Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Attribute reduction based scheduling algorithm with enhanced hybrid genetic algorithm and particle swarm optimization for optimal device selection

Nweso Emmanuel Nwogbaga[1,2], Rohaya Latip[1*], Lilly Suriani Affendey[3] and Amir Rizaan Abdul Rahiman[1]

Abstract

The applications of the Internet of Things in different areas and the resources that demand these applications are on the increase. However, the limitations of the IoT devices such as processing capability, storage, and energy are challenging. Computational offloading is introduced to ameliorate the limitations of mobile devices. Offloading heavy data size to a remote node introduces the problem of additional delay due to transmission. Therefore, in this paper, we proposed Dynamic tasks scheduling algorithm based on attribute reduction with an enhanced hybrid Genetic Algorithm and Particle Swarm Optimization for optimal device selection. The proposed method uses a rank accuracy estimation model to decide the rank-1 value to be applied for the decomposition. Then canonical Polyadic decomposition-based attribute reduction is applied to the offload-able task to reduce the data size. Enhance hybrid genetic algorithm and particle Swarm optimization are developed to select the optimal device in either fog or cloud. The proposed algorithm improved the response time, delay, number of offloaded tasks, throughput, and energy consumption of the IoT requests. The simulation is implemented with iFogSim and java programming language. The proposed method can be applied in smart cities, monitoring, health delivery, augmented reality, and gaming among others.

Keywords: Computation offloading, Mobile edge computing, Task and resource scheduling, Attribute reduction

Introduction

The rapid increase in the internet of things applications in our daily lives has led to the high production of data in different fields. The application areas such as Virtual Reality (VR), Augmented Reality (AR), and Elderly Care Monitoring involve heavy data and they are also computationally intensive. However, the limited processing capability and battery life capacity of the mobile devices are their inherent limitations. These limitations make it difficult for mobile devices to be used to execute such computationally intensive tasks [1–3]. Mobile Cloud Computing (MCC) helps to provide on-demand access to a shared pull of resources that have almost unlimited capability to assist mobile devices with their limitations [4–8]. Moving a massive volume of data from mobile devices to the cloud servers increases the latency. Because of the distance between mobile devices and cloud infrastructures, it is difficult, sometimes for real-time responses to be achieved. It is difficult to achieve real-time responses when real-time IoT applications are needed especially in healthcare, video games, self-driving cars, and natural language processing in MCC.

Mobile edge computing (MEC) was introduced to reduce the latency by moving the processing point from the cloud server to the mobile edge server [9]. MEC is characterized by low latency, low cost of processing, and

*Correspondence: rohayalt@upm.edu.my

[1] Department of Communication Technology and Network, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Seri Kembangan, Malaysia
Full list of author information is available at the end of the article

low energy consumption [7]. Computation offloading is the technique to offload computationally intensive tasks to MEC or MCC because of the inherent limitations of the mobile devices [10–13].

In the Internet of Things ecosystem, computation offloading is a very important step [14, 15]. It provides assisted means of processing a large amount of data generated by numerous IoT devices, speeds up processing on intensive tasks, and saves the battery life of the mobile devices [16]. There are so many definitions of computation offloading by different authors [17, 18]. Some looked at it as a means of saving battery life, minimizing processing time, or both. Wu, et.al., [19] defined Computation offloading as the decision on how to improve the IoT network quality of service (QoS) by minimizing response time, energy consumption, increasing throughput, etc. In the computation offloading policy, the decision is made based on when to perform the computation locally and when to offload the computation to the remote processing node. Computation offloading is the process of transferring resource-intensive computational tasks to a remote processor or external platform such as cloud or fog devices [19]. Gnana et.al, defined Computation Offloading as a paradigm in IoT applications environment to improve mobile services capabilities through dynamic migration of heavy computational tasks to a higher capability server in fogs or clouds. Computation offloading provides dynamic offloading that saves energy for mobile hosts while executing intensive computational services, which are capable of draining a device's battery if executed locally [20, 21]s. The mobile host must decide on the tasks to be offloaded to fog or cloud, to reduce the energy usage while satisfying a computational time constraint [22]. An emerging and prospective computing paradigm mobile cloud computing can significantly enhance the quality of service in the IoT environment and save energy for smart mobile devices (SMDs) through computation offloading. MCC needs to offload resource-intensive Computations from smart mobile devices (SMDs) to the cloud through wireless access, which is called computation offloading [23].

There are two ways we can view computational offloading [24]

- Coarse-grained procedure and
- Fine-grained procedure

Coarse grain (static) computational offloading considers the task either at the mobile device or in the fog or cloud. It does not involve processing at both mobile devices and fog or cloud servers. The decision needed here is either to execute the whole workload at the mobile node or execute the whole workload at the fog or cloud by offloading the task. Fine-grain (dynamic) computational offloading executes some part of the task at the mobile node while some of the tasks are offloaded to the fog or cloud node depending on certain criteria (such as the response time, delay, and energy consumption). In other to balance delay, network congestion, and energy usage in IoT applications, fine-grain computational offloading is preferred. This is because in coarse grain procedure, if you maintain processing at the mobile device, some of the computationally intensive tasks will drain off the battery life, and the processing time will exceed the required response time (latency threshold) of the task, whereas if you maintain the processing at the edge or cloud server for all the tasks, the network traffic will be flooded and some of the less computationally intensive tasks with low latency will not meet the response time requirement. To balance these issues in IoT applications, the dynamic computational offloading procedure is better to get the trade-off between network traffic, response time, delay, and energy consumption to maximize the overall network quality of service and quality of experience [25]. The decision on the efficient layer to execute the task is still challenging. Secondly, the volume of data involved in today's IoT applications if they are to be offloaded in their normal data size will keep on slowing down the network communication. Therefore, there is a need to downsize the offload-able data size before passing it through the network. This will make the data transfer faster; reduce latency and energy consumption. The major issue in computational offloading is the problem of transmission delay incurred in the process of offloading the task to the remote processing node. Since the transmission delay is a result of heavy data size, applying the attribute reduction technique during the offloading process will help to downsize the offload-able task before the offload. The reduced data size will reduce the delay encountered during the offloading. However, few researchers have considered downsizing the offload-able data size during computational offloading [26].

Some researchers have made several attempts to solve the problem of computation offloading and scheduling in handling real-time mobile application requests and mobile device battery life limitations. Q learning has been proposed for task offloading in [16]. But the approach is time-consuming because of all the possible options to be considered. A Deep Q Learning-based algorithm is proposed [4] for computation offloading, which has high computational complexity. PSO for resource allocation is proposed in [27]. PSO has its inherent limitation of being trapped in the local optima. The genetic algorithm (GA) based approach is proposed in [14], but GA has the limitation of taking

a long time to converge and works better with a large number of iterations.

This work proposes an enhanced hybrid genetic algorithm and particle swarm optimization (eh-GA-PSO) algorithm to eliminate the problem of PSO and GA-based algorithms for choosing the optimal device for processing. GA-based algorithms produce better results than other algorithms especially when the iteration size is big. But with a high iteration size, means that the device selection process will take more time to reach the optimal solution [28]. On the other side, PSO-based algorithms always produce better results faster than other algorithms. But the problem of PSO based algorithms is that most times their result may not be accurate due to the speedy convergences, which at times make the PSO based algorithms to be trapped in the local optima solution [29–31]. Because of the advantages of GA producing a more accurate result and PSO converging faster, we, therefore, proposed the eh-GA-PSO algorithm to eliminate the disadvantages of GA taking too much time to converge and PSO being trapped in the local optima. The researcher applied an enhanced hybrid genetic algorithm and particle swarm optimization algorithm for optimal device selection. The decision on where to offload the task is based on the time it will take to process the task, which is determined by the processing time at mobile, fog, or cloud nodes together with their transmission delay. The main contributions of this paper are as follows:

- We introduced attribute reduction in computation offloading
- We developed the rank accuracy estimation model (RAEM) in calculating the rank-1 value to be applied for the attribute reduction
- We proposed an enhanced hybrid Genetic Algorithm and Particle Swarm Optimization (eh-GA-PSO) for the optimal device selection
- We proposed Dynamic tasks scheduling algorithm (DTSA) based on attribute reduction for the task and resources scheduling
- We evaluated the proposed approach based on response time, delay, energy consumption, resource utilization, and the number of offloaded tasks. Our results validate the superiority of our proposed approach compared with the present state of the art in task offloading and resource scheduling approaches.

The rest of the paper is organized as follows: The related works are presented in Section 2. Section 3 is the proposed system. Section 4 is the discussion of the Simulation and Results. Finally, the conclusion of the paper is presented in Section 5.

Review of the related work

Computation offloading in IoT, Fog, and Cloud computing has attracted researchers' attention recently. The major challenge is how to apply the IoT in real-time data analysis. Most IoT applications involve sensitive cases that need real-time responses for instance self-driving cars, medical, monitoring, and traffic control among others. The limitations of IoT devices make it difficult to perform all the data analysis on the mobile device because of the limited processing capability, storage, and battery life. Because of these IoT limitations, there is a need for offloading to higher processing capability nodes (Fog or Cloud). Offloading from IoT to another remote device introduces another problem. The problem of delay is incurred as the computation is moved from mobile devices to Fog or Cloud. To address this problem, some researchers have presented different solutions, but none of them have considered applying attribute reduction to downsize the size of offload-able data. In this section, we present the related work to computation offloading in IoT- fog and cloud environments.

