Internet of Things 6 (2019) 100053

Contents lists available at ScienceDirect

Internet of Things

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

Research article

Energy efficient offloading strategy in fog-cloud environment for IoT applications

Mainak Adhikari*, Hemant Gianey

Department of Computer Science & Engineering, Thapar Institute of Engineering and Technology, Patalia, Punjab, India

A R T I C L E I N F O

Article history:
Received 28 January 2019
Revised 15 March 2019
Accepted 15 April 2019
Available online 26 April 2019

Keywords:
IoT application
Multi-objective optimization
Fog computing
Cloud computing
Quality-of-Service
Computational offloading

A B S T R A C T

Nowadays, cloud computing leverages the capability of Internet-of-Things (IoT) applications by providing computing resources as a form of virtual machine (VM) instances. However, the Cloud data center consumes a large amount of energy while transmitting and computing the IoT applications which lead to a high carbon footprint. On the other hand, the Fog nodes provide various cloud services at the edge of the network which can run the IoT applications locally with minimum energy consumption and delay. Due to the limited resource capacity, the Fog nodes are not suitable for processing the resource-intensive IoT applications. To address these challenges, in this paper, we build sustainable infrastructure in Fog-Cloud environment for processing delay-intensive and resource-intensive applications with an optimal task offloading strategy. The proposed offloading strategy uses Firefly algorithm for finding an optimal computing device based on two Quality-of-Service (QoS) parameters such as energy consumption and computational time. The main objectives of this strategy are to minimize the computational time and the energy consumption of the IoT applications with minimum delay. The effect of the control parameters of the Firefly technique is investigated thoroughly. Through comparisons, we show that the proposed method performs better than the existing ones in terms of various performance metrics including computational time, energy consumption, CO_2 emission, and Temperature emission.

1. Introduction

The Internet-of-Things (IoT) connects billions of complex heterogeneous objects to the network, which bridges the gap between the real and virtual world [1]. The IoT applications can bring a promising future to the smart agriculture, smart cities, smart health, smart transportation, smart waste and water management, smart power grid, etc. [2,3]. The ultimate goal of the IoT applications is to provide seamless services without the intervention of the users. However, in reality, sensing the data items from the smart objects and transferring the data to the centralized cloud environment is less useful [4]. This may increase the latency time and response time of the IoT applications with maximum delay. Further, this approach may increase the resource wastage and the computation time of the applications. To overcome the drawbacks of the centralized cloud environment, Fog computing paradigm has been developed by pushing the processes to the local Fog devices for faster processing. The key concept of Fog computing is to provide services at the edge of the network where the data are generated [5,6]. This may increase the potential of the IoT resources to process the real-time applications of the neighboring

* Corresponding author.
E-mail addresses: mainak.adhikari@thapar.edu (M. Adhikari), hgianey@gmail.com (H. Gianey).

https://doi.org/10.1016/j.iot.2019.100053
2542-6605/© 2019 Elsevier B.V. All rights reserved.

devices in a one-hop manner. However, the Fog devices have limited computational and storage capabilities for processing the applications. Due to the inherent strengths and weakness, neither cloud computing nor Fog computing paradigm addresses the above-mentioned challenges alone [7]. Therefore, both the paradigms need to work together in order to build a sustainable IoT infrastructure for smart IoT applications.

Efficient allocation of the IoT applications to the suitable computational devices including Fog nodes and cloud data center is a challenging problem in Fog-Cloud environment. Heterogeneous and powerful computing devices are required for processing various types of IoT applications including delay-intensive and resource-intensive applications. Task offloading strategy in Fog-Cloud environment has received much attention from the researchers and developers due to the growing development of IoT applications. The IoT devices should offload the applications to a suitable computational device based on different conflicting QoS parameters including computational time, latency delay, energy consumption, resource utilization, throughput, and so on. In the real world, there is a variety of offloading strategy involving more than one conflict objectives. These problems are referred to as multi-objective optimization problems (MOPs) [8]. Unlike single-objective optimization, the target of solving a MOP is to find a set of trade-off solutions known as a Pareto front in the objective space. In past two decades, two types of multi-objective evolutionary algorithms have been developed to solve MOPs- (i) finding a set of Pareto optimal solutions in a single run [9] and (ii) non-Pareto-based algorithm which solves the multi-objective evolutionary algorithm based on decomposition method [10]. Here, the algorithm decomposes a MOP into a number of single-objective optimization problems (called sub-problems). Most of the existing offloading strategies for IoT applications [11–20] ignore considering multiple-objective issues such as latency time, response time, computational time and energy consumption simultaneously.

