing - review & editing, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Elazhary H. Internet of Things (IoT), mobile cloud, cloudlet, mobile IoT, IoT cloud, fog, mobile edge, and edge emerging computing paradigms: Disambiguation and research directions. J Netw Comput Appl 2019;128:105–40.
[2] Aceto G, Persico V, Pescapé A. The role of Information and Communication Technologies in healthcare: taxonomies, perspectives, and challenges. J Netw Comput Appl 2018;107:125–54.
[3] Aceto G, Persico V, Pescapé A. A survey on information and communication technologies for industry 4.0: State-of-the-Art, taxonomies, perspectives, and challenges. IEEE Commun Surv Tutor 2019;21(4):3467–501.
[4] Gill SS, Garraghan P, Buyya R. Router: Fog enabled cloud based intelligent resource management approach for smart home IoT devices. J Syst Softw 2019;154:125–38.
[5] Botta A, De Donato W, Persico V, Pescapé A. Integration of cloud computing and internet of things: a survey. Future Gener Comput Syst 2016;56:684–700.
[6] Yousefpour A, Ishigaki G, Jue JP. Fog computing: Towards minimizing delay in the internet of things. In: 2017 IEEE international conference on edge computing (EDGE). IEEE; 2017, p. 17–24.
[7] Mahmud R, Kotagiri R, Buyya R. Fog computing: A taxonomy, survey and future directions. In: Internet of everything. Springer; 2018, p. 103–30.
[8] Dastjerdi AV, Gupta H, Calheiros RN, Ghosh SK, Buyya R. Fog computing: Principles, architectures, and applications. In: Internet of things. Elsevier; 2016, p. 61–75.
[9] Wan J, Chen B, Wang S, Xia M, Li D, Liu C. Fog computing for energy-aware load balancing and scheduling in smart factory. IEEE Trans Ind Inf 2018;14(10):4548–56.
[10] Ghobaei-Arani M, Souri A, Rahmanian AA. Resource management approaches in fog computing: A comprehensive review. J Grid Comput 2019;1–42.
[11] Roman R, Lopez J, Mambo M. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. Future Gener Comput Syst 2018;78:680–98.
[12] Alaba FA, Othman M, Hashem IAT, Alotaibi F. Internet of things security: A survey. J Netw Comput Appl 2017;88:10–28.