In [32], provisioning offloading as a service was presented. The offloading problem of IoT requests was considered. Large-scale offloading in an IoT environment with AutoScaler for task schedule was proposed. They also proposed a simulator that uses a mini-max algorithm to generate the offloading workload. Though the AutoScaler front-end component increases the response time of a request by ≈150 milliseconds, it reduces cost over the architecture of one IoT device and one server which may not be realistic in real life. Introducing fog devices will improve that response time. Secondly, introducing data attribute reduction will improve the transmission time and therefore reduce the overall response time. In [16], the problem of computation offloading is presented. They proposed secOFF-FCIoT for offloading computationally intensive tasks with low data size. The paper identified tasks requiring heavy computation with minimal data sharing and considered them for offloading. They considered tasks with small data sizes because it will take little time to be offloaded while the processing time will be highly reduced at the higher processing node (Fog or Cloud). However, those tasks that require heavy computations that have large data sizes were not considered for offloading in their approach because of high transmission time required. Applying attribute reduction to reduce the data size during transmission will improve the system performance thereby enabling both tasks with heavy computational requirement and large data size to be offloaded at a minimal transmission delay instead of

processing such heavy computation intensive tasks at mobile with limited processing capability. This will improve the system performance. It will as well allow IoT application that involves heavy computations like machine learning application with heavy data such as images or videos to be applied in low latency applications, for instance in telemedicine, vehicular network, monitoring systems, smart city, etc. Autonomous computation offloading is proposed in [7] to address the challenges of IoT application issues such as resource-intensive and time-intensive applications. Offloading scheme based on the combination of Q Learning and Deep Q Learning algorithm is proposed [4] to address the problem of task offloading and caching defined as a nonlinear problem to reducing the time and energy of the mobile devices. Monitor-Analysis-Plan-Execution control loop for mobile edge computing is proposed in [6] to decide on whether to execute locally or offload computation of the tasks to edge or cloud layer. The proposed method executes and updates the parameters of the system in the problem space. The decision is made based on the Bayesian learning automata probabilistic method. Latency-sensitive and resource-hungry issues in IoT application is addressed [33]. The paper proposed constrained multi-objective evolutionary algorithms to address IoT computation offloading issues in the collaborative edge-cloud computing environment. The proposed method improved the network quality of service in terms of time and energy consumption of the IoT devices. The problem of system cost based on delay and energy consumption is presented in [34]. The paper used a game-theoretical approach to analyze the computation offloading decision problem of IoT applications. Nash equilibrium was derived based on the definition of the potential game. The paper proposed a computation offloading decision (COD) algorithm via IoT-Cloudlet-Cloud decentralized approach. The proposed method improved the system cost in terms of processing delay and energy consumption. The IoT device's energy consumption in the computation offloading to the cloud through the base station was significantly improved. However, introducing attribute reduction to the transmitted tasks (especially for heavy image and video data) will significantly reduce the transmission delay. Deep reinforcement learning-based computational offloading for IoT devices with energy harvesting was proposed by [35]. The paper presented a Q-learning dynamic scheme to decide on an offloading policy for the system. The scheme determines a portion of the data be offloaded to mobile edge computing (MEC) devices following the system bandwidth, battery life level, and amount of energy harvested. The scheme was proposed based on a Markov Decision Process (MDP) and Q-learning method for optimal policy. But, the approach takes a significant time to converge. The Q-learning dynamic scheme uses the machine learning concept in computational offloading. However, the proposed approach doesn't consider important issues, for example, mobility, cost of use of MEC, bandwidths changes, and mobility of the devices, Computational offloading based on deep reinforcement learning in IoT that enabled device-to-device (D2D) communication has been proposed in [36]. The paper presented an optimal decision-making offloading algorithm for IoT systems in terms of user and cloudlet behaviour similar to [35]. But unlike [35] where offloading is achieved on one MEC at a time, the paper addressed the issue of offloading to several cloudlets (edge devices). The paper focused on the composite behavior of the queue in the cloud and the distance between the user equipment and the cloudlet. Zhang et al. [27] presented a hybrid computation offloading algorithm based on queue theory for workflow and PSO for resource allocation. The hybrid computation offloading is between the cloud and the mobile devices while cloudlet is used to calculate the waiting time and schedule the task to mobile or cloud-based on whichever place it will be processed faster. Analysis of time and energy consumption is proposed by leveraging the queue theory. The objective function is based on the deadline constraints of the mobile request concerning processing time and energy consumption. The task scheduling scheme is based on the queue model and First Come First Served (FCFS) while waiting time in the cloudlet is based on M/M/m/ ∞ queue. Particle Swarm Optimization (PSO) based heuristic algorithm is implemented to schedule mobile services by selecting mobile services which have not been scheduled in the workflow. FCFS based scheduling is usually slower compared to other scheduling approaches while using the PSO for selecting the cloud or mobile node sometimes improves network performance. Some researchers applied PSO to this kind of problem because IoT devices exhibit characteristics of a swarm. In [22] a Dynamic Task Offloading (DyTO) is proposed. The paper introduced the concept of a surrogate object (SO) in computational offloading in mobile cloud computing. The surrogate objects represent each mobile host at the mobile support station (MSS). The surrogate object takes requests from the mobile host and decides where to process the task (request) either at the host station, nearby station or federated cloud based on the compute, storage, and time required by the request. A surrogate object connected to the mobile cloud computing network handles the issue of tracking the mobile host thus maintaining the consistency of resources to the mobile host as it negotiates or loses connectivity. The surrogate object within the mobile cloud computing network preserves the information to guarantee suitable distribution of data particularly

when the network is unsteady. The proposed Dynamic Task Offloading model saves energy. It improves execution time because disruptions in processing as a result of disconnection or differences in available resources caused by mobility are avoided. Although the issue of tasks sensitivity is not considered, since tasks are considered on first come first serve basis, some tasks may need urgent attention more than the existing ones and need to be considered first. The scheduling problem in a computational grid environment is presented in [37]. The position vector is changed from the continuous value to the discrete value via small position value (SPV) rule. The paper aims at generating an optimal scheduling policy to enable the tasks to be completed in a minimum time as well as to achieve efficient resources utilization. The paper showed that PSO is faster than GA in terms of response time. Lei Yang [38] presented task offloading based on directed acyclic graph applications in edge computing. The paper aims at addressing the issue of computational offloading for computation – intensive application in industrial resources to offload tasks to edge server or cloud. The paper focused mainly on the industrial application that has delay and energy consumption constrains. They formulated a task offloading approach for industrial services for three layer paradigm of industrial, edge, and cloud layers. The paper proposed a linear programming based algorithm called ASO and intelligent heuristic based algorithm called Pro-ITGO to address the offloading problem. ASO and Pro-ITGO algorithms reduced the average energy usage of the industrial devices by 35% compared with the existing state of the art offloading algorithms.

In summary, it is clear from the literature that achieving real-time responses from IoT applications is still an issue of importance in IoT-Fog-Cloud distributed computing. It is clear also that offloading is a technique in MEC and MCC that also comes with its inherent transmission delay. Hence, there is a need to research how to minimize the delay resulting from offloading. It will make offloading faster and thereby make MEC and MCC more efficient in IoT applications.

Proposed system

In this section, the proposed system for dynamic computation offloading is presented.

Problem statement

The problem of computation offloading is addressed in this paper for the IoT-Fog-Cloud system. The problems of response time, delay, and energy utilization of mobile devices are addressed, especially for real-time IoT applications such as Smart City, Self-driven Cars, IoT Retail Shops, Smart Homes, Farming, Wearable, Smart Grids, and Industrial Internet among others. The mobile devices' heavy data generation especially when it involves video and images and the limitations of mobile devices in terms of processing capability and battery life were considered. We, therefore, present a dynamic tasks scheduling algorithm for IoT-Fog-Cloud task offloading to achieve high network performance in terms of response time, energy utilization, delay, resource utilization, and throughput.

System overview

We consider a network of N IoT devices, J fog devices, and K Virtual Machine (VMs) such that n = {1, 2, 3 … N}, j = {1, 2, 3 … J}, and k = {1, 2, 3 … K} as in Fig. 1. The IoT devices are connected to the Fog nodes through a smart gateway. The Fog nodes are connected to the cloud, thereby creating a hierarchical network continuum to the cloud. When tasks are coming from the IoT, the IoT evaluates the task to determine whether the task can be processed at the IoT, in fog, or the cloud-based on where the response time, energy consumption, and delay will be minimized while throughput and resource utilization of the network is maximized.

IoT layer

The IoT layer generates and measures the tasks based on the data involved in each task. They offload those tasks that involve heavy data size and are highly computational intensive to a higher processing node (fog or cloud). They also receive responses to the processed tasks from fog and cloud. IoT devices offload tasks with heavy data size and high computational intensive because of their limitations of processing capacity and battery life [39, 40].

Network layer

The network node is made up of a gateway and router which sometimes can function as fog devices in a small-scale network [41–43]. In this work, the smart gateway is used for data validation. It secures the network by evaluating the data coming from each IoT device using the Neuro-Fuzzy logic model [16]. The Neuro-Fuzzy model checks the incoming data, if invalid data is detected, the data will be discarded while the IoT device that sent the data will be requested to resend the data.

