This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2958400, IEEE Internet of Things Journal

Application Offloading Strategy for Hierarchical Fog Environment through Swarm Optimization

Mainak Adhikari[1], Satish Narayana Srirama[1,*], *Senior Member, IEEE,* and Tarachand Amgoth[2]

***Abstract*—Nowadays, billions of Internet of Things (IoT) devices generate various types of delay-sensitive tasks to process within a limited time frame. By processing the tasks at the network edge using distributed fog devices can efficiently overcome the deficiency of the centralized cloud data center (CDC), *i.e.* long latency and network congestion. Moreover, to overcome the inefficiency of the local fog devices, *i.e.* limited processing and storage capabilities, we investigate the collaboration between distributed fog devices and centralized CDC, where the delay-sensitive tasks can preferably be offloaded on the local fog devices, whereas the resource-intensive tasks are offloaded on the resource-rich CDC. However, one of the challenging task in the fog-cloud environment is to find a suitable computing device for each real-time task by considering trade-off between the latency and cost. To meet the above-mentioned challenge, in this paper, we introduce an optimal application offloading strategy in hierarchical fog-cloud environment using accelerated particle swarm optimization (APSO) technique. The proposed APSO-based strategy finds an optimal computing device (*i.e.* fog device or cloud server) for each real-time task using multiple Quality-of-Service (QoS) parameters, namely cost and resource utilization. The performance of the proposed algorithm is evaluated using four different real-time data sets with various performance matrices. The experimental results indicate that the proposed strategy outperforms the existing schemes in terms of average delay, computation time, resource utilization and average cost by 18%, 21%, 27%, and 23%, respectively.**

***Index Terms*—IoT; Multi-objective optimization; Fog computing; Delay-sensitive application; Application offloading; APSO technique.**

I. Introduction

With the emergence of the Internet of Things (IoT), billions of heterogeneous physical objects are connected through a network for collecting and sharing information, which can improve various aspects of daily lives including smart transportation, smart grid, smart city, smart home, smart agriculture, smart water, and waste management, etc. [1]. The main objectives of the IoT applications are to minimize the latency and processing time while utilizing the computing resources efficiently. Most of the physical devices in IoT are embedded with sensors, and participate in applications, which require processing data locally or offload to a suitable computing device (*i.e.* distributed fog device or centralized cloud data center (CDC)) [2]. Conventionally, most of these applications are offloaded to the CDC which has enough resources for processing and storage. However, the dramatic growth of the IoT applications and sensors across the globe lead to huge amounts of real-time data which is to be transferred to the centralized CDC. The Cisco Global Cloud Index estimates that the data produced by various physical objects and machines would be 500 zetta bytes within 2019; however, the global capacity to transmit the data to the centralized CDC is at most 10.4 zetta bytes [3]. As a result, offloading the applications to the CDC can increase the burden of the network which causes network congestion and increases the latency [4].

To tackle the above-mentioned challenges, there is an emerging computing paradigm, known as fog computing, which offers the cloud services at the edge of the network with limited processing and storage capacity [5]. Fog Computing has drawn extensive attention of the researchers for processing delay-sensitive tasks locally [6]. With the help of the fog computing, the real-time tasks are offloaded to the local fog devices for getting better services with minimum latency. The most popular and useful fog devices are Raspberry Pi, network switches, routers and micro data centres etc. However, due to the limited resource capacity, most of the resource-intensive applications do not fit for processing in the fog devices and are still offloaded to the CDC. However, offloading the data to the fog devices incur minimum latency and cost as compared with offloading to the centralized CDC [7]. Thus, the intuitive idea is to offload the delay-sensitive real-time tasks to the local fog devices that can meet the resource requirements of the tasks and achieve the trade-off between cost and latency [8].

The IoT devices should not only check the resource availability and the workload of a fog device, but also check the latency, and cost for processing and transmitting an application. Consequently, the selection of a fog device is a multi-objective optimization problem (MOP). MOP is used to find an optimal solution using more than one optimization function [9], [10]. However, designing an efficient algorithm for finding an optimal fog device using MOP strategy remains challenging and has not been investigated widely. Hence, the main purpose of this paper is to design an efficient algorithm for finding a suitable fog device using multiple Quality-of-Service (QoS) objectives in a dynamic IoT environment. Swarm intelligence (SI) and evolutionary computation are shown to be powerful methods for solving such a MOP efficiently [11]. Among many others, accelerated particle swarm optimization (APSO) is very attractive SI technique which can solve the MOP more efficiently [12]. APSO is the modified version of particle swarm optimization (PSO) technique. The standard PSO technique uses the best of the individuals and current global

* indicates the corresponding author.

[1] Mobile & Cloud Lab, Institute of Computer Science, University of Tartu, Estonia.

[2]Department of Computer Science and Engineering, Indian Institute of Technology (Indian School of Mines), Dhanbad, Jharkhand, India, 826004.

e-mails: mainak@ut.ee, satish.srirama@ut.ee, and tarachand@iitism.ac.in.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2958400, IEEE Internet of Things Journal

best individual for finding an optimal solution; however, APSO technique uses the current global best individual for finding an optimal result which should increase the convergence speed of the particles and reduces the randomness as the iteration proceeds.

In this paper, we have designed a new multi-objective offloading (MOO) strategy for hierarchical fog-cloud environment using APSO technique. The contribution of the work is two folded; 1) Design a multi-objective optimization function with a weighted-sum approach using two QoS parameters, namely resource utilization, and total cost; 2) Next, finds an optimal computing device using the multiple QoS objectives using APSO technique with higher convergence speed. The main goals of the work are to minimize the total cost and latency while utilizing the computing resources efficiently. We later evaluated the approach through multiple real-time data sets. The data sets are generated from a real-time fog-cloud environment where we have used mobile, tablet, and Raspberry Pi as fog devices and Amazon Web Services as the centralized CDC. The major contributions of this work are summarized as follows.

- Developed a multi-tier fog-cloud model for offloading the real-time tasks to the suitable computing devices as per their resource requirements.
- Designed a MOP function, also known as fitness function based on the multiple QoS objectives of the real-time IoT applications, namely resource utilization and total cost.
- We further introduced an efficient task offloading strategy on the multi-tier fog-cloud model using APSO technique for improving the QoS parameters in terms of latency, computation time, cost and resource utilization.
- Finally, we investigated the parameters of APSO technique over multiple real-time data sets and fixed their parametric values for further analyzing the proposed MOO strategy. Moreover, we analyzed the performance of the proposed method over three state-of-the-art algorithms using various performance matrices.

The rest of the paper is organized as follows. Related work of existing offloading methodologies in fog computing with MOP strategy is discussed in Section II. The technical overview of the multi-objective optimization along with the APSO technique is discussed in Section III. The system model of the work followed by the problem formulation is discussed in Section IV. The proposed MOO strategy is discussed in Section V. The performance analysis of the proposed strategy is discussed in Section VI. Finally, the paper is concluded in Section VII.

II. Related Work

Application offloading strategy on fog-cloud environment has attracted significant attention by researchers [13], [14] and IT industry [15]. Liang *et. al.* have designed a Software-defined networking based fog-cloud architecture for minimizing the latency [16]. Zhao *et. al.* have proposed an IoT application offloading strategy for minimizing the computation time and energy consumption of the computing resources [17]. Fricker *et. al.* have designed an optimal offloading strategy for balancing the loads among the active fog devices [18]. Zhang *et. al.* proposed an application-based offloading strategy for minimizing the latency and energy consumption of the fog devices [19]. Hasan *et. al.* [20] and Wang *et. al.* [21] also designed efficient offloading strategies for minimizing the energy usage and the processing time of the applications. Tan *et. al.* have designed a delay-sensitive offloading strategy for minimizing the overall computation time of the applications [22]. Similarly, He *et. al.* have developed a multi-tier model for offloading the applications efficiently with minimum latency [23]. Mansour *et. al.* have proposed an energy efficient offloading algorithm for minimizing the energy and computation time of the applications while processing [24]. Mahmud *et. al.* have designed a proft-aware application offloading strategy in fog environment, meeting multiple QoS parameters for real-time applications [25]. Adhikari *et. al.* have designed a priority-aware data offloading and scheduling strategy in a hierarchical fog-cloud environment [26]. The main goals of this work are to minimize the waiting time and latency of the tasks while meeting their deadline. Brogi *et. al.* have developed a single objective cost-aware data offloading strategy with genetic algorithm in fog environment [27]. The main objectives of the work are to minimize the computation and communication cost of the real-time tasks with efficient resource utilization. Most of the above-mentioned strategies have used heuristics or dynamic algorithms for improving a single QoS parameter.