[13] Hassija V, Chamola V, Saxena V, Jain D, Goyal P, Sikdar B. A survey on IoT security: application areas, security threats, and solution architectures. IEEE Access 2019;7:82721–43.
[14] Ni J, Zhang K, Lin X, Shen XS. Securing fog computing for internet of things applications: Challenges and solutions. IEEE Commun Surv Tutor 2017;20(1):601–28.
[15] Sathwara S, Parekh C. Distributed denial of service attacks–TCP syn flooding attack mitigation.. Int J Adv Res Comput Sci 2017;8(5).
[16] Dainotti A, Pescapé A, Ventre G. A cascade architecture for dos attacks detection based on the wavelet transform. J Comput Secur 2009;17(6):945–68.
[17] Farhady H, Lee H, Nakao A. Software-defined networking: A survey. Comput Netw 2015;81:79–95.
[18] Sahay R, Meng W, Jensen CD. The application of software defined networking on securing computer networks: A survey. J Netw Comput Appl 2019;131:89–108.
[19] Salman O, Elhajj I, Chehab A, Kayssi A. Iot survey: An sdn and fog computing perspective. Comput Netw 2018;143:221–46.
[20] Megyesi P, Botta A, Aceto G, Pescapé A, Molnár S. Challenges and solution for measuring available bandwidth in software defined networks. Comput Commun 2017;99:48–61.
[21] Li C, Qin Z, Novak E, Li Q. Securing sdn infrastructure of IoT–fog networks from mitm attacks. IEEE Internet Things J 2017;4(5):1156–64.
[22] Ramprasath J, Seethalakshmi V. Secure access of resources in software-defined networks using dynamic access control list. Int J Commun Syst 2021;34(1):e4607.
[23] Vishwakarma R, Jain AK. A survey of ddos attacking techniques and defence mechanisms in the IoT network. Telecommun Syst 2020;73(1):3–25.
[24] Salim MM, Rathore S, Park JH. Distributed denial of service attacks and its defenses in IoT: a survey. J Supercomput 2019;1–44.
[25] Sujana JAJ, Geethanjali M, Raj RV, Revathi T. Trust model based scheduling of stochastic workflows in cloud and fog computing. In: Cloud computing for geospatial big data analytics. Springer; 2019, p. 29–54.
[26] Auluck N, Rana O, Nepal S, Jones A, Singh A. Scheduling real time security aware tasks in fog networks. IEEE Trans Serv Comput 2019.
[27] Yan Q, Yu FR, Gong Q, Li J. Software-defined networking (sdn) and distributed denial of service (ddos) attacks in cloud computing environments: A survey, some research issues, and challenges. IEEE Commun Surv Tutor 2016;18(1):602–22.
[28] Daoud WB, Obaidat MS, Meddeb-Makhlouf A, Zarai F, Hsiao K-F. Tacrm: trust access control and resource management mechanism in fog computing. Hum-Cent Comput Inf Sci 2019;9(1):28.
[29] Bawany NZ, Shamsi JA, Salah K. Ddos attack detection and mitigation using sdn: methods, practices, and solutions. Arab J Sci Eng 2017;42(2):425–41.
[30] Ahmed ME, Kim H. Ddos attack mitigation in internet of things using software defined networking. In: 2017 IEEE third international conference on big data computing service and applications (BigDataService). IEEE; 2017, p. 271–6.
[31] Mohammadi R, Javidan R, Conti M. Slicots: An sdn-based lightweight countermeasure for tcp syn flooding attacks. IEEE Trans Netw Serv Manag 2017;14(2):487–97.
[32] Verma A, Kaushal S. A hybrid multi-objective particle swarm optimization for scientific workflow scheduling. Parallel Comput 2017;62:1–19.
[33] Abdullahi M, Ngadi MA, Dishing SI, Ahmad BI, et al. An efficient symbiotic organisms search algorithm with chaotic optimization strategy for multi-objective task scheduling problems in cloud computing environment. J Netw Comput Appl 2019;133:60–74.
[34] Moore H. MATLAB for engineers. Pearson; 2017.
[35] Benblidia MA, Brik B, Merghem-Boulahia L, Esseghir M. Ranking fog nodes for tasks scheduling in fog-cloud environments: a fuzzy logic approach. In: 2019 15th international wireless communications & mobile computing conference (IWCMC). IEEE; 2019, p. 1451–7.
[36] Pourjavad E, Mayorga RV. A comparative study and measuring performance of manufacturing systems with mamdani fuzzy inference system. J Intell Manuf 2019;30(3):1085–97.
[37] Bittencourt LF, Diaz-Montes J, Buyya R, Rana OF, Parashar M. Mobility-aware application scheduling in fog computing. IEEE Cloud Comput 2017;4(2):26–35.
[38] Mahmud R, Ramamohanarao K, Buyya R. Latency-aware application module management for fog computing environments. ACM Trans Internet Technol (TOIT) 2019;19(1):9.
[39] Bitam S, Zeadally S, Mellouk A. Fog computing job scheduling optimization based on bees swarm. Enterp Inf Syst 2018;12(4):373–97.
[40] Javanmardi S, Shojafar M, Persico V, Pescapè A. Fpfts: A joint fuzzy particle swarm optimization mobility-aware approach to fog task scheduling algorithm for internet of things devices. Softw - Pract Exp 2020.
[41] Sun Y, Lin F, Xu H. Multi-objective optimization of resource scheduling in fog computing using an improved NSGA-II. Wirel Pers Commun 2018;102(2):1369–85.
[42] Pajila PB, Julie EG. Detection of ddos attack using sdn in IoT: A survey. In: Intelligent communication technologies and virtual mobile networks. Springer; 2019, p. 438–52.