Fog layer

The fog layer comprises fog devices. Fog devices are systems with higher configurations than mobile devices but with lesser than cloud infrastructures. Fog devices can perform all the functionalities of the cloud, though their processing capabilities and storage are lower than that of

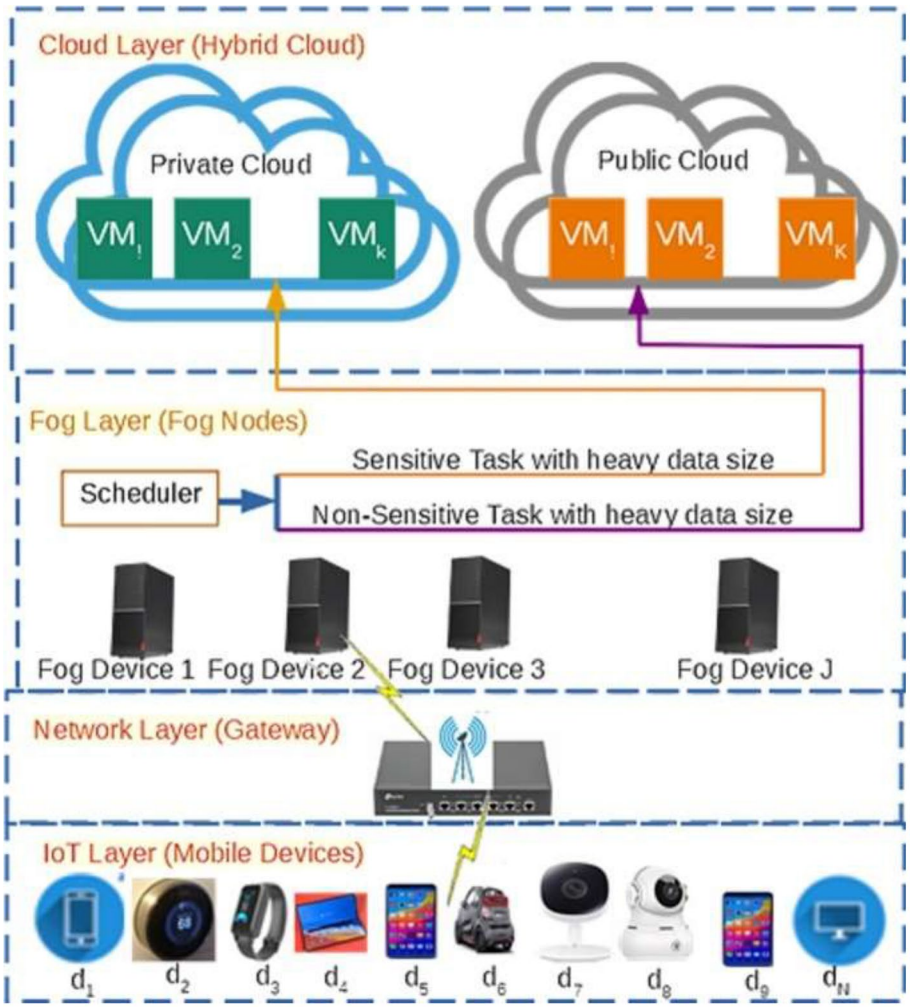

Fig. 1 System Model (d represents devices from I to N, Fog devices range from 1 to J, and cloud virtual machines (VM) range from 1 to K). N is the number of Mobile devices, J is the number of Fog devices and K is the number of virtual machines in the cloud infrastructure

the cloud systems. They support localized services and data analytics closer to the user [44–46]. Fog devices can be installed on the roadside unit or on moving cars.

Cloud layer

The cloud layer is made up of high powerful configuration systems. They have infinite processing and storage capacity. The cloud devices consist of high computational intensive devices, which can be public or private. The public cloud is provided as a service over the internet. The public cloud is based on multi-tenancy architecture. The user doesn't need to purchase any hardware or software. The acquisition of hardware and software are the responsibilities of the provider. But the user has to pay based on their agreed prizing model. It can be based on pay-as-you-use, yearly, or based on duration, etc.

The private cloud is the cloud infrastructure provided and managed by the corporate enterprise. It is always deployed within the firewall of their network. The user will always have full control of the private cloud. The duration of storage and services do not affect the cost [47, 48].

Optimal device selection

Multiple objectives optimization problems have been attempted by many researchers in different ways [33, 48, 49]. We adopted the available processing capability (APC) of the processing resources in calculating the fitness function to determine where to process the request. Our

proposed device selection approach combined the good qualities of genetic algorithm (GA) and particle swarm optimization (PSO) in device selection while dynamically considering the highest APC of the resources. In this subheading, we present the GA part, PSO part, fitness function, and finally the proposed algorithm.

GA part of the proposed algorithm

The proposed algorithm begins by initializing tasks and the available devices as shown in Fig. 2. After initialization of the tasks and available resources, the algorithm will check for the fitness function according to Eq. (3). If the tasks-resource allocation fits the fitness function, the process terminates else, the process will continue with the selection operator, crossover operator, and mutation operator as shown in Figs. 3, 4, and 5 respectively until the maximum iteration set for GA is reached. When the maximum iteration set for GA is reached without convergence, which is determined by the proposed fitness function, GA will pass its output to the PSO part of the proposed algorithm.

PSO part of the eh-GA-PSO algorithm

At the PSO part of the proposed algorithm, the chromosomes will be initialized based on the result from the GA part. The tasks – resources allocation will be checked for the fitness function using Eq. (3).

We considered PSO suitable for this study because mobile devices behave similarly to a swarm [16]. PSO has been applied in similar areas in science and engineering. For instance, in prediction [50], and alignment optimization [51]. In a fog environment, connectivity is through wireless connection sometimes, the resources are dynamic. The changes may be a result of weather, mobility of IoT devices, bandwidth fluctuation, and other factors in the network. Because of these reasons, the

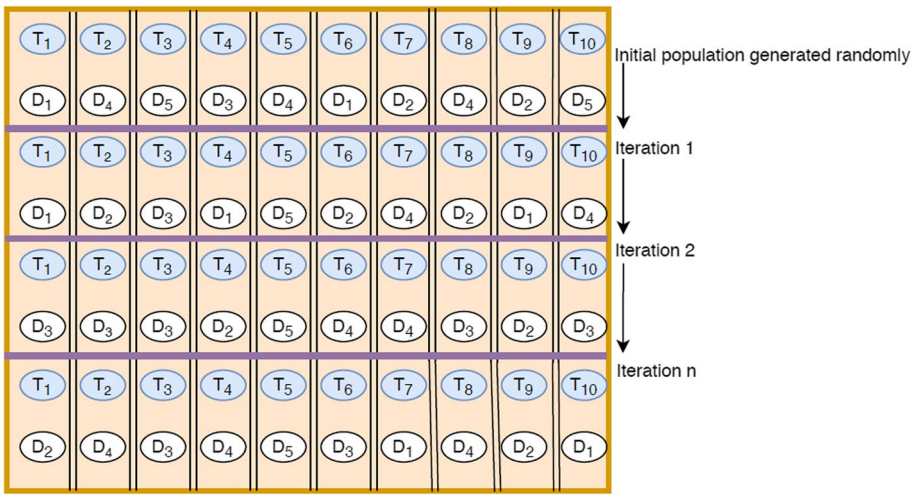

Fig. 2 Tasks – Resources Initialization Process (T represents Task while D represents device or cloud VM)

Algorithm 1: Selection Operator

Input: Randomly Generate Chromosomes
Output: Chromosomes best fitness
1 **Set:** Number of Tournaments

2 **for** $i = 0$ *To numberOfTournament* **do**
3 id = Math.Random() * chromosomes-size // Randomly select the chromosomes
4 tournament[i] = get-chromosome(id)
5 *Fitnessvalue* ← *tounament*[*i*].*Fitness*

Fig. 3 Selection Operator for the GA part of the eh-GA-PSO Algorithm

Algorithm 2: Crossover Operator

Input: Two Chromosomes

Output: Offspring Chromosome

```
r =(Math.Random()*chromosome.lenght)
for i = 0, j = 0 to r do
    offspring-chromosome [j] = chromosome[i]
for i = r to chromosome.lenght do
    offspring-chromosome [j] = chromosome2[r]
```

Fig. 4 Crossover Operator for the GA part of the eh-GA-PSO Algorithm

Algorithm 3: Mutation Operator

Input: offspring-Chromosomes

// Offspring Chromosomes from the Crossover Operator

Output: New-Chromosome

```
Set: MutationRate = 0.5
if Math.random() <MutationRate) then
    t1 = Rand[0,1] * offspring-chromosome.lenght      // Select a random number t1 between 0 and 1
    t2 = Rand[0,1] * offspring-chromosome.lenght      // Select a random number t2 between 0 and 1
    if offspring-chromosome[t1] = offspring-chromosome[t2] then
        Swap (offspring-chromosome[t1], offspring-chromosome[t2])
```

Fig. 5 Mutation Operator for the GA part of the eh-GA-PSO Algorithm

workload at the fog node changes frequently. To ensure the reliability of IoT getting responses as when due, there is a need to consider the data involved in a particular request, the bandwidth, and the processing capability at the fog, that can yield the best response time to the IoT request. PSO is a meta-heuristic algorithm that imitates the intelligence of a swarm. It uses communication and learning as its basic principles. Individual particles create their route to the optimal solution and communicate with every member. Every member of the particles gets this information and learns the best course of action to take. They will create a single global optimal solution for the whole swarm. This cooperation and intelligence exhibited by the swarm makes them achieve results faster [52]. As stated earlier, it is assumed that the IoT devices behave like swarms that can change their location thereby affecting the bandwidth and processing capability at the fog node or cloud node. The objective of PSO is to find the optimal solution through individual particles' cooperation and communication among themselves to find the global optimal solution. Assuming we have a certain number of tasks (I) hereafter referred to as particles and J fog nodes or K cloud VMs referred to as dimensional search space, the individual particles changes their position and velocity according to Eqs. (1 and 2) [52]

$$X(t+1) = X(t) + V(t+1) \tag{1}$$

where $X(t+1)$ is the position for the particle at t + 1 time, $X(t)$ is the position of the particle at t time and $V(t+1)$ is the velocity of the particle at t + 1 time.

The new velocity at t + 1 time is calculated in Eq. (2)

$$V(t+1) = \omega V(t) + C_1 R(0,1) * \left(X_{pbest} - X(t)\right) + C_2 R(0,1) * \left(X_{gbest} - X(t)\right) \tag{2}$$

where V(t) is the current velocity for the particle, ω is the inertia weight, C_1 and C_2 are the weighting coefficients for the personal best and global best positions, respectively. X_{pbest} is the particle's best-known position. X_{gbest} is the global best-known position of the particles, and R is a random number between 0 and 1.

Difference and similarity between PSO and GA

PSO is like a genetic algorithm because both systems are initialized with a population of randomly sampled solutions. But PSO is different from GA because each possible solution is also allocated a randomized velocity and the possible solution are then flown among the problem space [52].