There are still many recent optimization algorithms whose effectiveness is yet to be explored in the context of a multi-objective scheduling problem. One such algorithm is the Firefly algorithm (FA) [21]. The optimization algorithms are the high-level problem-independent algorithms which have a set of rules or strategies to find an optimal solution to a given problem [22]. The convergence speed of the optimization algorithms is the global (or nearly global) optimal which is better than the traditional techniques. Therefore, the meta-heuristic algorithms have also been increasingly used to find a suitable computational device for each IoT application using multiple QoS parameters [23]. FA is an effective optimization technique based on swarm intelligence, which has been successfully applied to various scientific problems. The FA uses the global best of the individuals for convergence of the algorithm and reduces the randomness as iteration proceeds. Another advantage of the FA is the global search ability by using the Levy flight's process. This algorithm is designed to simulate the intelligence of Firefly mating technique in the natural environment which involves complex selection procedure of best mating partner.

In this paper, we developed a new bi-objective energy efficient offloading strategy (EES) based on Firefly technique. The major contribution of the proposed strategy is to find an optimal computational device for each IoT applications using two objectives of IoT applications including computational time and energy consumption. Here, we design a fitness function based on bi-objective optimization parameters including energy consumption and computation time of the executing devices using the weighted-sum method. This method linearly aggregates all the individual objective function of a MOP into one objective by using a weight vector. Here, we define a modified attractiveness function to measure the Firefly (computational server) attractiveness instead of the distance-based attractiveness, which helps to determine the optimal position of a computational device automatically by using both the modified attractiveness function and distance from the brighter Fireflies (optimal computational server). The existing Firefly technique helps to find an optimal computational device for each application as a form of Fog devices or the cloud data center in a 2-D plane based on the bi-objective optimization parameters. The main objective of this strategy is to minimize the total energy consumption and computation time of the cloud data center. The proposed EES strategy is compared with the existing ones in terms of various performance metrics including computational time, energy consumption, CO_2 emission, and Temperature emission. The results of the performance metrics are further evaluated based on various statistical parameters including maximum, minimum, mean and standard deviation. The major contributions of the works are-

A. Design a bi-objective offloading strategy considering two major objectives of IoT applications, namely, energy consumption and computational time. The multi-objective aspects are dealt with a weighted sum approach based fitness function to evaluate the quality of the solution.
B. Firefly technique has been incorporated to address the optimal offloading strategy in Fog-Cloud environment, which selects a suitable computational device for each application.
C. Instead of the distance-based attractiveness, a modified attractiveness function is defined to measure the Firefly (computational server) attractiveness more accurately, which is determined automatically by using both the modified attractiveness function and distance from the brighter Fireflies (optimal computational servers).
D. Finally, the effect of the control parameters of the APSO technique is investigated thoroughly. We also evaluate the performance of the proposed algorithm over synthetic datasets using various performance metrics.

The rest of the paper is organized as follows. The related work of various offloading and resource allocation strategies in a Fog-Cloud environment is discussed in Section 2. The overview of the multi-objective optimization strategy and Firefly algorithm are presented in Section 3. The system model followed by the problem formulation is discussed in Section 4. The proposed EES algorithm is discussed in Section 5. The performance analyses of the proposed algorithm are discussed in Section 6. Finally, the conclusion is given in Section 7.

2. Related work

In this section, we review the merits and demerits of the existing offloading strategies in a Fog-cloud environment based on various QoS parameters. Liang et al. have developed an offloading strategy based on SDN technique in Fog-Cloud environment [11]. The proposed offloading strategy selects the best-fit computational server for each real-time application based on various performance metrics. The proposed method reduces the latency time of each application while ignoring other QoS parameters. Fricker et al. have proposed a load balancing strategy to trigger offloading the applications in a suitable Fog device [12]. The proposed method finds a suitable neighbor for each overloaded Fog device which may meet QoS constraints of the applications and balance the load of the network. The main objective of the proposed method is to balance the load among the Fog device with proper resource utilization. Craciunescu et al. have developed a latency-aware offloading strategy in a healthcare system [13]. The main objective of the algorithm is to improve the latency and minimize the response time of the tasks. This strategy may try to execute most of the task locally or Fog devices instead of the cloud data center. This may minimize the total latency delay and the computation time of the real-time applications.