[43] Evmorfos S, Vlachodimitropoulos G, Bakalos N, Gelenbe E. Neural network architectures for the detection of SYN flood attacks in IoT systems. In: Proceedings of the 13th ACM international conference on pervasive technologies related to assistive environments. 2020, p. 1–4.
[44] Kolias C, Kambourakis G, Stavrou A, Voas J. Ddos in the IoT: Mirai and other botnets. Computer 2017;50(7):80–4.
[45] Yan Q, Huang W, Luo X, Gong Q, Yu FR. A multi-level ddos mitigation framework for the industrial internet of things. IEEE Commun Mag 2018;56(2):30–6.
[46] Kumar P, Tripathi M, Nehra A, Conti M, Lal C. Safety: Early detection and mitigation of tcp syn flood utilizing entropy in sdn. IEEE Trans Netw Serv Manag 2018;15(4):1545–59.
[47] Mohammadi R, Conti M, Lal C, Kulhari SC. Syn-guard: An effective counter for syn flooding attack in software-defined networking. Int J Commun Syst 2019;32(17):e4061.
[48] Zhou L, Guo H, Deng G. A fog computing based approach to ddos mitigation in iIoT systems. Comput Secur 2019;85:51–62.
[49] Li Z, Ge J, Yang H, Huang L, Hu H, Hu H, et al. A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds. Future Gener Comput Syst 2016;65:140–52.
[50] Rjoub G, Bentahar J, Wahab OA. Bigtrustscheduling: Trust-aware big data task scheduling approach in cloud computing environments. Future Gener Comput Syst 2020;110:1079–97.
[51] Gill SS, Buyya R. Secure: Self-protection approach in cloud resource management. IEEE Cloud Comput 2018;5(1):60–72.
[52] Hu P, Dhelim S, Ning H, Qiu T. Survey on fog computing: architecture, key technologies, applications and open issues. J Netw Comput Appl 2017;98:27–42.
[53] Naranjo PGV, Pooranian Z, Shojafar M, Conti M, Buyya R. Focan: A fog-supported smart city network architecture for management of applications in the internet of everything environments. J Parallel Distrib Comput 2019;132:274–83.
[54] Rahman FH, Au T-W, Newaz SS, Suhaili WS, Lee GM. Find my trustworthy fogs: A fuzzy-based trust evaluation framework. Future Gener Comput Syst 2020;109:562–72.
[55] Shojafar M, Javanmardi S, Abolfazli S, Cordeschi N. Fuge: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method. Cluster Comput 2015;18(2):829–44.
[56] D'Arienzo M, Pescape A, Ventre G. Dynamic service management in heterogeneous networks. J Netw Syst Manage 2004;12(3):349–70.
[57] Muthanna A, A Ateya A, Khakimov A, Gudkova I, Abuarqoub A, Samouylov K, et al. Secure and reliable IoT networks using fog computing with software-defined networking and blockchain. J Sens Actuator Netw 2019;8(1):15.
[58] Farris I, Taleb T, Khettab Y, Song J. A survey on emerging sdn and nfv security mechanisms for IoT systems. IEEE Commun Surv Tutor 2018;21(1):812–37.
[59] Birkinshaw C, Rouka E, Vassilakis VG. Implementing an intrusion detection and prevention system using software-defined networking: Defending against port-scanning and denial-of-service attacks. J Netw Comput Appl 2019;136:71–85.
[60] Mehdi SA, Khalid J, Khayam SA. Revisiting traffic anomaly detection using software defined networking. In: International workshop on recent advances in intrusion detection. Springer; 2011, p. 161–80.
[61] Dotcenko S, Vladyko A, Letenko I. A fuzzy logic-based information security management for software-defined networks. In: 16th international conference on advanced communication technology. IEEE; 2014, p. 167–71.
[62] Branke J, Branke J, Deb K, Miettinen K, Słowiński R. Multiobjective optimization: Interactive and evolutionary approaches, Vol. 5252. Springer Science & Business Media; 2008.
[63] Kasabov NK. Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems. Fuzzy Sets Syst 1996;82(2):135–49.
[64] Feng S, Chen CP. Fuzzy broad learning system: A novel neuro-fuzzy model for regression and classification. IEEE Trans Cybern 2018;50(2):414–24.
[65] Hüllermeier E. Does machine learning need fuzzy logic?. Fuzzy Sets and Systems 2015;281:292–9.
[66] Alla HB, Alla SB, Touhafi A, Ezzati A. A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for cloud computing environment. Cluster Comput 2018;21(4):1797–820.
[67] Singh SP, Nayyar A, Kaur H, Singla A. Dynamic task scheduling using balanced VM allocation policy for fog computing platforms. Scalable Comput Pract Exp 2019;20(2):433–56.
[68] Arora S, Barak B. Computational complexity: A modern approach. Cambridge University Press; 2009.
[69] Javanmardi S, Shojafar M, Shariatmadari S, Ahrabi SS. Fr trust: a fuzzy reputation–based model for trust management in semantic p2p grids. Int J Grid Utility Comput 2015;6(1):57–66.
[70] Clerc M. Particle swarm optimization, Vol. 93. John Wiley & Sons; 2010.
[71] Xu R, Wang Y, Cheng Y, Zhu Y, Xie Y, Sani AS, et al. Improved particle swarm optimization based workflow scheduling in cloud-fog environment. In: International conference on business process management. Springer; 2018, p. 337–47.
[72] Angelakis V, Avgouleas I, Pappas N, Fitzgerald E, Yuan D. Allocation of heterogeneous resources of an IoT device to flexible services. IEEE Internet Things J 2016;3(5):691–700.
[73] https://www.mathworks.com/products/matlab.html, MATLAB.
[74] http://www2.imse-cnm.csic.es/Xfuzzy/, Xfuzzy.