Fitness function for optimal device selection

We considered the availability of fog node devices or VMs by their Available Processing Capability (APC). The APC of all available devices will be ranked with a value (w_1) and optimal device selection will be based on Eq. (3)

$$f(x_i) = w_1 ApC(x_i) \tag{3}$$

where x_i represents the particular task in consideration and i ranges from 1 to the number of available tasks (I).

The proposed enhanced hybrid genetic algorithm and particle swarm optimization algorithm (eh-GA-PSO)

In this paper, we proposed an Enhanced hybrid Genetic Algorithm and Particle Swarm Optimization Algorithm (eh-GA-PSO). The eh-GA-PSO algorithm aims to schedule the workflow tasks over the available fog and cloud VM resources. The eh-GA-PSO algorithm is used for the allocation of tasks coming from the IoT devices to the optimal fog or VM machines based on where the system can achieve higher network performance in terms of response time, energy utilization, delay, and resource utilization, and throughput. The eh-GA-PSO algorithm is presented in Fig. 6. GA-based algorithms produce better results than other algorithms especially when the iteration size is big. But with a high iteration size, means that the device selection process will take longer to reach the optimal solution [28]. On the other side, PSO-based algorithms always produce better results faster than other algorithms. But the problem of PSO based algorithms is that sometimes their result may not be accurate due to the speedy convergences, which at times make the PSO based algorithms to be trapped in the local optima solution [29–31]. Because of the advantages of GA producing a more accurate result and PSO converging faster, we, therefore, proposed the eh-GA-PSO algorithm to eliminate the disadvantages of GA taking too much time to converge and PSO being trapped in the local optima. The eh-GA-PSO uses the available processing capability of the resource to determine the fitness function as presented in Eq. (3). This is achieved by truncating the iteration process of the GA halfway if the fitness function is

Algorithm 4: Enhanced Hybrid Genetic Algorithm and Particle Swarm Optimization Algorithm (eh-GA-PSO) for optimal Fog device and Cloud VM selection

Input: number of Workload (nW), number of iterations (n), Rate of crossover (Cr), Rate of mutation (Mr)
Output: gbest

```
for i = 0 To nW do
    for k = 1 to no of device or VMs do
        W(id) ← randomize(Task(id)) D(id) ← randomize(devices(id))
    Chromosomes ≙ Task[W(id), D(id)]
for j = 0 To n/2 do
    Check for Fitness using f(x_i) = w_1 ApC(x_i)
    while Fitness condition is NOT yet met for all tasks do
        Apply Selection Operator
        Apply CrossOver Operator
        Apply Mutation Operator
Chromosomes(i) ≙ set of chromosomes from last mutation operator
X_i ← Chromosome(i)
V_i ← Randomly()
for j = n/2 To n do
    Update V_i^t and X_i^t according to
    X(t+1)=X(t)+V(t+1)
    V(t+1) = wV(t) + C_1 R(0,1) * (X_pbest-X(t))+C_2 R(0,1) * (X_gbest-X(t))
    Calculate the g_best and p_best values
    Check for Fitness using f(x_i) = w_1 ApC(x_i)
    while Fitness condition is NOT yet met for all tasks do
        if Pbest(X_i)<Pbest(X_i^t) then
            Pbest(X_i)←Pbest(X_i^t)
        if Pbest(V_i)<Pbest(V_i^t) then
            Pbest(V_i)←Pbest(V_i^t)
Chromosomes ≙ set of chromosomes from last PSO iteration
```

Fig. 6 Enhanced hybrid Genetic Algorithm and Particle Swarm Optimization Algorithm (eh-GA-PSO)

not yet attained at that level and then feeding the output of GA to PSO. With the half-processed result from GA, PSO will not start with a random selection. Instead, it starts with a nearly optima solution from GA. Because PSO did not start with initial random selection, it will not be trapped in the local optima solution. Secondly, the whole process will converge faster than allowing GA to complete the whole iterations process till convergence.

Proposed Dynamic Tasks Scheduling Algorithm (DTSA)

In this paper, we propose Dynamic tasks scheduling algorithm (DTSA) based on attribute reduction. We proposed an enhanced hybrid Genetic Algorithm and Particle Swarm Optimization for optimal device selection as presented in Fig. 7. Given an offload-able task from an IoT device, the task needs to be processed and a response received within the expected time limit. However, mobile devices are limited in processing capacity, storage, and energy. The energy of the mobile device (IoT device) needs to be maximally utilized. A generalized system model of the proposed attribute reduction-based secured offloading scheme is presented in Fig. 8. The task from the mobile device involves data that can be expressed in the tensor format with a certain r rank -1 for r ϵ {1, 2 ... R}. r = 1 means that the tensor is decomposed into 1 vector for each dimension of the tensor while r = R means that the tensor is decomposed into R vectors for each dimension of the tensor. Canonical Polyadic Decomposition (CPD) decomposes original data to a certain percentage of the original data according to the CPD data decomposition ratio presented in our previous work [53]. The data transmitted over the network through CPD at times result in a slight change in the data accuracy. The essence of decomposing the data is to reduce its size during transmission over the network to reduce the transmission time and improve bandwidth usage. Therefore, to choose the r-value that will guarantee certain accuracy (AC), Eq. (4) is applied.

$$AC = \alpha \ln(R) + \beta \tag{4}$$

AC is the accuracy of the data analysis given R rank-1 values and β is the constant for estimating the AC [53]. Figure 7 presents the algorithm used to schedule the tasks generated at the IoT layer among the available devices at the IoT, fog, or cloud layer based on the fitness function of the task. The fitness function of the tasks is calculated in Eq. (3). The objective of the algorithm is to minimize the latency. The latency is calculated in Eq. (5)) [16]. While Eq. (6) is the enhanced latency when attribute reduction is applied during offloading to reduce the data size.

$$L(x_i) = \frac{sz_i \times cx_i \times u_i + bs(x_i)}{F(x_i)} \tag{5}$$

$$L_R(x_i) = \frac{sz_{iR} \times cx_i \times u_i + bs(x_i)}{F(x_i)} \tag{6}$$

$L(x_i) \geq L_R(x_i)$ where sz_i, cx_i denotes the size and complexity of the task, sz_{iR} is the reduced data size after passing through CPD, U_i denotes the mean latency of the

Algorithm 5: Dynamic tasks scheduling algorithm (DTSA) based on CPD Attribute Reduction (IoTCP-DL algorithm) and eh-GA-PSO device selection

```
Generate tasks randomly at IoT layer
Calculate the processing time at IoT (Tm), Fog (T_Fog), and Cloud (T_Cloud) required by the task
    based on processing capabilities at IoT, fog and cloud as in steps 3, 4, and 5
Tm = w/PC_m
T_Fog = w/PC_f
T_Cloud = w/PC_c
if T_M >= T_f + T_Fog or T_M >= T_f + T_c + T_Cloud then
                        // Compare the results of steps 2 to determine whether the task is offloadable
    if Task is offloadable then
        Apply IoTCP-DL Algorithm for attribute reduction
        if Fog is the optimal Node then
            i. Apply eh-GA-PSO algorithm to select the devices to allocate the task at the Fog level
            ii. Process the task and return the results
        if Cloud is the optimal node then
            i. Apply eh-GA-PSO algorithm to select the VM to allocate the task at the Cloud level
            ii. Process the task and return the results
if Task is not offloadable then
    Process the task at the IoT layer
/* Tm = processing time at the IoT/mobile device, w = worklaod, PC_m = processing capability at mobile
   */
/* PC_f = the processing capability of the fog device                                          */
/* PC_c = the processing capability of the cloud virtual machine (VM)                         */
/* T_f = transmission time from mobile to fog and T_c = transmission time from fog to cloud   */
```

Fig. 7 Dynamic tasks scheduling algorithm based on attribute reduction and eh-GA-PSO for optimal device selection algorithms

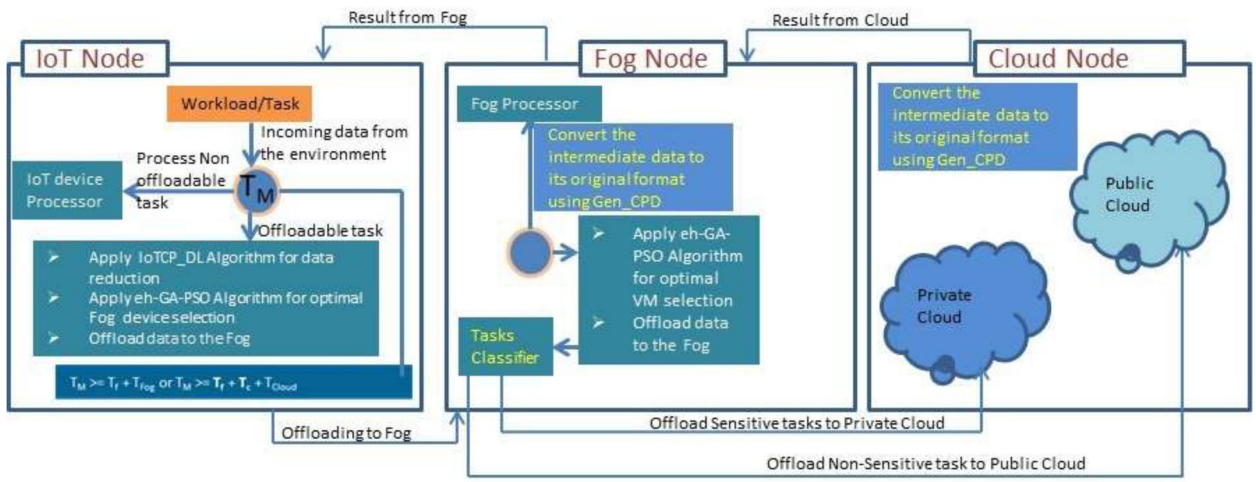

Fig. 8 Generalized system model of the proposed attribute reduction based secured offloading scheme

task, bs, and F are the current buffer size and CPU frequency of the fog or VM machines.

Simulation and results

The simulation process and its results are presented in this section.