Hasan et al. have developed an intensive-based offload strategy for IoT applications in a Fog-Cloud environment [14]. The authors have developed a mechanism which helps to create an ad-hoc cloud environment for IoT applications. The main purpose of the algorithm is to execute the IoT applications mostly in Fog devices which may minimize the energy consumptions of the computing servers and networks. Wang et al. have designed an intelligent offloading strategy between mobile devices and femto clouds [15]. Rather than sending the full application to the femto cloud, the mobile devices transmit the partial information to the cloud and shared the other information among the nearby devices. This strategy also minimizes the overall energy consumption of the computing servers and the network. Orsini et al. have developed a task offloading strategy in mobile cloud environment [16]. The authors have designed a hierarchical Fog-Cloud model which has four levels. The authors transmit the resource-intensive applications to the upper-level computing servers such as cloud data center and for the latency-intensive applications, the applications may execute locally or in the Fog devices. This may reduce the overall latency time and computation time of the applications.

Habak et al. are mostly emphasized on the underutilized Mobile devices or sensors rather than cloud [17]. The proposed method tries to offload most of the tasks locally or the nearby Mobile devices or sensors which have some computational capability, for minimizing the latency time and computational time of the applications. Mansouri et al. have designed an energy-aware and cost-efficient task offloading strategy in Fog-Cloud environment [18]. The proposed algorithm used the concept of game theory for finding a cost-efficient computational server which can minimize the computation time and the energy consumption of the servers. The authors applied the concept of Nash equilibrium over game theory for getting more accuracy for a section of the computational server. This may achieve multiple Quality of Experience parameters of the IoT users. He et al. have developed a task offloading strategy for assigning the IoT applications to the suitable computational server [19]. The authors have designed a multi-tire Fog-Cloud environment to achieve multiple QoS parameters of IoT applications. The proposed method minimizes the computational time of the IoT applications while meeting QoS constraint. Tan et al. have developed a latency-aware task offloading strategy in Fog environment [20]. The proposed approach selects a computational server which has less response time and nearer to the devices. This may minimize the total computational time and latency time of the real-time applications.

Most of the existing tasks offloading strategies try to optimize a single objective of the QoS parameters. However, single objective optimization may reduce the performance and efficiency of the cloud data center. Nowadays, most of the applications or tasks are generated from real-time devices IoT devices. So, to run such types of applications or tasks with minimum computational time and energy consumption is one of the challenging issues in a cloud environment, which have not discussed yet in the existing literature. To overcome the short comes of the existing strategies; we have developed an energy efficient offloading strategy for solving multiple objectives of the IoT applications. In this paper, we have designed a task offloading strategy using Firefly technique. Here, the IoT devices should offload the applications or tasks to the local Fog nodes or the centralized cloud data center based on the multiple QoS parameters including energy consumption and computational time. This may improve the accuracy of finding the optimal computational devices for each IoT application or task with minimum delay. The proposed EES strategy meets the multiple QoS parameters of the IoT applications including computation time and energy consumption. This may also minimize the CO_2 emission rate and Temperature of the computational servers of the cloud data center.

3. Preliminaries

Here, we first discuss the multi-objective optimization strategy followed by the overview of Firefly algorithm and the system model. Next, we elaborate on the workflow model followed by the problem formulation.

3.1. Multi-objective optimization

Multi-Objective optimization problems (MOPs) at time t are defined as follows

Minimize $\boldsymbol{F(x, t)} = \{f_1(x, t), f_2(x, t), f_3(x, t), f_4(x, t), ..., f_n(x, t)\}^T$

Subject to: $\boldsymbol{x} \in \theta$ where, $\boldsymbol{x} = (x_1, x_2, x_3, x_4, ..., x_m)^T$, is the m dimensional decision vector, θ is the decision space and $\boldsymbol{F} : \theta \to R^i : i = (1, 2, 3, \ldots, n)$, consists of n real-valued objective functions and R^n is called the objective space.