Nowadays, few initiatives are tailored for optimizing multiple scheduling parameters of IoT applications using different swarm and evolutionary algorithms [28]. Jain *et. al.* [29] have designed a multi-objective offloading strategy in edge-cloud environment for minimizing the overall processing time and cost of the applications. Sharif *et. al.* [30] have developed an optimal resource allocation strategy in fog-cloud environment for efficient resource utilization. Bitam *et. al.* [31] have designed a multi-objective scheduling strategy in fog environment using Bee Swarm algorithm for minimizing the computation time with efficient resource utilization. Shi *et. al.* [32] have proposed an online offloading strategy for selecting the suitable computing devices for the real-time tasks to minimize the transmission and processing time in fog-cloud domain. However, the above-mentioned papers did not consider the two important offloading objectives of the real-time tasks, namely transmitting and processing cost and the resource utilization.

Similarly, Aryal *et. al.* have designed a genetic algorithm (GA) based offloading strategy for optimizing multiple objectives of the real-time applications [33]. The main purpose of this work is to minimize the total cost with efficient resource utilization of the computing resources. Wan *et. al.* have developed a Particle Swarm Optimization (PSO)-aware application offloading strategy in fog environment [34]. The main purpose of this work is to optimize the energy consumption of the overall network with minimum latency. Ezhilarasie *et. al.* have proposed a GA-PSO based optimal offloading strategy in fog computing [35]. The main goal of this work is to minimize the latency and computation time while meeting the deadline of the tasks.

From the review of the related work, it is found that most

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2958400, IEEE Internet of Things Journal

of the single objective optimization strategies have minimized the latency or energy consumption rate of the computing devices with a heuristic or meta-heuristic technique. However, in a complex distributed environment such as in fog computing, the offloading strategy needs to optimize multiple QoS parameters for improving the efficiency of the model. This lead to the multi-objective solutions, as discussed above. The current work complements the state-of-the-art multi-objective offloading strategies by applying APSO technique for solving the MOP, with significant focus at accuracy and efficiency of the environment. Thus, we have developed a multi-objective offloading strategy in a multi-tier fog-cloud environment that meets the multiple QoS parameters of the real-time tasks such as resource utilization and cost. Moreover, the proposed strategy deploys the tasks to the suitable computing devices that can minimize the overall latency and computation time.

III. Technical Overview of the MOP Strategy and the APSO Technique

In this section, we discuss the overview of the MOP strategy followed by the APSO technique in brief.

A. Overview of MOP

Nowadays, most of the real-world problems in a distributed environment can be formulated mathematically with certain goals or objectives which are optimized by various heuristic or meta-heuristic strategies while meeting relevant constraints. If there is more than one objective to optimize a problem, such a problem is called MOP which is formulated as follows.

Minimize $\mathbf{F(x,\ t)} = f_1(x,t), f_2(x,t), f_3(x,t), , f_n(x,t)^T$

Subject to:

$g_j(x) \leq 0 : j = 1, 2, 3, ..., v$

$h_i(x) = 0 : i = 1, 2, 3, ..., u$

$\mathbf{x} \in \theta$

Here, $x = (x_1, x_2, x_3, x_4, ..., x_m)^T$, is a m-dimensional decision vector in the decision space θ; $F : \theta \rightarrow \phi \subseteq R^n$ is an objective vector which consists of n objective functions and maps m-dimension vector space from θ to n-dimensional object space. $\phi : g_j(x) \leq 0$ and $h_i(x) = 0$ represent inequality and equality constraints respectively. In general, the objectives of MOP are conflicting, thus a single best solution of all of them may not exist. Hence, the researchers need to use Pareto Optimality (PO) to resolve such problem. In a PO solution, x_1 dominates a solution x_2, *i.e.* $(x_1 \preceq x_2)$ if and only if x_1 is better than x_2 in at least one objective, which is formulated as follows.

$$\begin{array}{rcl} F(x) & = & \forall j = 1, ..., n, \{f_j(x_1) \preceq f_j(x_2)\} \\ F(x) & = & \exists j = 1, ..., n, \{f_j(x_1) \prec f_j(x_2)\} \end{array} \quad (1)$$

Let us consider x_1 and x_2 are two decision vectors. The decision vector x_2 is said to be non-dominated if and only if there is no other decision vector x_1 such that $x_1 \succ x_2$. The Pareto-Optimal Set (PS) consists of the set of all PO solutions which is defined as follows.

$$PS = \{x_2 \mid \exists x_1 : x_1 \succ x_2\} \quad (2)$$

The Pareto-optimal Front (PF) is the corresponding objective function which is represented as follows.

$$PF = F\{x_2 \mid x_2 \in PS\} \quad (3)$$

It is important to solve such real-world MOPs with or without constraints as accurate as possible. However, due to the unavailability of the computing resources with different constraints, the true PF can hardly produce an accurate result. Thus, the meta-heuristic strategies are used to find the non-dominated solution set as accurate as possible to approximate the true PF.

B. Overview of APSO Technique

PSO technique is designed based on the behavior of bird or fish schooling in nature [36]. The PSO technique searches the space of an objective function by adjusting the position of the particles in a quasi-stochastic manner. Each particle has been represented by its own velocity and position. A particle i keeps track of its best position in space by P_{best} which is represented as $(X_i^*) = \{X_1^*, X_2^*, X_3^*, ..., X_n^*\}$. The global best position among all P_{best} is represented as g_{best} (G^*). The velocity is defined as the movement of a particle which has its own magnitude and direction, which is defined as follows.

$$v_i(t+1) = v_i(t) + \alpha\varepsilon_1(G^* - X_i(t)) + \beta\varepsilon_2(X^*(t) - X_i(t)) \quad (4)$$

The position of a particle of PSO technique is represented as follows.

$$X_i(t+1) = X_i(t) + v_i(t+1) \quad (5)$$

Here, v_i and X_i represent velocity and position vector of the i^{th} particle respectively. ε_1 and ε_2 are two random uniform variables whose values are in the interval [0,1]. The standard PSO technique uses both the current g_{best} location and individual best location p_{best} for finding the optimal position. The p_{best} location is useful for PSO to increase the diversity and quality of the solution. However, this diversity should increase the randomness of the problem and there is no compelling to use the individual best unless the optimization problem is highly multimodal or nonlinear. Thus, the simplified version of the PSO technique is to use the g_{best} location of a particle to accelerate the convergence speed of the problem which is called as APSO technique [12]. The velocity of particle i of APSO technique is formulated as follows.

$$v_i(t+1) = v_i(t) + \alpha\varepsilon + \beta(X^* - X_i(t)) \quad (6)$$

Here ε is a random vector whose value lies between [0, 1]. The updated position of a particle is simply represented by Eq. (7). However, to increase the convergence of the particle even further, the updated position of the particle in a single step is represented as follows.

$$X_i(t+1) = (1-\beta)X_i(t) + \beta G^* + \alpha\epsilon^t \quad (7)$$

IV. System Model

In this section, we elaborate the multi-tier fog-cloud model followed by the execution model in details. Finally, we discuss the problem statement for this work.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2958400, IEEE Internet of Things Journal

A. Multi-tier Fog-Cloud Model

The proposed multi-tier fog-cloud model is shown in Fig. 1, which is composed of four tiers, *i.e.*, the IoT device layer, Low-capacity fog (LCF) device layer, High-capacity fog (HCF) device layer, and centralized CDC layer. The IoT layer consists of a set of real-time IoT devices and sensors, which can generate various types real-time applications for processing. The IoT devices have minimum processing and storage capacity. The delay-sensitive IoT applications need to deploy the real-time tasks on the local fog devices with enough resource capacity that can process the task efficiently with minimum latency. In this work, we classify the fog devices into two levels based on their resource capacity, *i.e.* LCF devices and HCF devices. Here, we assume that the LCF devices including mobile devices, Tabs, Raspberry Pi, etc. have limited processing capacity and are deployed nearer to the IoT devices. The LCF devices are deployed on Tier 2 of the model and minimize the latency of the tasks with minimum processing and transmission cost. The HCF devices including desktop, workstations, private cloud of the educational institute or industries have more processing capacity with higher latency as compared with LCF devices. The HCF devices are deployed in Tier 3 and have enough processing capacity for processing the tasks with minimum latency and cost, when compared with CDC. The fog devices of Tier 2 and Tier 3 are represented as $LF = \{LF_1, LF_2, LF_3, ..., LF_n\}$ and $HF = \{HF_1, HF_2, HF_3, ..., HF_m\}$ respectively. Finally, the Tier 4 consists of the centralized CDC which has enough resources for processing any type of resource-intensive applications.