Experimental settings

To effectively evaluate the performance of our proposed scheduling algorithm, simulation experiments are conducted. We created an IoT-Fog-Cloud system consisting of 10 mobile devices, 5 fog nodes, a smart gateway, and 1 hybrid cloud. We chose this configuration to enable us to link the mobile devices to the fog-cloud paradigm, which has higher processing capabilities than the mobile device. The simulations are conducted with the iFogSim simulator and Java programming. The iFogSim framework is designed for an efficient way of evaluating resource management policies especially as applicable to the fog paradigm concerning their impact on response time, energy consumption, latency, operational costs, and network congestion. It simulates mobile devices, Fog nodes, network links, and cloud data centers to measure performance metrics [54]. There are n tasks generated at the mobile devices which are to be offloaded to either the fog or cloud depending on where it will have reduced response time, delay, and where more tasks will be offloaded to reduce the energy consumption at the mobile nodes.

Performance metrics

i. Response time (R_T): The time between when a user places a request and when the response is received.

ii. Throughput (T_P): This is the number of tasks offloaded by mobile devices per unit of time (T) used.

$T_P = \#-\text{task Offloaded/T}$

iii. Delay (D_T): The difference in time between the actual response time and the expected (calculated) response time of a task of the application. It is also computed as follows:

$D_T = PRO_T + Queue_T + Tran_T + Propagation_T.$

where PRO_T denotes processing delay, $Queue_T$ denotes queuing delay, $Tran_T$ represents transmission delay, and $Propagation_T$ denotes propagation delay.

iv. Energy consumption (E_c): This is the amount of energy consumed by mobile devices to perform a particular task. $E_c = E_{pro} + E_{trans}$

where E_{pro} is processing energy, E_{trans} is transmission energy.

xxii. Resource utilization rate (ReU): This is the total amount of resources used as compared with the number of resources budgeted for the task. ReU is presented as the percentage of time mobile device uses the resources in 24 h.

$ReU = N_i/24 \times 100$, where N_i is the nth resource.

Results

In this section, the results from the simulations are presented. The results of our proposed offloading schemes and other existing algorithms as in [16, 22] are compared. The proposed offloading scheme offers a scalable solution for the IoT tasks offloading process. In the simulations, we set the latency for sensitive tasks to be 1 s while the

non-sensitive task's latency requirement is set to be 1.5 s throughout the simulation process. We also set the number of tasks to be n ∈ {10, 20, 30, 40, 50}. In each case, the maximum simulation period is set to 100 seconds. Sensitive of the tasks here refers to those tasks that have to satisfy certain time constraints to be acceptable to the user. In this work, tasks have two categories of time constraints. If a task has a time constraint of 1 second (1 s) it is referred to as a sensitive task, but if the tasks' time constraint is 1.5 seconds (1.5 s) it is categorized as a non-sensitive task.

Response time

Figure 9 shows the result of response time for the proposed (DTSA) offloading algorithm compared with SecOFF-FCIoT [16] and DTO-SO [22] algorithms.

The response time of the tasks increases with the increase in the number of tasks. Our proposed algorithm takes less time to respond to IoT requests. For instance, when the number of simulated tasks is 20 for sensitive tasks, the response time for our proposed algorithm is 6 ms while secOFF-FCIoT and DTO-SO are 8 ms and 11 ms respectively. When the number of simulated tasks increased to 40 for sensitive tasks, our proposed algorithm response time is 11 ms as against 13 ms and 15 ms for secOFF-FCIoT and DTO-SO respectively. In the same way, for the non-sensitive tasks, our proposed offloading scheme response time is also faster compared with secOFF-FCIoT and DTO-SO. When the number of simulated tasks is 30, our proposed algorithm response time for the non-sensitive task is 9.5 ms as against 11 ms and 15 ms for secOFF-FCIoT and DTO-SO. Therefore, the proposed approach is faster in terms of response time compared with the existing schemes.

Delay

Figure 10 shows the impact on delay for both sensitive tasks and non-sensitive tasks of the system. The delay in the offloading process increases with an increase in the number of IoT requests. For sensitive tasks, our proposed offloading scheme reduced the delay by 8% and 21% compared to secOFF-FCIoT and DTO-SO. For the non-sensitive tasks, the delay is reduced by 23% and 35% compared to secOFF-FCIoT and DTO-SO respectively. In particular, when the number of simulated tasks is 30 for sensitive tasks, the delay is 1.1 ms for our proposed scheme while it is 1.2 ms and 1.4 ms for secOFF-FCIoT and DTO-SO. In the same vein, when the number of simulated tasks is 20 for non-sensitive tasks, the delay for our proposed scheme is 0.9 ms as against 1.1 ms and 1.4 ms for secOFF-FCIoT and DTO-SO respectively. This shows that our proposed scheme reduces the delay in the offloading process of the IoT fog-cloud system compared to the existing schemes.

Offloaded tasks

Figure 11 shows the number of offloaded tasks of our proposed scheme compared with that of the secOFF-FCIoT algorithm. Our proposed dynamic offloading

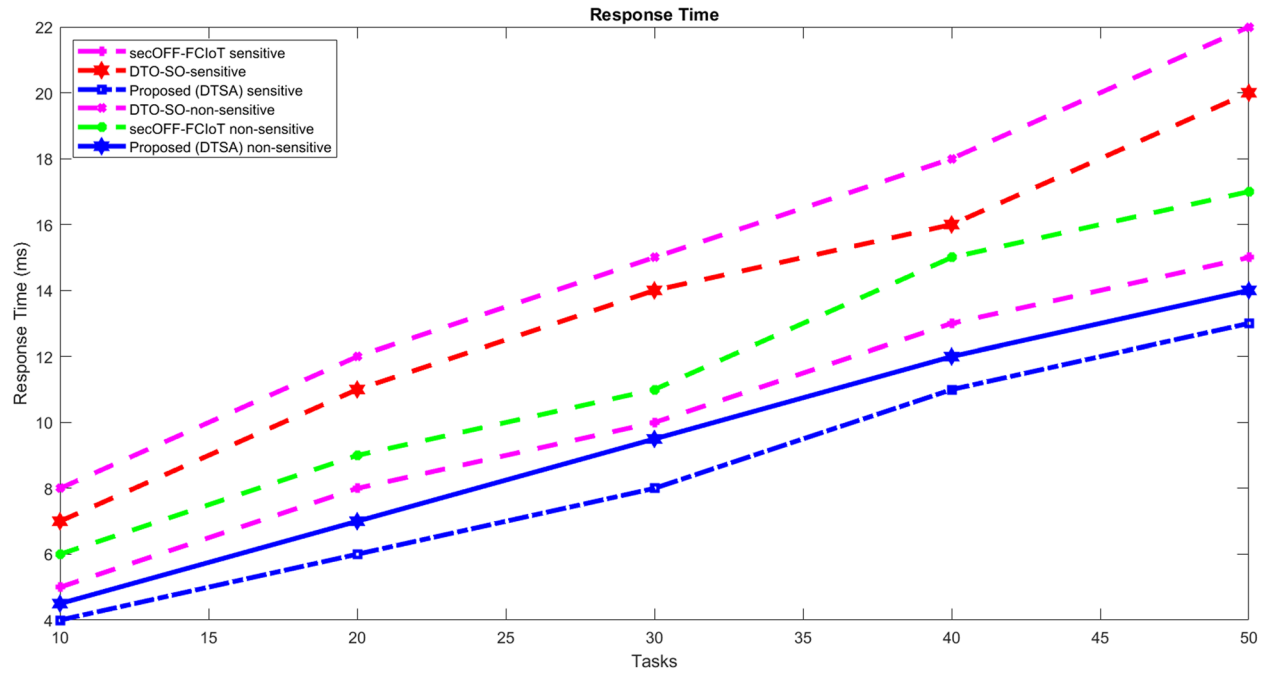

Fig. 9 Response time

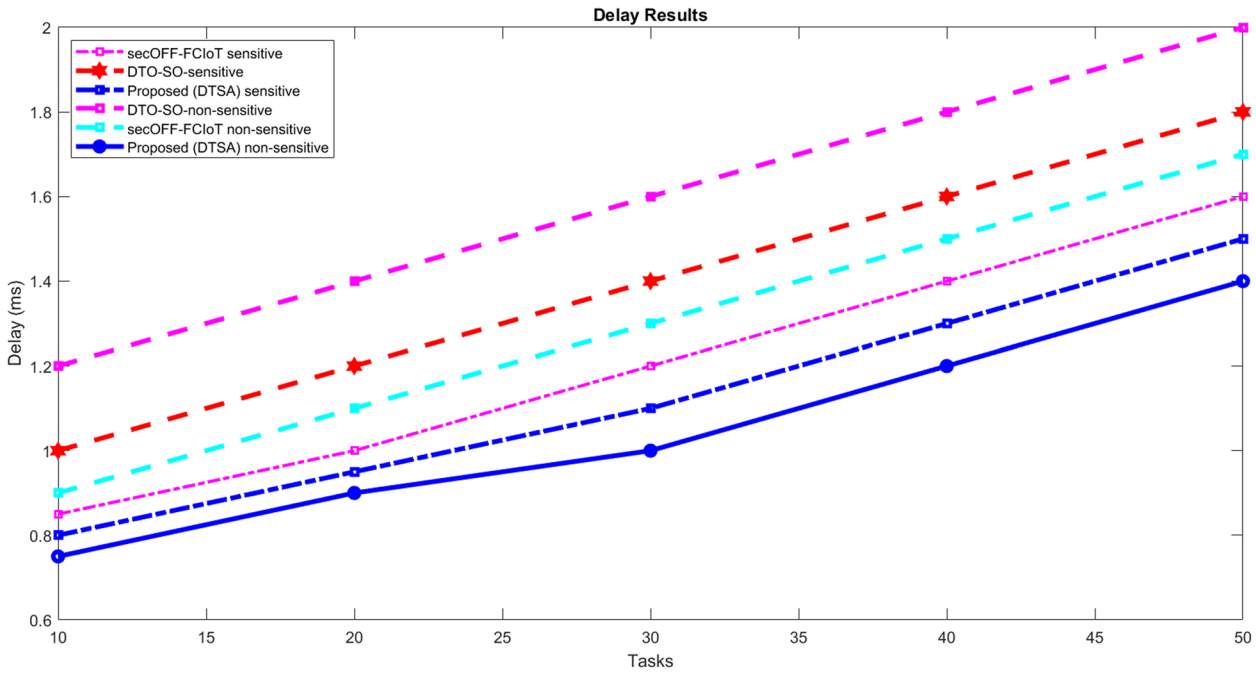

Fig. 10 Delay

scheme increased the number of offloaded tasks to about 53% on average. This increase in the number of offloaded tasks increased the duration of the mobile device's battery life since more tasks are offloaded.