Definition 1. Decision Vector Domination: A decision vector x_1 Pareto dominates another vector x_2 at time t, denoted by $x_1(t) \succ x_2(t)$, if and only is,

$$\begin{Bmatrix} \forall i = 1, \ \ldots, \ M, \ f_i(x_1, \ t) \precsim f_i(x_2, \ t) \\ \exists i = 1, \ \ldots, \ M, \ f_i(x_1, \ t) \prec f_i(x_2, \ t) \end{Bmatrix} \tag{1}$$

Definition 2. Pareto Optimal Set: Let x and x^* are decision vectors, and if the decision vector x^* is said to be non-dominated at time t if and only if there is no other decision vector x such that $x(t) \succ x^*(t)$. The Pareto-Optimal Set (POS) is the set of all Pareto optimal solutions, *i.e.:*

$$POS = \{x^*(t) | \nexists x(t), \mathrm{x}(t) \succ x^*(t)\}$$

Definition 3. Pareto-optimal Front: Pareto-optimal Front (POF) is the corresponding objective vectors of the POS at time t.

$$POF = F\{x^*(t) | x^* \epsilon \ POS$$

An ideal multi-objective optimization strategy must have the ability to find a set of optimal solutions and at the same time, the solutions must be as diverse as possible.

3.2. Firefly algorithm

Biologically inspired algorithms such as particle swarm optimization, Firefly algorithm, etc., are become powerful for optimizing modern numerical problems. Among these biology-derived algorithms, Firefly algorithm is one of the important optimization techniques [21,22]. Firefly algorithm is proposed by Xin-She Yang in 2007 at Cambridge University, inspired by the flashing behavior of Fireflies. Different species of Fireflies produce short and rhythmic flashes which are often unique to particular species. The two important variables in Firefly algorithms are the variation of light intensity and the attractiveness of the Fireflies. Here the attractiveness of Firefly depends on the brightness of the Fireflies. The brightness of a Firefly is chosen as $I(i) \infty f(i)$, where i is a location of a Firefly. Another variable, the attractiveness (β) of a Firefly is relative and depends on the judgment of other Fireflies. Thus it will vary with the distance v_{ij} between Firefly i and Firefly j. In addition, the light intensity is inversely proportional to the distance between the Fireflies and the light is absorbed by the media. The Cartesian distance between two Fireflies i and j at f_i and f_j respectively are mentioned in below equation.

$$v_{ij} = \left\| f_i - f_j \right\| = \sqrt{\sum_{k=1}^{d} (f_{i,k} - f_{j,k})^2} \tag{2}$$

Here $f_{i,k}$ is the kth component of the special coordinate f_i of the ith Firefly. In 2-D case, the Cartesian distance between two fireflies is represented as-

$$v_{ij} = \sqrt{(f_i - f_j)^2 + (g_i - g_j)^2} \tag{3}$$

The movement of Firefly i towards the brighter Firefly j due to its attractiveness is shown below

$$f_i = f_i + \beta_0 e^{-\gamma v_{i,j}^2} (f_j - f_i) + \alpha \varepsilon_i \tag{4}$$

where β_0 is the initial attractiveness at $r = 0$. Here the second term and third term of Eq. (4) represent the attraction and randomization walk of Firefly via Levy flights respectively. α being the randomization parameter in the interval [0, 1] and ε is a vector of random numbers drawn from the uniform distribution. Finally, the parameter γ represents the variation of the attractiveness, and its value helps to determine the speed of the convergence of the algorithm. In most cases, the values of, γ vary [0.01, 100]. The values of α and γ vary based on the movement of the Firefly from the initial location to the updated position in 2-D space. In this algorithm, randomly generated solutions are considered as fireflies. The brightness of the Firefly is adjusted over the iterations of the optimization phase. Fig. 1 schematically represents the updating path of a Firefly in a 2-D space with 12 Fireflies. The Fireflies are sorted in the 2D space according to their objective function. For example, Fireflies 1–5 is brighter than Firefly 6. According to Eq. (4), Firefly 6 changes its position and moves towards the Fireflies1–5 and eventually reaches to its final position. Therefore, the relocation of a Firefly depends on its objective at the initial position. The triangle and the solid line of the figure represent the updated position and the updated path of the Firefly. The solid circles represent the brighter Fireflies (1–5) whereas the hollow circles represent the remaining Fireflies.