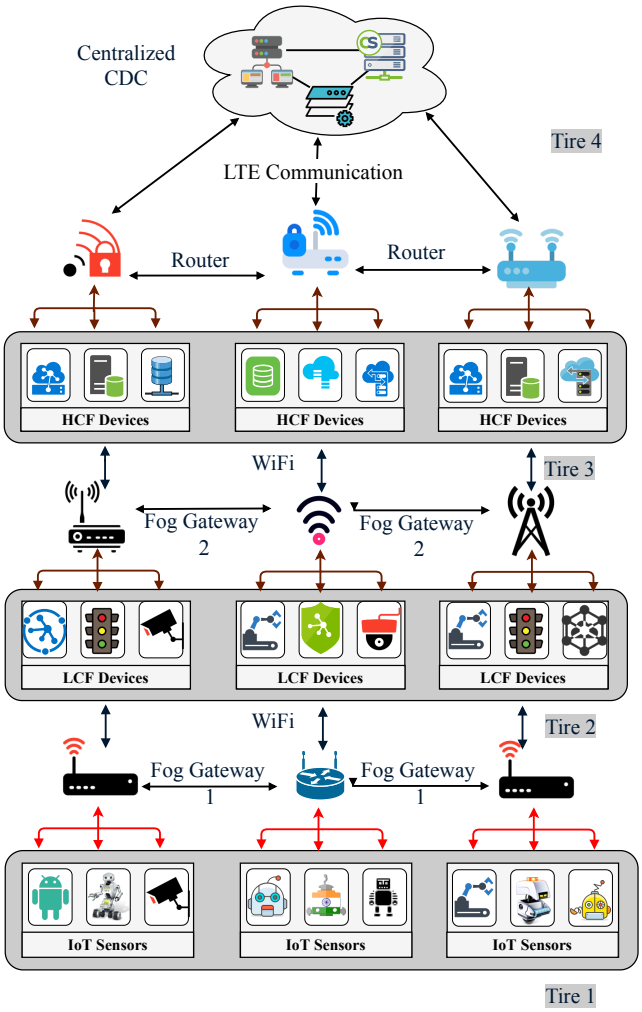


Fig. 1: Multi-tier fog-cloud Model

B. Execution Model

Let us consider that there are N IoT devices denoted by the set $I = \{I_1, I_2, I_3, ..., I_N\}$. Each IoT device $I_N \in I$ generates a real-time task T_n for further processing. The main focus of this work is to find an optimal computing device for each task based on the multiple QoS parameters. Here, we focus on two primary objectives, namely total cost and resource utilization of the computing devices while processing the tasks, which are discussed below.

1) Resource Utilization Model: Given a set of fog devices $F = \{F_1, F_2, F_3, ..., F_f\}$, each element $F_i \in F, 1 \leq i \leq f$ represents a fog device in Tier 2 or Tier 3 based on its resource capacity. The resource capacity of each fog device is represented as $U_i \in (U_i^C, U_i^{mm})$, where U_i^C denotes the maximum CPU capacity (in a unit of CPU hours) and U_i^{mm} represents maximum memory usage (in the unit of GB) of fog device i. Let, the CDC consists of h number of heterogeneous cloud servers $H = \{H_1, H_2, H_3, ..., H_h\}$, where each server $H_j \in H, 1 \leq j \leq h$ can run multiple resource-intensive tasks in parallel order. Moreover, the maximum CPU and memory capacity of a cloud server j is represented as $U_j \in (U_j^C, U_j^{mm})$. The maximum workload (W_l) consumed by a computing device l within a time interval $t = (t_1 - t_2)$ is represented as follows.

$$W_l(t) = \alpha \sum_{t=t_1}^{t_2} U_l^C(t) + (1-\alpha) \sum_{t=t_1}^{t_2} U_l^{mm}(t), \forall l \in (i,j) \quad (8)$$

Here, α is a user defined constant whose value lies between the range [0, 1]. The values of U_l^C and U_l^{mm} lie in the interval [0,100]. Assume that at a certain timestamp t, k number of tasks are received by the IoT gateways for offloading to the appropriate computing devices, *i.e.* $T = \{T_1, T_2, T_3, ..., T_K\}$, where each task $T_k \in T$, $1 \leq k \leq K$ should be offloaded to a suitable computing device based on its resource requirements. Now, each task requests for specific set of resources for processing in a computing device such as $\{(R_1^C, R_1^{mm}), (R_2^C, R_2^{mm}), (R_3^C, R_3^{mm}), ..., (R_k^C, R_k^{mm})\}$ and the requested workload (L_l) of the task k is defined as follows.

$$L_k(t) = \alpha \sum_{t=t_1}^{t_2} R_k^C(t) + (1-\alpha) \sum_{t=t_1}^{t_2} R_k^{mm}(t), \forall k \in K \quad (9)$$

Let x_{kl} be an indicator variable, which is equal to 1 if task T_k is offloaded to a computing device l, *i.e.*,

$$x_{kl} \in (0,1), \forall (k,l) \quad (10)$$

Note that, $k \in K, l \in (i,j)$ and there exists

$$\sum_k x_{kl} = 1, \forall (k,l) \quad (11)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2958400, IEEE Internet of Things Journal

Thus, each task needs to be offloaded to a suitable computing device for further processing. Note that the overall CPU cores consumed by a task T_k should be less than or equal to the maximum CPU capacity of a computing device l, *i.e.*,

$$\sum_{k=1}^{K} R_k^C(t).x_{kl} \leq U_l^C(t), \forall(k,l,t) \tag{12}$$

Similarly, the amount of memory blocks is consumed by the tasks T_k should be less than or equal to the maximum memory capacity of a computing device l, *i.e.*,

$$\sum_{k=1}^{K} R_k^{mm}(t).x_{kl} \leq U_l^{mm}(t), \forall(k,l,t) \tag{13}$$

Thus, the overall workloads consumed by the task T_k must be less than or equal to the maximum workload of a computing device l, *i.e.*,

$$\sum_{k=1}^{K} L_k(t).x_{kl} \leq W_l(t), \forall(k,l,t) \tag{14}$$

Thus, the resource utilization of a computing device l for processing the task T_k is defined as follows.

$$RU_{kl} = \frac{L_k(t)}{W_l(t)} \times 100, \forall(k,l) \tag{15}$$

2) Cost Model: The cost of a real-time task depends on the processing time of the task on the selected computing device with multiple resources and the latency for transmitting the task from the source IoT device to the selected computing device. The processing time of the task T_k depends on its size (S_k) and the CPU capacity of the computing device l (CP_l^C), *i.e.*,

$$P_{kl} = \frac{S_k}{CP_l^C}, \forall(k,l) \tag{16}$$

Thus, the processing cost for the CPU usage by the task T_k is defined as follows.

$$CO_{kl}^C = \frac{P_{kl} \times CO^C}{\tau_1}, \forall(k,l) \tag{17}$$

where CO^C represents the CPU cost of a computing device l for a unit time interval τ_1. The memory of the computing device l is consumed by the task T_k until the completion of its processing. Thus, the memory cost of the application T_k for primary data storage is defined as follows.

$$CO_{kl}^{mm} = \frac{P_{kl} \times CO^{mm}}{\tau_2}, \forall(k,l) \tag{18}$$

Here, CO^{mm} represents the memory cost of a computing device l for a unit time interval τ_2. So, the total processing cost of the task T_k is represented as follows.

$$CO_{kl} = CO_{kl}^C + CO_{kl}^{mm}, \forall(k,l) \tag{19}$$

The latency between the IoT devices k and the selected computing device l depends on the bandwidth of the network and distance. The latency between the devices k and l (LD_{kl}) is represented as follows.

$$LD_{kl} = P_{kl} + SD_{kl}, \forall(k,l) \tag{20}$$

Here SD_{kl} and P_{kl} represent the serialization delay and propagation delay respectively. The propagation delay is the ratio between the distance among the devices k and l (DT_{kl}) and the bandwidth of the network (B_{kl}), *i.e.* $P_{kl} = \frac{DT_{kl}}{B_{kl}}$, where $DT_{kl} = \sqrt{(X_k - X_l)^2 + (Y_k - Y_l)^2}$ in a two-dimensional space (X, Y). However, the serialization delay is the ratio between the sizes of the task T_k (S_k) to the transmission rate of the network (TT_{kl}) in bit per second *i.e.* $SD_{kl} = \frac{S_k}{TT_{kl}}$. Here, we assume that the LCF devices in Tier 2 are nearer to the IoT devices. So, the latency between the IoT devices and LCF devices is 0. The cost for latency is defined as follows.

$$CO_{kl}^{LD} = \frac{LD_{kl} \times CO^{LD}}{\tau_3}, \forall(k,l) \tag{21}$$

Here, CO^{LD} represents the communication cost per unit time to reach the task to the selected computing device l for a unit time interval τ_3. So, the total cost required for processing and transmitting the task T_k is defined as follows.