Throughput

The throughput increases with the increase in the number of simulated tasks (n) as shown in Fig. 12. For instance, at $n = 20$, the throughput of our proposed scheme is 55 kb/s for sensitive tasks as against 44 kb/s and 35 kb/s for secOFF-FCIoT and DTO-SO. At $n = 30$, the throughput of our proposed scheme for non-sensitive tasks is 64 kb/s as against 60 kb/s and 43 kb/s for secOFF-FCIoT and DTO-SO. At $n = 50$, the throughput attained by our proposed dynamic offloading scheme is 130 kb/s and 110 kb/s for the sensitive and non-sensitive tasks as against throughput for secOFF-FCIoT which is 120 and 99 for sensitive and non-sensitive tasks. Therefore, our proposed offloading scheme is better compared with the existing schemes.

The increase in the throughput is a result of the effect of the attribute reduction method applied on the offloadable tasks in addition to the enhanced hybrid GA-PSO which is applied for optimal device selection.

Energy consumption

Figure 13 shows the impact on the energy consumption of IoT devices. Energy is one of the most important constraints in IoT applications. Executing complex tasks is always at the expense of the battery life of IoT devices. Offloading intensive tasks that are computationally intensive to fog and cloud saves energy for the IoT devices. Our proposed offloading scheme achieved reduced energy consumption at the IoT devices since more tasks are offloaded to fog and cloud. The delay is reduced because of the reduced data size from our attribute reduction method. This results in reduced energy consumption. At $n = 40$, for instance, our proposed offloading scheme energy consumption is 0.094 J as against 0.1 J and 0.13 J for secOFF-FCIoT and DTO-SO on sensitive tasks. While at the same $n = 40$, for non-sensitive tasks, our proposed scheme energy consumption is 0.097 J as against 0.104 J and 0.14 J for secOFF-FCIoT and DTO-SO.

Resource utilization

The impact on resource utilization is presented in Fig. 14. Resource utilization is a very important metric in IoT applications. Resource utilization is important for maintaining high productivity in the network. It ensures that resources are not underutilized or over-utilized by workloads. Our proposed scheme achieved better resource utilization compared with the existing approaches. For instance, at $n = 20$, our proposed system achieved 95% as against 93% and 90% for secOFF-FCIoT and DTO-SO. Similarly, for non-sensitive tasks at $n = 30$, our proposed offloading scheme attained 93.5% as against 93% and 91% for secOFF-FCIoT and DTO-SO resource utilization.

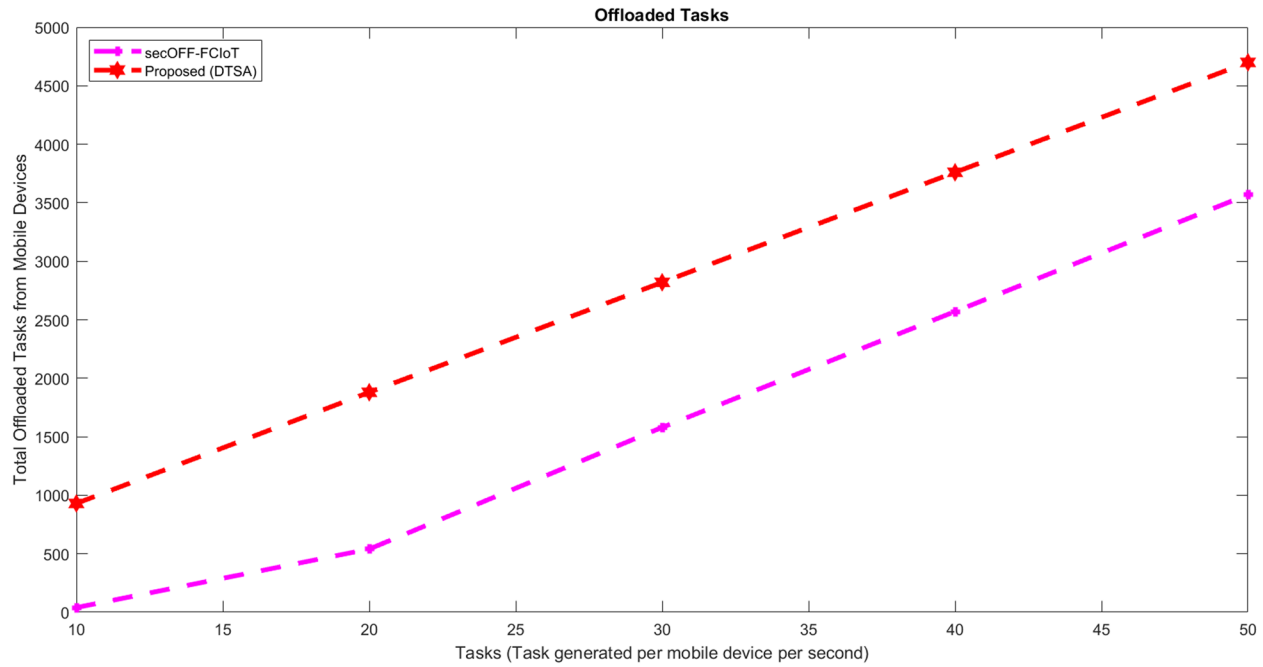

Fig. 11 Offloaded Tasks

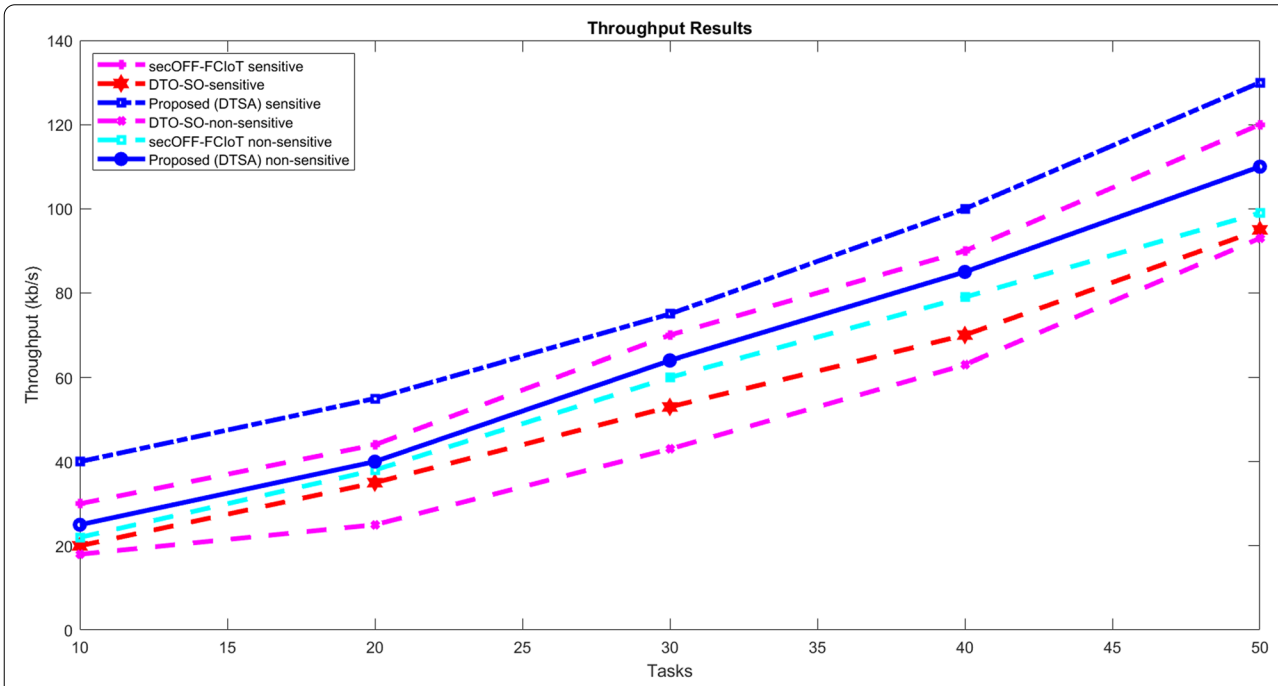

Fig. 12 Throughput

Conclusion

In this paper, we proposed a scalable dynamic offloading scheme to mainly minimize IoT requests' response time, energy consumption, and delay. The proposed three-tier system architecture for the offloading scheme is good for balancing the trade-off between response time and