The major two advantages of Firefly algorithm over existing meta-heuristic scheduling strategies are: (i) automatically subdivision and (ii) ability to deal with multimodality. First, the Firefly algorithm uses the global best of the individuals for convergence based on attraction and reduces the randomness as iteration proceeds. This leads to subdivide the population into subgroups and each subgroup can swarm around each local optimum. Among these local optimal, Firefly algorithm easily finds the global best solution. Secondly, the subdivision allows the fireflies to find all the optima simultaneously if the population size is sufficiently higher than the number of local optima. The whole population can be subdivided into subgroups with a given average distance. This automatic subdivision ability makes the Firefly algorithm more suitable for clustering and classification problems, multimodal and combinatorial optimization problems. The Firefly algorithm also combines both the local and global search capabilities of the problem and guaranteed to produce the global convergence.

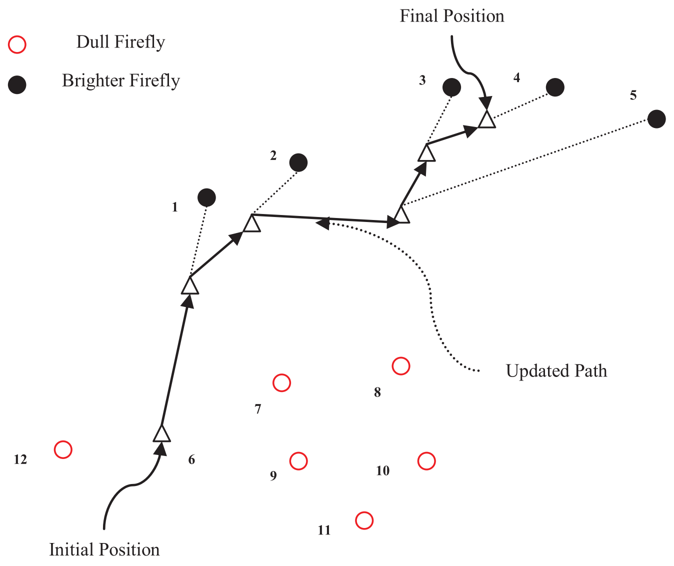

Fig. 1. Firefly updated path.

4. System model and problem formulation

4.1. Transmission model

We consider that there are N IoT users denoted by the set $I = \{I_1, I_2, I_3, ..., I_N\}$. Each IoT user $I_n \in I$ submits an IoT application T_n. Here, we consider each IoT device has a processing unit (i.e. microprocessors, FPGAs, SOCs, microcontrollers) and software applications which has some computational ability. So, the applications may be executed locally or to be offloaded to the computational servers, i.e. Fog devices ($F = \{F_1, F_2, F_3, ..., F_M\}$) or cloud servers ($C = \{C_1, C_2, C_3, ..., C_O\}$) based on the objective function. Let, P computational servers are deployed in an area, denoted by the set $S = \{S_1, S_2, S_3, ..., S_G\}$where $G \in \{C, F\}$, and the local servers are connected by the Local Area Network, WiFi, Bluetooth or Long Term Evolution (LTE). The IoT devices generate k number of applications according to the Poisson process which are categorized as delay-intensive and resource-intensive applications and the applications are executed locally or to the computational servers. The transmission time is incurred during offload the applications to the computational servers through an uplink channel or transmit back to the requesting IoT devices through a downlink channel. In a noise-free channel, the transmission time between users and computational servers depends on the distance and the bandwidth of the network. The transmission time between the user a and computational servers i (TT_{ai}) is defined as follows.