$$TCO_{kl} = CO_{kl} + CO_{kl}^{LD}, \forall(k,l) \tag{22}$$

The main objectives of the work are to maximize the resource utilization of the computing devices and to minimize the total processing and communication cost of the real-time tasks. The mathematical formulation of the objectives of the proposed work with the constraints are defined as follows.

$$Maximize \quad RU_{kl} \tag{23a}$$

$$Minimize \quad TCO_{kl} \tag{23b}$$

$$\text{Subject to} \quad \sum_{k=1}^{K} R_k^C(t).x_{kl} \leq U_l^C(t), \tag{23c}$$

$$\sum_{k=1}^{K} R_k^{mm}(t).x_{kl} \leq U_l^{mm}(t), \tag{23d}$$

$$\sum_{k=1}^{K} L_k(t).x_{kl} \leq W_l(t), \tag{23e}$$

$$RU_{kl} \leq 100, \tag{23f}$$

$$\sum_{k} x_{kl} = 1, \tag{23g}$$

V. Proposed Work

In this section, we design a Multi-Objective Offloading (MOO) strategy to solve the formulated problems using APSO technique. The MOO strategy finds a suitable computing device for each real-time task which has sufficient resources and requires minimum cost for processing the task. Here, we divide the real-time tasks into two categories, namely resource-intensive tasks and delay-sensitive tasks. The resource-intensive tasks mostly concern about the higher capacity of the computing resources without concerning about the latency and the cost. Such type of tasks prefer to execute on the remote CDC or the HCF devices. However, the delay-sensitive tasks mostly concern about the latency and cost. Those tasks mostly execute in the local IoT devices or the LCF devices for faster response. Here, we introduce two types of QoS parameters for offloading each task, namely resource utilization and cost. Here, we first determine the computation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2958400, IEEE Internet of Things Journal

time for running a task on both the local fog device and the centralized CDC.

Case 1: Local Processing: In this scenario, the tasks process in local IoT devices or offload to the LCF devices. The computation time of the task T_k in the computing device l is defined as follows.

$$CT_{kl} = P_{kl} + LD_{kl}, \forall(k,l) \tag{24}$$

As the task processes locally, so the required latency of that task is 0, *i.e.* LD_{kl} = 0. So, the computation time required by the task is equal to its processing time, *i.e.* $CT_{kl} = P_{kl} + 0 = P_{kl}$.

Case 2: Remote Processing: In this scenario, the task is offloaded to the HCF devices or the centralized CDC. Thus, the tasks are transmitted through a long-term evolution network which requires a latency. In such scenario, the computation time of a task T_k consists of the latency and processing time in the selected computing device l, *i.e.* $CT_{kl} = P_{kl} + LD_{kl}$. As a result, dispatching the tasks to the HCF devices or the CDC impose a long latency which causes higher completion time. The detail of the proposed MOO strategy is discussed below.

1) Representation of multi-objective Function: This phase is responsible to generate a multi-objective optimization function using a weighted-sum strategy based on two QoS parameters, *i.e.* resource utilization and cost. The resource utilization of the computing devices for processing a task T_k is presented as $RU_{kl} = \{RU_{k1}, RU_{k2}, RU_{k3}, ..., RU_{kn}\}$. Similarly, the cost of the computing devices for processing the task T_k is represented as $TCO_{kl} = \{TCO_{k1}, TCO_{k2}, TCO_{k3}, ..., TCO_{kn}\}$. The proposed multi-objective function using the two QoS parameters is defined as follows.

$$MO_{kl} = \Delta_1 RU_{kl} + \Delta_2 TCO_{kl}, \forall(k,l) \tag{25}$$

Here, the QoS parameters are aggregated in a single objective function, which is helpful for finding the optimal computing devices in the network. In Eq. (24), the coefficients Δ_1, and Δ_2 represent the user defined constants to indicate the priority of the QoS parameters. In this work, a general problem is considered that gives equal preference to both the objectives. Therefore, the summation of the weights of the coefficients is equal to 1, *i.e.* $\sum_{x=1}^{2} \Delta_x = 1$. The pseudo code of this phase is discussed in Algorithm 1.

2) Selection of optimal computing device: The main contribution of this phase is to find an optimal computing device for each real-time task based on multi-objective function. The APSO technique is used to find the computing device on fly with higher accuracy and minimum error. The multi-objective functions of the n computing devices are presented as $MO_{kl} = \{MO_{k1}, MO_{k2}, MO_{k3}, MO_{k4}, ..., MO_{kn}\}$. Here, the particles are presented by the values of the objective functions of the computing devices. For minimum optimization problem, the updated velocity of each particle using the APSO technique is represented as follows.

$$v_{kl+1} = v_{kl} + \alpha\varepsilon + \beta(MO^*_{kl} - MO_{kl}), \forall(k,l) \tag{26}$$

Algorithm 1 Representation of multi-objective Function

INPUT: $U_l^C, U_l^{mm}, R_k^C, R_k^{mm}, S_k, CP_l^C$
OUTPUT: MO_{kl}

for l: 1 to n **do**
  $W_l(t) = \alpha \sum_{t=t_1}^{t_2} U_l^C(t) + (1-\alpha) \sum_{t=t_1}^{t_2} U_l^{mm}(t)$
end for
for k: 1 to K **do**
  $L_k(t) = \alpha \sum_{t=t_1}^{t_2} R_k^C(t) + (1-\alpha) \sum_{t=t_1}^{t_2} R_k^{mm}(t)$
end for
for k: 1 to K **do**
  for l: 1 to n **do**
    if $(\sum_{k=1}^{K} L_k(t).x_{kl} \leq W_l(t))$ **then**
      $RU_{kl} = \frac{L_k(t)}{W_l(t)} \times 100$
    end if
    $CO_{kl} = CO_{kl}^C + CO_{kl}^{mm}$
    $CO_{kl}^{LD} = \frac{LD_{kl} \times CO^{LD}}{\tau_3}$
    Total Cost $TCO_{kl} = CO_{kl} + CO_{kl}^{LD}$
    Function $MO_{kl} = \Delta_1 RU_{kl} + \Delta_2 TCO_{kl}$
  end for
end for

Here, ε represents a random vector whose value lies between [0, 1] and α, β are the user-defined constraints. MO^*_{kl} and MO_{kl} represents the global-best position and the local-best position of the particle in the previous iteration respectively. However, the above-mentioned velocity function of the particle is usually used as a single objective optimization problem. For a multi-objective optimization problem, the above-mentioned velocity function does not provide the prominent result. To overcome such drawback of the APSO technique in measuring the velocity function, we define a modified velocity function which is defined as follows.

$$MO_k(i,j) = \frac{MO_{kj} - MO_{ki}}{MO_{max} - MO_{ki}}, \forall(i,j) \in l \tag{27}$$

Here, we consider two computational devices i and j, where the position of the computing device j is in a better position in the execution space than the computing device i. The modified velocity function obtained by computing device i from j is denoted as $MO_k(i,j)$. MO_{max} represents the maximal value of the objective function. The distance between the computing device i and j is represented as the euclidean distance which is formulated below.

$$ED(MO_i) = \sqrt{\sum_{x=1}^{D} ED(MO_{kix} - MO_{kjx}), \forall(i,j) \in l} \tag{28}$$

where D represents the dimension of the objective Function which is two for the above problem. Finally, we develop a Pareto dominance strategy based on the modified velocity function. For a population of the computing devices at the d^{th} iteration, taking the modified velocity function and the distance as two measures. Here, we first select the non-dominated computing devices in the population and put them into a PO set. Next, for selecting a computing device from the current population, we randomly select a computing device from the set as an optimal computing device. Here, at each iteration,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2958400, IEEE Internet of Things Journal

Algorithm 2 Selection of optimal computing device

INPUT: $n, \alpha, \beta, \varepsilon$
OUTPUT: l

for l: 1 to n **do**
  MO_{kl} = Initial Solution()
end for
while $MO_{kl} \leq \delta$ **do**
  arg min(MO_{kl})
  for i: 1 to n **do**
    for j: 1 to n **do**
      $v_{kl+1} = v_{kl} + \alpha\varepsilon + \beta(MO^*_{kl} - MO_{kl})$
      $MO_k i, j = \frac{MO_{kj} - MO_{ki}}{MO_{max} - MO_{ki}}$
      $ED(MO_i) = \sqrt{\sum_{x=1}^{D} ED(MO_{kix} - MO_{kjx})}$
      if $MO_{kjx} > MO_{kix}$ **then**
        $ED(MO_{kix}, MO_{kjx}) = (MO_{kjx} - MO_{kix})$
      end if
      Pareto dominance relationship for MO_{kl}
      $ds_i = (1-\beta)ds_{i-1} + \beta MO^*_{kl} + \alpha\epsilon^i$
    end for
  end for
  Find an optimal computing device i
end while

each particle moves towards another particle whose velocity value may be better than its personal best or even the global best value in the previous iteration. We classify the process of selection of computing device into three sets:

Set 1: The set of computing devices whose true value at current iteration d^{th} are better than the best position in previous iteration $(d-1)^{th}$, *i.e.* $S_1 = \{L : MO_{dk}(i,j) \leq MO_{d-1k}(i,j)\}, \forall(i,j) \in l, L \in (F,H)$

Set 2: The set of computing devices whose true velocity values at current iteration d^{th} are worse than in previous iteration $(d-1)^{th}$, *i.e.* $S_1 = \{L : MO_{dk}(i,j) > MO_{d-1k}(i,j)\}, \forall(i,j) \in l, L \in (F,H)$

Set 3: The set of computing devices whose true values at current iteration d^{th} are better than the global best at the $(d-1)^{th}$ iteration, *i.e.* $S_3 = \{L : MO_{dk}(i,j) > MO_g(i,j)\}, \forall(i,j) \in l, L \in (F,H)$

Finally, we define the optimal computing device at the d^{th} iteration, *i.e.* $MO^o_{dk} \leq min_{i \neq o} MO_{d-1k}(i,j)$. In the above sets, **Set 1** and **Set 2** are mutually exclusive, denoting, the set of computing devices whose personal best position can be changed and the set of computing devices whose personal best do not need to be updated at the d^{th} iteration. **Set 3** indicates whether the global best can be changed or not. Finally, update the positions of the computing devices based on the following function.

$$ds_i = (1-\beta)ds_{i-1} + \beta MO^*_{kl} + \alpha\epsilon^i \tag{29}$$

β is the user-defined constant and ϵ is a random vector whose value lies in the range [0, 1]. Finally, the real-time tasks are assigned to the selected computing devices in First Come First Serve order. The pseudo code of this phase is presented in Algorithm 2. The overall setup of the MOO strategy is shown in Fig. 2.

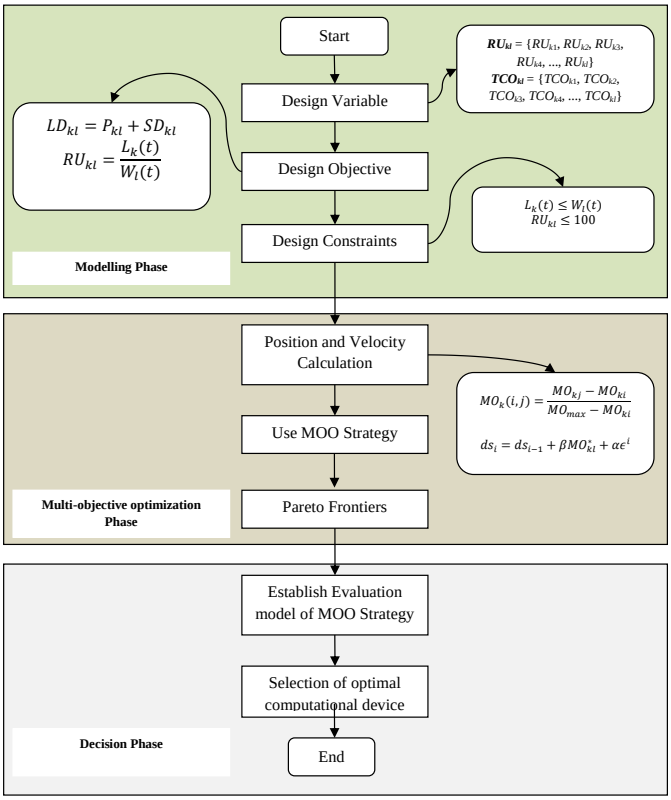


Fig. 2: The Flowchart of MOO strategy

TABLE I: Simulation parameters

Parameter Description	Values
Number of IoT devices	100
Number of HCF devices	10
Number of LCF devices	20
Number of cloud servers	4
CPU frequency of LCF devices	1.4 GHz
RAM Size of LCF devices	1 GB
CPU frequency of HCF devices	2.4 GHz
RAM Size of HCF devices	4 GB
CPU frequency of each cloud server	3.5 GHz
RAM Size of each cloud server	64 GB
The storage capacity of each cloud server	1 TB

VI. Empirical Evaluation

In this section, we investigate the performance of the proposed offloading strategy through extensive simulation run. In addition, we also analyze the parameters of the APSO technique for further improving the performance of the proposed MOO algorithm. We also verify the efficiency and feasibility of the MOO algorithm with existing state-of-art algorithms, proposed in [22], [23] and [24] on various performance matrices, *i.e.* latency, computation time, resource utilization and average cost.

A. Performance Setup

In the simulation, we consider 100 IoT devices as a form of various temperature and moisture sensors for collecting the data from the environment which are evaluated in the fog devices or the cloud servers based on the resource availability

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2958400, IEEE Internet of Things Journal

TABLE II: P-value analysis

Data sets	D-1	D-2	D-3	D-4
D-1	0	3.87E-07	5.64E-08	2.65E-06
D-2	2.84E-09	0	6.87E-09	4,67E-04
D-4	1.76E-08	1.65E-06	0	1.76E-05
D-4	2.23E-05	5.32E-08	4.31E-06	0

TABLE III: Result analysis of synthetic data sets

Error of Data sets		
Data sets	**Minimum Error**	**Mean Error**
D-1	3.12E-06	0.0241561
D-2	2.76E-08	0.0365423
D-3	1.56E-07	0.0342154
D-4	8.34E-08	0.021342

and priority of the tasks. The priority of each task is assigned based on the urgency of its results for the IoT devices. Here, we consider 10 HCF devices in terms of workstations and high-performance laptops with minimum CPU frequency of each logical core is 2.4 GHz and physical memory (*i.e.* RAM) capacity is around 4 GB. The fog devices relate to a high-speed short-term communication protocol, *i.e.* WiFi. In lower-level, we consider 20 LCF devices with minimum CPU and memory capacity in terms of Raspberry Pi and mobile devices. The CPU frequency and memory capacity of such devices are 1.4 GHz and 1 GB respectively. The low-capacity fog devices are connected with the IoT devices with WiFi or Bluetooth protocols. However, the IoT devices and the fog devices communicate with the centralized CDC with long term evolution protocols. The simulation parameters of the proposed real-time environment are shown in Table I.

B. Data set Validation

During the evaluation process, we have generated four types of data sets based on the processing and communication time and cost of the real-time tasks on the selected computing devices with their resource utilization. However, before running the MOO strategy over the data sets, a data pre-processing technique is used to convert the raw data into clear data sets. In reality, the APSO technique does not support zero and null values, therefore to run the APSO technique, a data pre-processing technique is used for managing the zero and null values from the original raw data set. The significance level of the clear data sets with an unpaired t-test analysis is shown in Table II. In statistics, the P value of the t-test is used to find the significance level of the data sets. Conventionally, the data sets are called significant if their P value is less than 0.05 and highly significant if $P \leq 0.001$. From Table II, it is observed that the P value between the data sets is less than 0.001, *i.e.* the data sets are used for performance analysis of the proposed MOO algorithm are highly significant.

C. Parameter Analysis

In this section, we analyze the two important parameters of the APSO-based MOO technique, *i.e.* α, and β and also fix the population size of the algorithm for improving its performance. Here, we consider the values of α, and β parameters are in the range [0, 1] and the population size is within [0, 100]. During experimental analysis, we fix the values of other parameters when finding the best possible value of a single parameter, *i.e.* we fix the value of β and population size while finding the appropriate value of α. The proposed strategy is evaluated using 100 iterations for finding the fixed value of each parameter and the minimum error is recorded with 1000 independent runs. The minimum error and mean error of each parameter over different possible parametric values are shown in Table III. The β parameter produces a minimum error at point 0.65 and the α parameter produces a minimum error at point 0.15. These values are finalized for further evaluation of the proposed strategy. Similarly, the population size of the proposed algorithm is evaluated over different data sets and it is observed that it produces a minimum error at point 60. Thus, the population size of the MOO strategy is fixed at point 60. The final convergence speed of the MOO strategy over the fixed values of the different parameters of the APSO technique is presented in Table IV. Here, the minimum error of the convergent rate of the MOO strategy is less than the predefined threshold value 0.0001, which proves that the algorithm should reach its target level for finding the optimal position in the space. Moreover, the results also satisfy the stability, convergence speed and the quality of the solution.

D. Comparative Analysis

We verify the feasibility of the proposed algorithm with existing three state-of-the-art algorithms *i.e.* COG [24], MFC [23] and OJD [22], with various performance matrices.