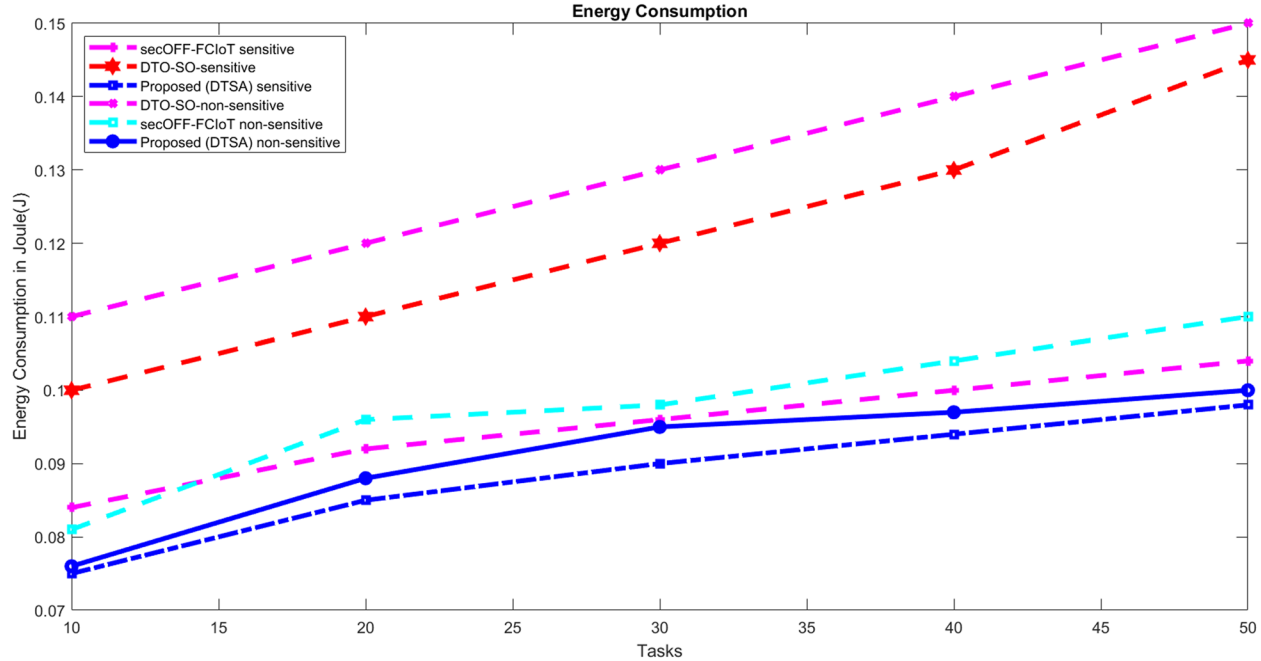

Fig. 13 Energy consumption

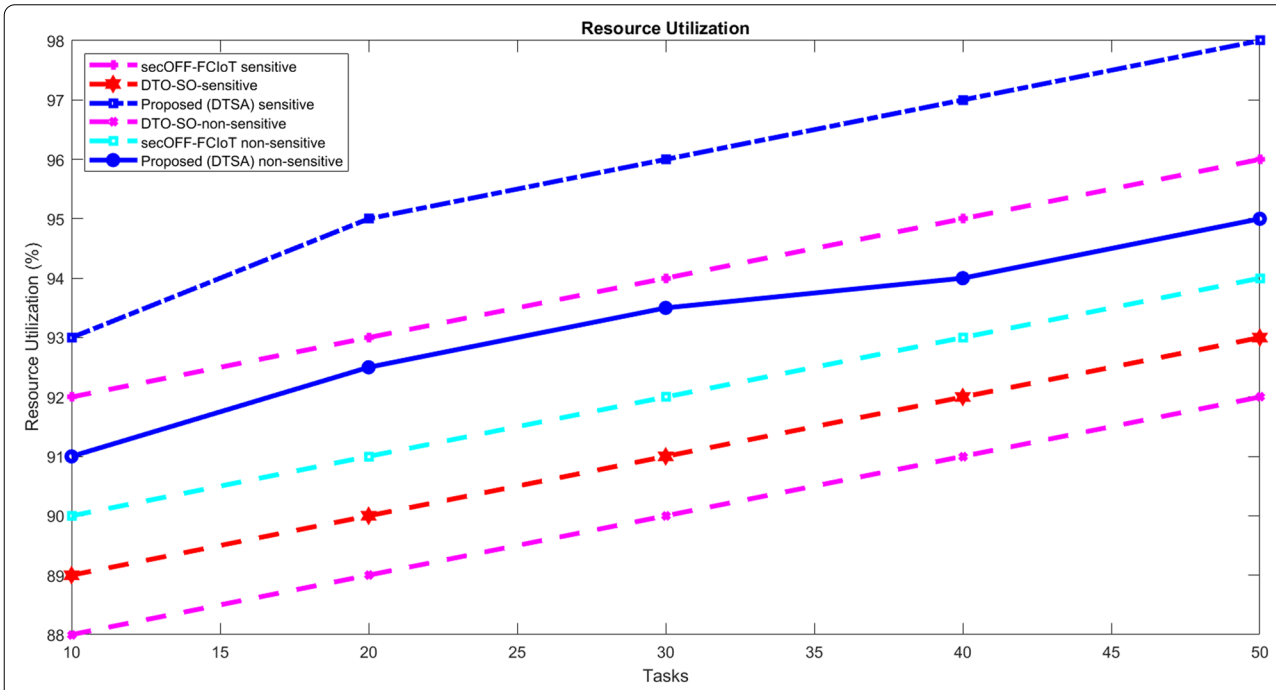

Fig. 14 Resource utilization

energy consumption. We also proposed an enhanced hybrid GA-PSO for device selection, while CPD based attribute reduction method is applied to downsize the offload-able task to minimize the delay during the task offloading process. Introducing a new technology as an attribute reduction method in task offloading enables the

offloading scheme to offload more tasks faster than the existing approaches since data size is reduced. In addition, the enhanced GA-PSO also reduces the time for searching for optimal devices and therefore makes the system more efficient and faster.

Acknowledgments
The authors would like to thank all the staff and pg students of the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia for their contribution during this research process.

Authors' contributions
Data curation, Nweso Emmanuel Nwogbaga; Conceptualization, Nweso Emmanuel Nwogbaga; Software, Nweso Emmanuel Nwogbaga; Formal analysis, Nweso Emmanuel Nwogbaga; Funding acquisition, Rohaya Binti Latip, and Amir Rizaan Abdul Rahiman; Investigation, Nweso Emmanuel Nwogbaga; Methodology, Nweso Emmanuel Nwogbaga; Project administration, Rohaya Binti Latip and Lilly Suriani Affendey; Supervision, Rohaya Binti Latip, Lilly Suriani Affendey, and Amir Rizaan Abdul Rahiman; Resources, Lilly Suriani Affendey, and Amir Rizaan Abdul Rahiman; Writing – original draft, Nweso Emmanuel Nwogbaga; Writing – review& editing, Nweso Emmanuel Nwogbaga, Rohaya Binti Latip, Lilly Suriani Affendey, and Amir Rizaan Abdul Rahiman. The author(s) read and approved the final manuscript.

Funding
This work is supported by UPM Journal Publishing Grant (Grant No: 9001103), University Putra Malaysia, and the Ministry of Education Malaysia.

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
None.

Author details
[1]Department of Communication Technology and Network, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Seri Kembangan, Malaysia. [2]Ebonyi State University, Abakaliki, Nigeria. [3]Department of Computer Science, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Seri Kembangan, Malaysia.

Received: 25 January 2022 Accepted: 21 May 2022
Published online: 04 June 2022

References

1. Abou-Nassar EM, Iliyasu AM, El-Kafrawy PM, Song OY, Bashir AK, El-Latif AAA (2020) DITrust chain: towards Blockchain-based trust models for sustainable healthcare IoT systems. IEEE Access 8:111223–111238. https://doi.org/10.1109/ACCESS.2020.2999468
2. El-Latif AAA, Abd-El-Atty B, Mazurczyk W, Fung C, Venegas-Andraca SE (2020) Secure data encryption based on quantum walks for 5G internet of things scenario. IEEE Trans Netw Serv Manag 17(1):118–131. https://doi.org/10.1109/TNSM.2020.2969863
3. Liu Y, Peng J, Kang J, Iliyasu AM, Niyato D, El-Latif AAA (2020) A secure federated learning framework for 5G networks. IEEE Wirel Commun 27(4):24–31. https://doi.org/10.1109/MWC.01.1900525
4. Elgendy IA, Zhang WZ, He H, Gupta BB, Abd El-Latif AA (2021) Joint computation offloading and task caching for multi-user and multi-task MEC systems: reinforcement learning-based algorithms. Wirel Netw 27(3):2023–2038. https://doi.org/10.1007/s11276-021-02554-w
5. Farahbakhsh F, Shahidinejad A, Ghobaei-Arani M (2021a) Context-aware computation offloading for mobile edge computing. J Ambient Intell Humaniz Comput 0123456789. https://doi.org/10.1007/s12652-021-03030-1
6. Farahbakhsh F, Shahidinejad A, Ghobaei-Arani M (2021b) Multiuser context-aware computation offloading in mobile edge computing based on Bayesian learning automata. Trans Emerg Telecommun Technol 32(1):1–26. https://doi.org/10.1002/ett.4127
7. Shakarami A, Shahidinejad A, Ghobaei-Arani M (2021) An autonomous computation offloading strategy in Mobile edge computing: a deep learning-based hybrid approach. J Netw Comput Appl 178. https://doi.org/10.1016/j.jnca.2021.102974
8. Zhang Q, Gui L, Zhu S, Lang X (2021) Task offloading and resource scheduling in hybrid edge-cloud networks. IEEE Access 9:85350–85366. https://doi.org/10.1109/access.2021.3088124
9. Nwogbaga NE, Emewu BM, Ogbaga IN (2016) Critical analysis of cloud computing and its advantages over other computing techniques. J Multidiscip Eng Sci Technol 3(2):3955–3960
10. Alshahrani A, Elgendy IA, Muthanna A, Alghamdi AM, Alshamrani A (2020) Efficient multi-player computation offloading for VR edge-cloud computing systems. Appl Sci 10(16):1–19. https://doi.org/10.3390/app10165515
11. Huynh LNT, Pham QV, Pham XQ, Nguyen TDT, Hossain MD, Huh EN (2020) Efficient computation offloading in multi-tier multi-access edge computing systems: a particle swarm optimization approach. Appl Sci 10(1):1–17. https://doi.org/10.3390/app10010203
12. Khan PW, Abbas K, Shaiba H, Muthanna A, Abuarqoub A, Khayyat M (2020) Energy-efficient computation offloading mechanism in multi-server mobile edge computing—an integer linear optimization approach. Electronics (Switzerland) 9(6):1–20. https://doi.org/10.3390/electronics9061010
13. Liu J, Lian X, Liu C (2021) Research on task-oriented computation offloading decision in a space-air-ground integrated network. Future Internet 13(5). https://doi.org/10.3390/fi13050128
14. Fang J, Shi J, Lu S, Zhang M, Ye Z (2021) An efficient computation offloading strategy with mobile edge computing for IoT. Micromachines 12(2). https://doi.org/10.3390/mi12020204
15. Huang M, Zhai Q, Chen Y, Feng S, Shu F (2021) Multi-objective whale optimization algorithm for computation offloading optimization in mobile edge computing. Sensors 21(8):1–24. https://doi.org/10.3390/s21082628
16. Alli AA, Alam MM (2019) SecOFF-FCIoT: machine learning-based secure offloading in fog-cloud of things for smart city applications. Internet Things 7(2019):100070. https://doi.org/10.1016/j.iot.2019.100070
17. Li L, Wen X, Lu Z, Jing W (2020) An energy-efficient design of computation offloading enabled by UAV. Sensors (Switzerland) 20(12):1–19. https://doi.org/10.3390/s20123363
18. Wei D, Xi N, Ma J, He L (2021) UAV-assisted privacy-preserving online computation offloading for internet of things. Remote Sens 13:1–18
19. Wu H, Sun Y, Wolter K (2018) Energy-efficient decision making for Mobile cloud offloading. IEEE Transact Cloud Comput 7161(2). https://doi.org/10.1109/TCC.2018.2789446
20. Ismail L, Materwala H (2021) Escove: energy-SLA-aware edge–cloud computation offloading in vehicular networks. Sensors 21(15):1–20. https://doi.org/10.3390/s21155233
21. Koubaa A, Ammar A, Alahdab M, Kanhouch A, Azar AT (2020) Deep brain: experimental evaluation of cloud-based computation offloading and edge computing in the internet-of-drones for deep learning applications. Sensors (Switzerland) 20(18):1–25. https://doi.org/10.3390/s20185240
22. Gnana Jeevan AN, Maluk Mohamed MA (2018) DyTO: dynamic task offloading strategy for Mobile cloud computing using surrogate object model. Int J Parallel Prog 48(3):399–415. https://doi.org/10.1007/s10766-018-0563-0
23. Guo S, Xiao B, Yang Y, Yang Y (2016) Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. In: Proceedings - IEEE INFOCOM, 2016-July. https://doi.org/10.1109/INFOCOM.2016.7524497