$$TT_{ai} = PD_{ai} + SD_{ai} : a \in N \tag{5}$$

where PD_{ai} and SD_{ai} represent the propagation delay and serialization delay. The propagation delay (DS_{ai}) is defined as the ratio between the distance among the user a and computational servers i and the bandwidth of the network (BW_{ai}), i.e. $PD_{ai} = \frac{DS_{ai}}{BW_{ai}}$, where $DS_{ai} = \sqrt{(X_a - X_i)^2 + (Y_a - Y_i)^2}$ in a two-dimensional space (X, Y). The serialization delay is the ratio between the sizes of an application k emitted from a device a, $SZ_{ka} \in [0, SZ_{max}]$ to the transmission rate of the network (TR_{ai}), i.e. $SD_{ai} = \frac{SZ_{ka}}{TR_{ai}}$. However, during data transmission, some noise may be added with the actual data, which may change our assumption. So, we can consider that each computational servers i consumes a fixed transmission power (P_i), and the data are transmitted over an orthogonal channel, and the achievable transmission rate TR_{ai} is given by Shannon Capacity,

$$TR_{ai} = BW_{ai} log_2\left(1 + \frac{P_i H_{ai}^t}{\delta^2}\right) \tag{6}$$

where ${H_{aj}}^t$ is the channel gain between user a and computational servers i and δ^2 represents the noise power. Then, the amount of time required to uplink the data from the user a to computational servers i (${TT_{ai}}^u$) is $TT_{ai}^u = \frac{P_i SZ_{ka}}{TR_{ai}}$. However, the downlink traffic, from computational servers i to the user a, consists of the computational result $SZ_{ik} \in [0, SZ_{max}]$ and a few communication data a $CD_{ai} \in [0, CD_{max}]$. Then, total time requires to transmit the downlink traffic is $TT_{ia}^d = \frac{P_i(SZ_{ki} + CD_{ai})}{TR_{ia}}$, where TR_{ia} is the transmission rate of and SZ_{kj} is the size of the application k, emitted from computational servers i. So, the total transmission time is the combination of uplink time to transmit data from users to computational servers and users and the downlink time to forward the data from computational servers to users, which is formulated as

$$TT_{ak} = TT_{ai}^{u} + TT_{ia}^{d} = \frac{P_i SZ_a}{TR_{ai}} + \frac{P_i(SZ_k + CD_{ai})}{TR_{ia}} \tag{7}$$

4.2. Computational model

The computational time of an IoT application depends on the capacity of the computational device and total transmission time required for transmitting the applications and receiving the applications from a computational server which is defined as follow.

$$CT_{ki}^{a} = ET_{ki}^{a} + TT_{ik} : \; i \in G, \; k \in N \tag{8}$$

Here, $ET_{ki}{}^{a}$ represents the execution time of application k emitted from a device a in computational device i, which is defined as $ET_{ki}^{a} = \frac{SZ_k}{CP_i}$, where CP_i is the CPU capacity of a computational device i. Here, the computational device for IoT based applications is either the IoT device itself or the computational server $i \in N$. IoT devices and the Fog devices have some limited processing and storage capacity to execute the small delay-intensive applications, also referred to as local computing, otherwise they can offload the resource-intensive applications to the cloud data center for further processing, also denoted as remote computing. An application can able to execute in a computational device if the remaining resource capacity (also referred as the remaining load capacity) of the device meets the resource requirements of the application. The current load of a device is defined as the percentage of the resources have been utilized within a specific time interval (T' to T''). Here we consider each computing device is configured based on two types of resources i.e. CPU ($C_i(t)$) and memory ($M_i(t)$). The current load of a computational device c at each time interval is defined as follows.

$$L_i(C, M) = \beta \times \sum_{t=T'}^{T''} C_i(t) + (1-\beta) \times \sum_{t=T'}^{T''} M_i(t) \tag{9}$$

where, β is the constant in the interval [0, 1]. The value of $C_i(t)$ and $M_i(t)$ are represented in the interval [0,100]. If the remaining load of the IoT device itself or the local Fog device i, $RL_i(C,\ M) = (1 - L_i(C, M))$is less than the required load ($TL_k(C,\ M)$) by the application T_k: $k \in \{N\}$, i.e. $RL_i(C, M) \le TL_k(C,\ M)$), then the applications should be assigned to that device. Otherwise, the application should be offloaded to the cloud data center. The aim should be to keep the resource utilization of an individual device within the permissible level. Utilization is another key decision parameter used to indicate whether a device capacity is adequate to assign a new application or not. However, very high resource utilization often compromises the performance of the system and may increase the completion time of the applications. For a given computational server i, the resource utilization, RU_i, is calculated as follows.

$$RU_i = \frac{L_i(C, M)}{L_i^{\max}(C, M)} \times 100 \tag{10}$$

where $L_i{}^{\max}(C,\ M)$ indicates the maximum resource capacity of the computational server i.