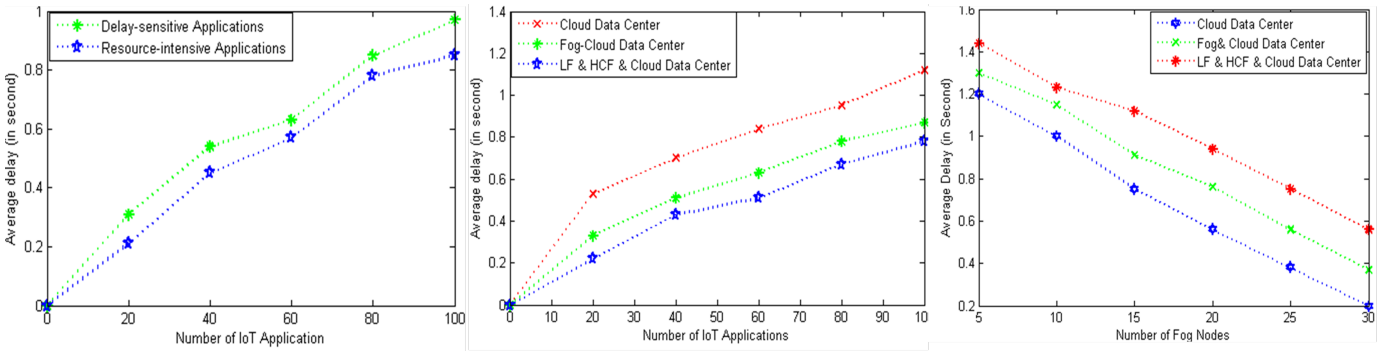
1) Average Delay (AD): Here, we first focus on the delay-sensitive tasks and investigate the objective function for finding the suitable computing device for each task. The proposed MOO strategy decides whether it assigns the task locally or offloads to the fog or CDC for further processing. The Boolean variable $x_{ij} = 0$ represents that the task T_k is offloaded to the computing devices l. For evaluation purpose, initially, we consider a set of cloud servers for processing the real-time tasks instead of local fog devices. The IoT devices and sensor execute the tasks locally or offload them to the CDC. Fig. 3(a) shows the average delay (also known as latency) for different types of tasks while processing in the remote CDC. From this scenario, we observe that most of the delay-sensitive tasks experienced a huge delay as compared with resource-intensive tasks without fog devices. However, by increasing the number of fog devices, the average delay is going to reduce significantly as the tasks are offloaded to the local fog devices which meet the resource capacity and produce minimum delay. Fig. 3(b) represents the average delay for the fog-cloud environment with 20 LCF and 10 HCF devices while varying the number of tasks and Fig. 3(c) represents the average delay for 100 delay-sensitive tasks while varying the number of fog devices. With increasing the number of real-time tasks, the average delay in cloud environment increased significantly as compared with our multi-level fog-cloud model. Similarly, by increasing the number of fog devices, the proposed model reduces the average delay as compared with the centralized CDC.

Next, we investigate the average delay of the tasks of the proposed MOO strategy over the existing offloading algo-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2958400, IEEE Internet of Things Journal

TABLE IV: Parametric analysis of APSO technique

Data sets	Error calculation of Beta parameter									
	Beta	β= 0.15	β = 0.25	β = 0.35	β = 0.45	β = 0.55	β = 0.65	β = 0.75	β = 0.85	β = 1.0
D-1	Minimum	1.89E-07	2.09E-06	1.94E-05	1.45E-05	2.93E-05	4.78E-05	4.91E-06	0.04321	6.75E-05
	Mean	0.013876	0.01323	0.01421	0.010542	0.016731	0.009541	0.021032	0.020091	0.024121
D-2	Minimum	2.09E-07	2.15E-06	1.65E-06	2.12E-05	4.76E-05	6.12E-05	8.01E-05	6.11E-06	7.10E-05
	Mean	0.00985	0.01432	0.01432	0.01943	0.01984	0.019843	0.02314	0.21321	0.01041
D-3	Minimum	1.23E-07	1.04E-06	2.13E-05	3.12E-06	2.19E-06	7.16E-06	9.12E-05	4.12E-05	4.01E-06
	Mean	0.01417	(0.01735	0.01932	0.02132	0.014312	0.02341	0.014323	0.019893	0.020121
D-4	Minimum	1.49E-07	1.87E-06	1.24E-05	3.95E-05	2.86E-06	1.96E-05	2.01E-05	2.91E-05	5.01E-05
	Mean	0.01234	0.01761	0.02134	0.02013	0.013214	0.03984	0.02312	0.091234	0.02934
	Error calculation of Beta parameter									
Alpha		α = 0.15	α = 0.25	α = 0.35	α = 0.45	α = 0.55	α = 0.65	α = 0.75	α = 0.85	α = 1.0
D-1	Minimum	2.98E-06	2.12E-05	8.92E-06	4.91E-06	2.01E-06	2.13E-07	1.54E-05	1.65E-05	3.76E-07
	Mean	0.02314	0.02012	0.02098	0.01965	0.01956	0.01956	0.0198	0.01543	0.02314
D-2	Minimum	3.43E-05	4.10E-05	4.72E-06	5.09E-05	1.32E-06	7.56E-07	2.07E-05	1.87E-06	7.02E-05
	Mean	0.02165	0.04012	0.01642	0.01983	0.02198	0.03972	0.012151	0.036521	0.01982
D-3	Minimum	2.15E-05	1.98E-06	5.77E-05	7.83E-05	1.98E-06	1.98E-08	1.98E-05	2.67E-06	2.98E-06
	Mean	0.054311	0.02132	0.02915	0.02134	0.02315	0.02361	0.02019	0.02341	0.017631
D-4	Minimum	7.15E-06	3.97E-05	3.09E-07	3.15E-06	1.76E-06	2.54E-08	3.14E-06	2.95E-06	4.34E-05
	Mean	0.08312	0.02019	0.0291	0.02054	0.05413	0.02014	0.02134	0.02983	0.04123
	Error calculation of population size									
Pop_Size		P=10	P=20	P=30	P=40	P=50	P=60	P=70	P=80	P=100
D-1	Minimum	2.91E-06	2.91E-06	9.01E-05	4.09E-06	1.42E-06	2.08E-07	2.01E-06	1.87E-05	3.01E-05
	Mean	0.02182	0.02012	0.03973	0.02019	0.021342	0.01984)	0.01432	0.01983	0.02314
D-2	Minimum	4.19E-05	5.91E-06	4.17E-05	3.97E-06	1.77E-05	1.01E-06	8.92E-05	5.01E-05	2.13E-06
	Mean	0.052134	0.01234	0.03214	0.02198	0.029813	0.01214	0.02134	0.02165	0.02918
D-3	Minimum	4.06E-06	2.89E-06	7.93E-06	5.01E-06	1.89E-06	9.01E-07	2.15E-05	1.89E-06	8.02E-05
	Mean	0.051234	0.02174	0.02132	0.01982	0.031453	0.023145	0.012031	0.05312	0.01532
D-4	Minimum	2.01E-05	1.73E-06	8.92E-06	9.12E-05	1.81E-06	2.06E-07	1.99E-05	2.98E-05	3.98E-06
	Mean	0.0321	0.02016	0.01912	0.03212	0.019832	0.002134	0.02019	0.01098	0.01983



(a) (b) (c)

Fig. 3: Average latency delay: (a) for delay-sensitive and resource-intensive tasks on CDC; (b) for various number of tasks; (c) for various number of fog devices

rithms. The superiority of the MOO strategy over the existing algorithms is shown in Fig. 4(a). The MOO strategy finds a suitable computing device for each real-time task based on the APSO technique which reduces the latency of the tasks.

2) Computation Time (ACT): Computation time is defined as the amount of time required by a real-time task to complete its execution within stipulated time. The computation time depends on the resource capacity of the selected computing device and the latency for transmitting the task. In previous subsection, we have already discussed that the average delay of the tasks is reduced for proposed system model due to the MOO strategy. This minimizes the latency and the overall computation time. Fig. 5(a) and 5(b) represents the average computation time for various tasks and computational model in fog-cloud environment, respectively. In our multi-tier model, we have considered 20 LCF and 10 HCF devices while varying the number of real-time tasks. With increasing the number of tasks, the average delay in the cloud environment significantly increases as compared with the proposed system model. Similarly, by increasing the number of fog devices, the proposed system model reduces the computation time as compared with the existing fog-cloud environment. The main reason being; as more delay-sensitive tasks are assigned locally or they are offloaded to the local fog devices, it significantly reduced the overall computation time.