24. Estlin TA, Mooney RJ (1997) Learning to improve both efficiency and quality of planning. In: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, pp 1227–1232
25. Kumar K, Liu J, Lu YH, Bhargava B (2013) A survey of computation offloading for mobile systems. Mobile Netw Appl 18(1):129–140. https://doi.org/10.1007/s11036-012-0368-0
26. Li H, Ota K, Dong M (2018) Learning IoT in edge: deep learning for the internet of things with edge computing. IEEE Netw 32(1):96–101. https://doi.org/10.1109/MNET.2018.1700202
27. Zhang J, Zhou Z, Li S, Gan L, Zhang X, Qi L et al (2017) Hybrid computation offloading for smart home automation in mobile cloud computing. Pers Ubiquit Comput
28. Katoch S, Chauhan SS, Kumar V (2021) A review on the genetic algorithm: past, present, and future. In: Multimedia Tools and Applications, vol 80. https://doi.org/10.1007/s11042-020-10139-6
29. Babanezhad M, Behroyan I, Nakhjiri AT, Marjani A, Rezakazemi M, Heydarinasab A, Shirazian S (2021) Investigation of the performance of particle swarm optimization (PSO) algorithm-based fuzzy inference system (PSOFIS) in a combination of CFD modeling for prediction of fluid flow. Sci Rep 11(1):1–14. https://doi.org/10.1038/s41598-021-81111-z
30. Jahandideh-Tehrani M, Jenkins G, Helfer F (2021) A comparison of particle swarm optimization and genetic algorithm for daily rainfall-runoff modeling: a case study for Southeast Queensland, Australia. Optimization Engin 22(1):29–50. https://doi.org/10.1007/s11081-020-09538-3
31. Manasrah AM, Ali HB (2018) Workflow scheduling using hybrid GA-PSO algorithm in cloud computing. Wirel Commun Mob Comput 2018. https://doi.org/10.1155/2018/1934784
32. Flores H, Su X, Kostakos V, Ding AY, Nurmi P, Tarkoma S et al (2017) Large-scale offloading in the internet of things. In: 2017 IEEE international conference on pervasive computing and communications workshops, PerCom workshops 2017, pp 479–484. https://doi.org/10.1109/PERCOMW.2017.7917610
33. Peng G, Wu H, Wu H, Wolter K (2021) Constrained multi-objective optimization for IoT-enabled computation offloading in collaborative edge and cloud computing. IEEE Internet Things J. https://doi.org/10.1109/JIOT.2021.3067732
34. Ma X, Lin C, Zhang H, Liu J (2018) Energy-aware computation offloading of IoT sensors in cloudlet-based mobile edge computing. Sensors (Switzerland) 18(6):1–12. https://doi.org/10.3390/s18061945
35. Min M, Xiao L, Chen Y, Cheng P, Wu D, Zhuang W (2019) Learning-based computation offloading for IoT devices with energy harvesting. IEEE Trans Veh Technol 68(2):1930–1941. https://doi.org/10.1109/TVT.2018.2890685
36. Van Le D, Tham C (2018) A deep reinforcement learning based offloading scheme in ad-hoc Mobile clouds. In: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp 760–765
37. Zhang L, Chen Y, Sun R, Jing S, Yang B (2008) A task scheduling algorithm based on PSO for grid computing. Int J Comput Intell Res 4(1). https://doi.org/10.5019/j.ijcir.2008.123
38. Yang L, Zhong C, Yang Q, Zou W, Fathalla A (2020) Task offloading for directed acyclic graph applications based on edge computing in industrial internet. Inf Sci 540:51–68. https://doi.org/10.1016/j.ins.2020.06.001
39. Dastjerdi AV, Buyya R (2016) Fog computing: helping the internet of things realize its potential. Computer 49(8):112–116. https://doi.org/10.1109/MC.2016.245
40. Varghese B, Wang N, Barbhuiya S, Kilpatrick P, Nikolopoulos DS (2016) Challenges and opportunities in edge computing. In: Proceedings - 2016 IEEE International Conference on Smart Cloud, SmartCloud 2016, pp 20–26. https://doi.org/10.1109/SmartCloud.2016.18
41. Atlam HF, Walters RJ, Wills GB (2018) Fog computing and the internet of things: a review. Big Data Cogn Comput 2(2):1–18. https://doi.org/10.3390/bdcc2020010
42. Bonomi F, Milito R, Zhu J, Addepalli S (2012) Fog computing and its role in the internet of things. In: MCC'12 - proceedings of the 1st ACM Mobile cloud computing workshop, pp 13–15. https://doi.org/10.1145/2342509.2342513
43. Kumari S, Singh S, April M (2017) Fog computing: characteristics and challenges. 6(2):113–117
44. Al-Fuqaha A, Guizani M, Mohammadi M, Aledhari M, Ayyash M (2015) Internet of things: a survey on enabling technologies, protocols, and applications. IEEE Commun Surv Tutorials 17(4):2347–2376. https://doi.org/10.1109/COMST.2015.2444095
45. Luan TH, Gao L, Li Z, Xiang Y, Wei G, Sun L (2016) Fog computing: focusing on Mobile users at the edge, pp 1–11 Retrieved from http://arxiv.org/abs/1502.01815
46. Mahmud R, Kotagiri R, Buyya R (2018) Fog computing: a taxonomy, survey and future directions, pp 103–130. https://doi.org/10.1007/978-981-10-5861-5_5
47. Gubbi J, Buyya R, Marusic S, Palaniswami M (2013) Internet of things (IoT): a vision, architectural elements, and future directions. Futur Gener Comput Syst 29(7):1645–1660. https://doi.org/10.1016/j.future.2013.01.010
48. Han P, Du C, Chen J, Ling F, Du X (2021) Cost and makespan scheduling of workflows in clouds using list multiobjective optimization technique. J Syst Archit 112. https://doi.org/10.1016/j.sysarc.2020.101837
49. Chen J, Du T, Xiao G (2021) Multi-objective optimization for resource allocation of emergent demands in cloud computing. J Cloud Comput 10(1). https://doi.org/10.1186/s13677-021-00237-7
50. Samanataray S, Sahoo A (2021) A comparative study on prediction of monthly Streamflow using hybrid ANFIS-PSO approaches. KSCE J Civ Eng. https://doi.org/10.1007/s12205-021-2223-y
51. Song T, Pu H, Schonfeld P, Zhang H, Li W, Hu J et al (2021) Bi-objective mountain railway alignment optimization incorporating seismic risk assessment. Comput Aided Civ Infrastruct Engin 36(2):143–163. https://doi.org/10.1111/mice.12607
52. Eberhart RC, Shi Y (2001) Particle swarm optimization: developments, applications, and resources. In: Proceedings of the IEEE Conference on Evolutionary Computation, ICEC, 1(February 2001), pp 81–86. https://doi.org/10.1109/cec.2001.934374
53. Nwogbaga NE, Latip R, Affendey LS, Rizaan ARA (2021) Investigation into the effect of data reduction in off loadable task for distributed IoT-fog-cloud computing. J Cloud Comput 10:1–2
54. Gupta H, Dastjerdi AV, Ghosh SK, Buyya R (2016) iFogSim : a toolkit for modeling and simulation of resource management techniques in the internet of things. Edge Fog:1–22

Publisher's Note