[75] Gupta H, Vahid Dastjerdi A, Ghosh SK, Buyya R. Ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. Softw - Pract Exp 2017;47(9):1275–96.

[76] Campanile L, Gribaudo M, Iacono M, Marulli F, Mastroianni M. Computer network simulation with ns-3: A systematic literature review. Electronics 2020;9(2):272.

[77] Varga A. A practical introduction to the omnet++ simulation framework. In: Recent advances in network simulation. Springer; 2019, p. 3–51.

[78] De Oliveira RLS, Schweitzer CM, Shinoda AA, Prete LR. Using mininet for emulation and prototyping software-defined networks. In: 2014 IEEE colombian conference on communications and computing (COLCOM). IEEE; 2014, p. 1–6.

[79] Ghosh S, Busari S, Dagiuklas T, Iqbal M, Mumtaz R, Gonzalez J, Stavrou S, Kanaris L. Sdn-sim: integrating a system-level simulator with a software defined network. IEEE Commun Stand Mag 2020;4(1):18–25.

[80] Wang J, Li D. Adaptive computing optimization in software-defined network-based industrial internet of things with fog computing. Sensors 2018;18(8):2509.

[81] Xiao K, Liu K, Xu X, Zhou Y, Feng L. Efficient fog-assisted heterogeneous data services in software defined vanets. J Ambient Intell Humaniz Comput 2019;1–13.

[82] Lv Z, Xiu W. Interaction of edge-cloud computing based on sdn and nfv for next generation IoT. IEEE Internet Things J 2019;7(7):5706–12.

[83] Niyaz Q, Sun W, Javaid AY. A deep learning based ddos detection system in software-defined networking (SDN). 2016, arXiv preprint arXiv:1611.07400.

[84] Duan X, Wang X. Fast authentication in 5g hetnet through sdn enabled weighted secure-context-information transfer. In: 2016 IEEE international conference on communications (ICC). IEEE; 2016, p. 1–6.

[85] Lange S, Gebert S, Zinner T, Tran-Gia P, Hock D, Jarschel M, et al. Heuristic approaches to the controller placement problem in large scale sdn networks. IEEE Trans Netw Serv Manag 2015;12(1):4–17.

[86] Cho H-H, Lai C-F, Shih TK, Chao H-C. Integration of sdr and sdn for 5g. IEEE Access 2014;2:1196–204.

[87] Kupershtein LM, Martyniuk TB, Voitovych OP, Kulchytskyi BV, Kozhemiako AV, Sawicki D, et al. Ddos-attack detection using artificial neural networks in matlab. In: Photonics applications in astronomy, communications, industry, and high-energy physics experiments 2019, Vol. 11176. International Society for Optics and Photonics; 2019, p. 111761S.

[88] Ali S, Alvi MK, Faizullah S, Khan MA, Alshanqiti A, Khan I. Detecting ddos attack on sdn due to vulnerabilities in openflow. In: 2019 international conference on advances in the emerging computing technologies (AECT). IEEE; 2020, p. 1–6.

[89] Polat H, Polat O, Cetin A. Detecting ddos attacks in software-defined networks through feature selection methods and machine learning models. Sustainability 2020;12(3):1035.

[90] Kaur G, Gupta P. Hybrid approach for detecting ddos attacks in software defined networks. In: 2019 twelfth international conference on contemporary computing (IC3). IEEE; 2019, p. 1–6.

[91] Fadlil A, Riadi I, Aji S. Ddos attacks classification using numeric attribute-based Gaussian Naive Bayes. Int J Adv Comput Sci Appl (IJACSA) 2017;8(8):42–50.

[92] David J, Thomas C. Efficient ddos flood attack detection using dynamic thresholding on flow-based network traffic. Comput Secur 2019;82:284–95.

[93] Wang H, Jia Q, Fleck D, Powell W, Li F, Stavrou A. A moving target ddos defense mechanism. Comput Commun 2014;46:10–21.

[94] Hazra A. Using the confidence interval confidently. J Thorac Dis 2017;9(10):4125.

[95] Conti M, Lal C, Mohammadi R, Rawat U. Lightweight solutions to counter ddos attacks in software defined networking. Wirel Netw 2019;25(5):2751–68.

[96] Guo F, Chiueh T-c. Sequence number-based mac address spoof detection. In: International workshop on recent advances in intrusion detection. Springer; 2005, p. 309–29.

[97] Dasgupta D, Gomez J, Gonzalez F, Kaniganti M, Yallapu K, Yarramsetti R. MMDS: multilevel monitoring and detection system. In: Proceedings of the 15th annual computer security incident handling conference. 2003, p. 22–7.

[98] Javanmardi S, Shojafar M, Shariatmadari S, Abawajy JH, Singhal M. Pgsw-os: a novel approach for resource management in a semantic web operating system based on a p2p grid architecture. J Supercomput 2014;69(2):955–75.