In addition, we compared the average computation time of the tasks using the proposed MOO strategy over the existing offloading algorithms. The superiority of the MOO strategy over the existing algorithms is shown in Fig. 4(b).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2958400, IEEE Internet of Things Journal

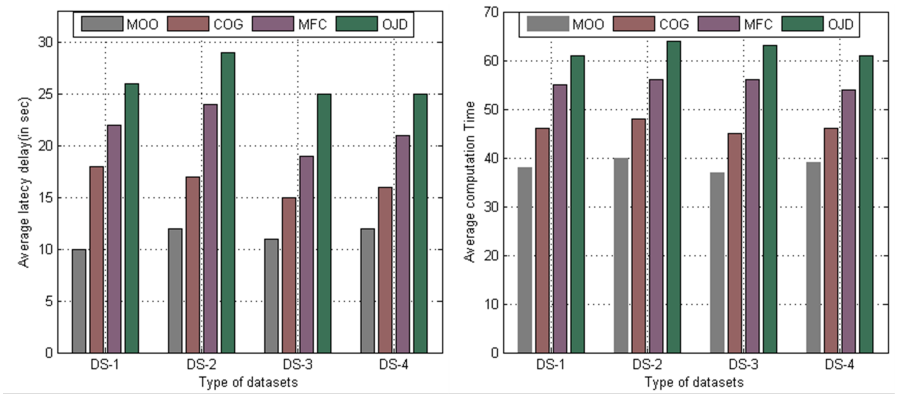


Fig. 4: Performance analysis: (a) Average delay; (b) Average computation time

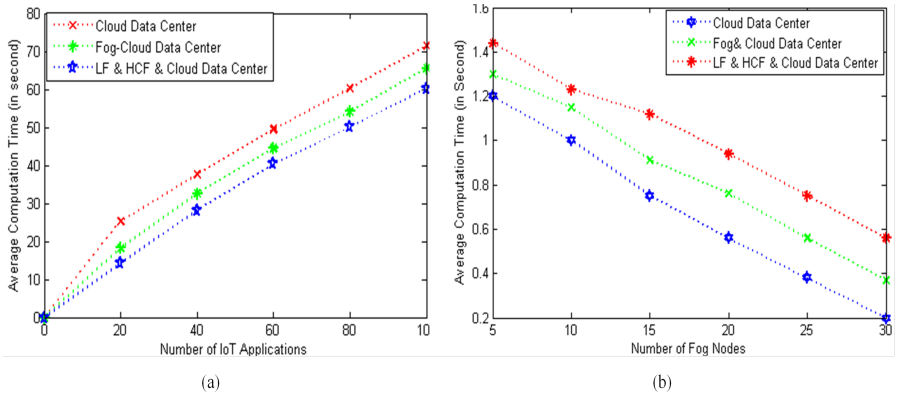


Fig. 5: Average computation Time: (a) for various number of IoT applications; (b) for IoT applications using a various number of fog devices

3) Resource Utilization (RU): Resource utilization of the computing devices depend on the readability of the computing resources and the amount of the resources used for processing the current real-time tasks. Maximum resource utilization of the computing devices also reduces the resource wastage. The comparative analysis of the proposed MOO algorithm and the existing state-of-the-art algorithms is shown in Fig. 6(a). The MOO strategy finds an optimal computing device based on the current resource availability and cost of the computing resources. The MOO strategy uses the APSO technique for offloading the real-time tasks efficiently and maximizes the overall resource utilization of the computing devices.

4) Average Cost (AC): The average cost of the real-time tasks on the selected computing devices is shown in Fig. 6(b). The main aim of the offloading strategies is to find an optimal computing device that can minimize the transmission and processing cost of the tasks. As a result, the tasks need to be assigned to the local fog devices with a minimum delay while meeting the resource requirements. To meet the above challenge, the APSO technique helps to deploy each task to the optimal fog device using multiple QoS objectives. This minimizes the total transmission and processing cost of the tasks. However, COG, MFC and OJD algorithms use a single objective optimization strategy for finding optimal computing device, that make the difference.

TABLE V: Statistical analysis of different performance matrices

Parameter	Data sets	Algo.	Statistical Parameters			
			Best	Worst	Mean	Std.
AD	D-1	OJD	4.43E-01	9.43E-01	6.12E-01	5.76E-01
		MFC	3.49E-01	8.62E-01	5.97E-01	4.97E-01
		COG	2.11E-01	7.98E-01	5.15E-01	4.43E-01
		MOO	1.44E-01	6.18E-01	4.89E-01	4.01E-01
ACT	D-1	OJD	7.23E-01	9.40E-01	6.84E-01	7.32E-01
		MFC	6.97E-01	8.87E-01	5.87E-01	5.41E-01
		COG	6.34E-01	8.45E-01	5.43E-01	4.71E-01
		MOO	2.84E-01	7.88E-01	4.01E-01	3.87E-01
RU	D-1	OJD	5.43E-01	1.04E+00	8.12E-01	7.76E-01
		MFC	5.01E-01	9.62E-01	7.97E-01	6.97E-01
		COG	4.34E-01	8.98E-01	7.15E-01	6.43E-01
		MOO	2.97E-01	7.18E-01	5.89E-01	5.01E-01
AC	D-1	OJD	7.13E-01	9.21E-01	6.84E-01	7.32E-01
		MFC	6.67E-01	8.54E-01	5.87E-01	5.41E-01
		COG	6.24E-01	8.01E-01	5.43E-01	4.71E-01
		MOO	2.64E-01	7.32E-01	3.01E-01	2.87E-01

5) Results and Discussion: We later performed statistical analysis between the performance of the proposed MOO strategy and existing state-of-the-art algorithms in terms of mean and standard deviation. The results show that the MOO strategy performs better than the existing strategies over all

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2958400, IEEE Internet of Things Journal

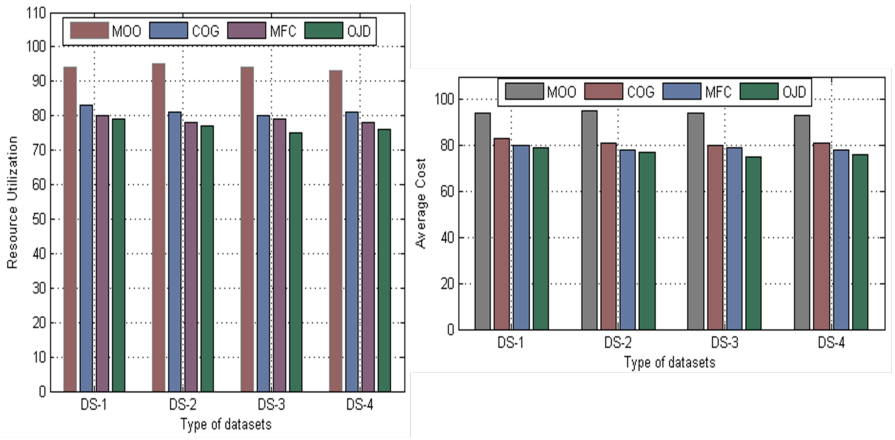


Fig. 6: Performance analysis: (a) Resource Utilization; (b) Average cost

the four real-time data sets. Table V presents the statistical analysis for data set 1, as a representative. This analysis proves that the MOO strategy finds the optimal computing device efficiently for each real-time task with minimum error and standard deviation as compared with the existing algorithms. The improvement percentages of the average cost of the MOO strategy are 14%, 19% and 22% more than COG, MFC, and OJD algorithms respectively. MOO strategy maximizes the resource utilization 18%, 24% and 27% as compared with the existing COG, MFC, and OJD algorithms, respectively.

VII. Conclusion and Future Work

In this paper, we have designed a multi-tier fog-cloud model for processing real-time tasks efficiently. Here, we have formulated an optimal objective function with multiple QoS parameters, namely total cost and resource utilization. Moreover, an APSO-based offloading strategy, namely MOO strategy has been designed to find an optimal computing device for processing each real-time task. The main purpose of introducing the APSO technique is to minimize the error rate and increasing the accuracy of the offloading strategy. On the basis of converging speed of the multiple QoS parameters of the real-time tasks, our algorithm can be more adaptable to find the suitable computing devices for the tasks. In the experimental analysis section, initially, we investigated the parameters of the APSO technique over various real-time data sets and fixed their parametric values. Moreover, we evaluated the performance of the proposed algorithms over existing state-of-the-art algorithms using various performance metrics. The analytical results show that the proposed strategy minimizes the overall delay and cost of the tasks while increasing the resource utilization of the computing devices as compared with state-of-the-art algorithms. The experimental results indicate that the proposed strategy outperforms the existing schemes in terms of average delay, computation time, resource utilization and average cost by 18%, 21%, 27%, and 23%, respectively.

In the future work, we plan to design an energy efficient fog-cloud model for meeting multiple QoS parameters of the real-time IoT applications. Moreover, we plan to introduce different advanced machine learning strategies, namely reinforcement learning and deep reinforcement learning for improving the performance and accuracy of the offloading strategies in a complex fog-cloud environment.

Acknowledgement

This work has been partially supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825040 (RADON), Estonian Centre of Excellence in IT (EXCITE) funded by the European Regional Development Fund, and DST(SERB), India, Sanction No. EEQ/2018/000888.