4.3. Energy model

The energy consumption of an IoT application mainly depends on two factors: (i) Computational energy: consumes energy based on the amount of time the computing resources are busy to execute the application and (ii) Transmission energy: produces energy based on the time required to transmit the application to the computational server and receive the computational results from that computational server. The IoT devices may execute the small delay-intensive applications locally or the local Fog devices which minimize the transmission time as well as the energy consumption for transmission. However, for the resource-intensive applications for the IoT devices should be offloaded to the cloud data center which consumes computational energy and transmission energy for executing the applications.

4.3.1. Computational energy

The computational energy consumption varies greatly depending on the workload of the computing devices. The power consumption of a device is determined mainly by the processors and memory and the power consumption of the resources is determined primarily by resource utilization. However, the energy consumption of a device is estimated based on the utilization of the resources and the execution time of the applications. The amount of energy consumed by the IoT device or the computational servers is defined as follows.

$$TP_i^C = \begin{cases} DP_i^C + SP_i^C \; : \; When\, CPU\, executed\, a\, task \\ SP_i^C \; : \; When\, CPU\, is\, active\, state\, without\, executing\, a\, task \\ 0 : \; When\, CPU\, is\, in\, idle\, state \end{cases} \tag{11}$$

where, DP_i^C and SP_i^C represents the dynamic power and static power consumption by the processors respectively. The dynamic power of a device depends on the frequency (f) and the voltage (V_{dd}) of the cores of the CPU while executing an application. The dynamic power consumption of a computational server is defined as $DP_i^C = \sum_{a=1}^{CR} \beta f V_{dd}^2$. Here CR represents the number of cores of the processor and β is a constant, where $\beta = C_l N\alpha$.; C_l and N represent the capacity and the number of logic gates available in each core and α is constant where $\alpha \leq 1$. Similarly, the static power consumption of a CPU is defined as $SP_i^C = dV_{dd}$, where d is a constant. The power consumption of the memory depends on the read write operation while executing an application on an IoT device or a computational server is defined as follows

$$TP_i^M = \begin{cases} DP_i^{MW} : \mathit{When\ memory\ performs\ write\ operation} \\ DP_i^{MR} : \mathit{When\ memory\ performs\ read\ operation} \\ DP_i^{MW} + DP_i^{MR} + A_M : \mathit{When\ memory\ performs\ read\ and\ write\ operation} \\ A_M : \mathit{When\ memory\ is\ in\ idle\ state} \\ 0 : \mathit{When\ memory\ is\ in\ idle\ state} \end{cases} \tag{12}$$

where, DP_i^{MR} and DP_i^{MW} represent the dynamic power consumption by the memory while performing reading and writing operation and A_M represents the static power required by the memory during an ideal state. The dynamic power consumption by the memory during read and write operation is defined as $DP_i^{MW} = DP_i^{MR} = \frac{1}{2} c V_{dd}^2 f$, where c is a constant. So, the total power consumed by the computational server i in each unit time is defined as follows.

$$TP_i = TP_i^C + TP_i^M \tag{13}$$

So, the total computational energy (TCE_k) by an application T_k, executing in a computational server i is $TCE_k = TP_i \times ET_{ki}^a$.

4.3.2. Transmission energy

The transmission energy of an application depends on the amount of bandwidth consumed by the applications during offload the application to the computational server i through an uplink network and transmits back the application to the user through a downlink channel. Here, we consider the bandwidth consumes by the application for uplink and downlink channel between user a and computational server i is BW_{ai} and BW_{ia} respectively. The total time requires to transmit the data from the user a and computational server i and vice-versa is ${TT_{ai}}^u$ and ${TT_{ia}}^d$, respectively. So the total transmission energy (TTE_k) consumed by an application T_k is.

$$TTE_k = \left(BW_{ai} \times TT_{ai}^u\right) + \left(BW_{ia} \times TT_{ia}^d\right) \tag{14}$$