References

[1] S. S. Shah, M. Ali, A. W. Malik, M. A. Khan, and S. D. Ravana, "vfog: A vehicle-assisted computing framework for delay-sensitive applications in smart cities," *IEEE Access*, vol. 7, pp. 34 900–34 909, 2019.

[2] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, 2014.

[3] C. Perera, P. P. Jayaraman, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context-aware dynamic discovery and configuration of things in smart environments," in *Big Data and Internet of Things: A Roadmap for Smart Environments*. Springer, 2014, pp. 215–241.

[4] R. Buyya, S. N. Srirama, G. Casale *et al.*, "A manifesto for future generation cloud computing: research directions for the next decade," *ACM computing surveys (CSUR)*, vol. 51, no. 5, p. 105, 2018.

[5] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, "Quality of experience (qoe)-aware placement of applications in fog computing environments," *Journal of Parallel and Distributed Computing*, vol. 132, pp. 190–203, 2019.

[6] X. Hou, Z. Ren, W. Cheng, C. Chen, and H. Zhang, "Fog based computation offloading for swarm of drones," in *IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.

[7] A. Brogi and S. Forti, "Qos-aware deployment of iot applications through the fog," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185–1192, 2017.

[8] M. Asif-Ur-Rahman, F. Afsana, M. Mahmud, M. S. Kaiser, M. R. Ahmed, O. Kaiwartya, and A. James-Taylor, "Towards a heterogeneous mist, fog, and cloud based framework for the internet of healthcare things," *IEEE Internet of Things Journal*, 2018.

[9] H.-L. Liu, L. Chen, K. Deb, and E. D. Goodman, "Investigating the effect of imbalance between convergence and diversity in evolutionary multiobjective algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 3, pp. 408–425, 2018.

[10] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii," in *International Conference on Parallel Problem Solving From Nature*. Springer, 2000, pp. 849–858.

[11] S. Jiang, S. Yang, Y. Wang, and X. Liu, "Scalarizing functions in decomposition-based multiobjective evolutionary algorithms," *IEEE Trans. on Evolutionary Computation*, vol. 22, no. 2, pp. 296–313, 2018.

[12] X.-S. Yang, S. Deb, and S. Fong, "Accelerated particle swarm optimization and support vector machine for business optimization and applications," in *International Conference on Networked Digital Technologies*. Springer, 2011, pp. 53–66.

[13] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1826–1857, 2018.

[14] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for iot: Review, enabling technologies, and research opportunities," *Future Generation Computer Systems*, vol. 87, pp. 278–289, 2018.

[15] Y. Wang, K. Wang, H. Huang, T. Miyazaki, and S. Guo, "Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 976–986, 2018.

[16] K. Liang, L. Zhao, X. Chu, and H.-H. Chen, "An integrated architecture for software defined and virtualized radio access networks with fog computing," *IEEE Network*, vol. 31, no. 1, pp. 80–87, 2017.

[17] X. Zhao, L. Zhao, and K. Liang, "An energy consumption oriented offloading algorithm for fog computing," in *International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*. Springer, 2016, pp. 293–301.

[18] C. Fricker, F. Guillemin, P. Robert, and G. Thompson, "Analysis of an offloading scheme for data centers in the framework of fog computing," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 1, no. 4, p. 16, 2016.

[19] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks," *IEEE access*, vol. 4, pp. 5896–5907, 2016.

[20] R. Hasan, M. Hossain, and R. Khan, "Aura: An incentive-driven ad-hoc iot cloud framework for proximal mobile computation offloading," *Future Generation Computer Systems*, vol. 86, pp. 821–835, 2018.

[21] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. on Communications*, vol. 64, no. 10, pp. 4268–4282, 2016.

[22] H. Tan, Z. Han, X.-Y. Li, and F. C. Lau, "Online job dispatching and scheduling in edge-clouds," in *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*. IEEE, 2017, pp. 1–9.

[23] J. He, J. Wei, K. Chen, Z. Tang, Y. Zhou, and Y. Zhang, "Multitier fog computing with large-scale iot data analytics for smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 677–686, 2018.

[24] H. Shah-Mansouri and V. W. Wong, "Hierarchical fog-cloud computing for iot systems: A computation offloading game," *IEEE Internet of Things Journal*, 2018.

[25] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, "Profit-aware application placement for integrated fog-cloud computing environments," *Journal of Parallel and Distributed Computing*, vol. 135, pp. 177–190, 2020.

[26] M. Adhikari, M. Mukherjee, and S. N. Srirama, "DPTO: A Deadline and Priority-aware Task Offloading in Fog Computing Framework Leveraging Multi-level Feedback Queueing," *IEEE Internet of Things Journal*, pp. 1–1, 2019.

[27] A. Brogi, S. Forti, C. Guerrero, and I. Lera, "Meet genetic algorithms in monte carlo: Optimised placement of multi-service applications in the fog," in *Int. Conf. on Edge Computing (EDGE)*. IEEE, 2019, pp. 13–17.

[28] M. Adhikari and S. N. Srirama, "Multi-objective accelerated particle swarm optimization with a container-based scheduling for internet-of-things in cloud environment," *Journal of Network and Computer Applications*, vol. 137, pp. 35–61, 2019.

[29] C. Jian, M. Li, and X. Kuang, "Edge cloud computing service composition based on modified bird swarm optimization in the internet of things," *Cluster Computing*, pp. 1–9, 2018.

[30] M. U. Sharif, N. Javaid, M. J. Ali, W. A. Gilani, A. Sadam, and M. H. Ashraf, "Optimized resource allocation in fog-cloud environment using insert select," in *International Conference on Network-Based Information Systems*. Springer, 2018, pp. 611–623.

[31] S. Bitam, S. Zeadally, and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm," *Enterprise Information Systems*, vol. 12, no. 4, pp. 373–397, 2018.

[32] H. Shi, Y. Feng, R. Luo, and J. Zheng, "Multi-objective optimization for iot devices association in fog-computing based ran," in *Intl. Conf. on Internet of Things as a Service*. Springer, 2018, pp. 340–347.

[33] R. G. Aryal and J. Altmann, "Dynamic application deployment in federations of clouds and edge resources using a multiobjective optimization ai algorithm," in *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, 2018, pp. 147–154.

[34] J. Wan, B. Chen, S. Wang, M. Xia, D. Li, and C. Liu, "Fog computing for energy-aware load balancing and scheduling in smart factory," *IEEE Trans. on Industrial Informatics*, vol. 14, no. 10, pp. 4548–4556, 2018.

[35] R. Ezhilarasie, M. S. Reddy, and A. Umamakeswari, "A new hybrid adaptive ga-pso computation offloading algorithm for iot and cps context application," *Journal of Intelligent & Fuzzy Systems*, no. Preprint, pp. 1–9, 2019.

[36] N. Lynn, M. Z. Ali, and P. N. Suganthan, "Population topologies for particle swarm optimization and differential evolution," *Swarm and evolutionary computation*, vol. 39, pp. 24–35, 2018.

Mainak Adhikari is currently working as a Post Doctorate Research Fellow at University of Tartu, Estonia. He has completed his Ph.D in Cloud Computing from IIT(ISM) Dhanbad, India in 2019. He has obtained his M.Tech.From Kalyani University in the year 2013. He earned his B.E.Degree from West Bengal University of Technology in the year of 2011. His area of research includes Internet of Things, Fog Computing, Cloud Computing, Serverless Computing and Evolutionary algorithm. He has Contributed numerous research Articles in various national and inter-national journal and Conference.

Satish Narayana Srirama is a Research Professor and the head of the Mobile & Cloud Lab at the Institute of Computer Science, University of Tartu, Estonia and a Visiting Professor at University of Hyderabad, India. He received his PhD in computer science from RWTH Aachen University, Germany. His current research focuses on cloud computing, mobile web services, mobile cloud, Internet of Things, fog computing, migrating scientific computing and enterprise applications to the cloud and large-scale data analytics on the cloud. He is IEEE Senior Member, an Editor of Wiley Software: Practice and Experience, a 49 year old Journal, was an Associate Editor of IEEE Transactions in Cloud Computing and a program committee member of several international conferences and workshops. Dr. Srirama has co-authored over 130 refereed scientific publications in international conferences and journals.

Tarachand Amgoth received B.Tech in Computer Science and Engineering from JNTU, Hyderabad and M.Tech in Computer Science Engineering from NIT, Rourkela in 2002 and 2006 respectively and Ph.D. form Indian Institute of Technology (ISM), Dhanbad in 2015. Presently, he is working as an Assistant professor in the Department of Computer Science and Engineering, Indian Institute of Technology (ISM), Dhanbad. His current research interest includes wireless sensor networks, cloud computing, Fog/Edge computing, serverless computing and Internet of Things.