So, the total energy consumes by an application while T_k executing in a user a and computational server i is

$$TEC_{ki} = (TCE_k + TTE_k) = (TP_c \times ET_{ki}^a) + \left(BW_{ai} \times TT_{ai}^u\right) + \left(BW_{ia} \times TT_{ia}^d\right) \tag{15}$$

The power consumption of a device also affects two parameters: the CO_2 emission rate and the temperate of the device. The relationship between power consumption and CO_2 emission is an important factor to account in consideration of the sustainability of a device. The CO_2 emission rate of the CPU ($CE_i^C(t)$)and memory ($CE_i^M(t)$) from a computational server i at time t is defined as follows.

$$\begin{aligned} CE_i^C(t) &= \gamma_{CO_2}^C \times TP_i^C \\ CE_i^M(t) &= \gamma_{CO_2}^{mm} \times TP_i^M \end{aligned} \tag{16}$$

where, $\gamma_{CO_2}^C$ and $\gamma_{CO_2}^M$ are two user-defined constants. So, the total CO_2 emission rate of a computational server i is defined as follow.

$$TCE_i = CE_i^C(t) + CE_i^M(t) \tag{17}$$

Similarly, the relationship between the temperature and power emission of the CPU ($T_i^C(t)$) and memory ($T_i^M(t)$) of a device is defined as follows.

$$\begin{aligned} T_i^C(t) &= \sqrt{\frac{TP_i^C}{\alpha V_{dd}}} \\ T_c^M(t) &= \sqrt{\frac{TP_i^M}{\alpha V_{dd}}} \end{aligned} \tag{18}$$

where α is a user-define constant. So, the total temparature (TM_i) emitted by a computational server i is defined as follow.

$$TM_i = T_i^C + T_i^M \tag{19}$$

4.4. Problem formulation

In this work, the applications may execute locally or offload the applications to the cloud data center for processing. The main focus of the paper is to minimize the energy consumption and computational time of the IoT applications while minimizing CO_2 and temperature emission of the computational devices. These objectives can be achieved by executing the applications to a suitable computational server. Consequently, the optimization of the applications is formulated as a bi-objective optimization problem which is discussed below.

Minimize TEC_{ki}
Minimize $CT_{ki}{}^{a}$
Subject to
$TEC_{ki} \le \theta_1$ .... (i)
$TCE_i \le \theta_2$ .... (ii)
$TM_i \le \theta_3$ .... (iii)
$L_i(C,\ M) \le \theta$ .... (iv)
$L_i(C,\ M) \le L_i^{max}(C,\ M)$ .... (v)
$RU_i \le 100$ .... (vi)
$x_{ki} \in \{0,\ 1\};\ k \in N,\ i \in G$ .... (vii)
$\sum_{i=1}^{G} x_{ik};\ k \in N,\ i \in G$ .... (viii)

Constraint (i) indicates that the total power consumption of a device i must not beyond a threshold value θ_1. Constraint (ii) represents that the total CO_2 emission rate of a device i must not beyond a threshold value θ_2. Constraint (iii) indicates that the total temperature consumption of a device i must not beyond a threshold value θ_3.Constraint (iv) indicates that a load of a computing device i must not beyond a threshold value θ. Constraint (v) represents that a load of a computing device i must not beyond the maximum capacity of that server. Constraint (vi) indicates that the resource utilization of a computing device i must not beyond 100%. Constraint (vii), x_{ki} denotes whether an application T_k is assigned to a computational server i. Constraint (viii) describes that each computational server can execute multiple numbers of applications concurrently.

5. Energy efficient offloading strategy (EES)

The basic Firefly algorithm is mainly used to solve the single objective optimization problem. However, to solve MOPs, we use the weighted sum method to find the optimal computational server and deploy the application on that computational server. The detail of the algorithm is discussed below. As we discussed earlier that some of the IoT applications may execute locally if the micro-services of the IoT devices meet the resource requirements; otherwise the tasks should be offloaded to the cloud data center for further processing. Initially, this strategy generates a multi-objective optimization function (referred as Fitness Function) based on the two QoS parameters i.e. total energy consumption and the computational time, defined in Section 3. The total energy consumption of the executing devices are denoted as

$$\boldsymbol{TEC}_z(d_z, D) = \{TEC_1, TEC_2, TEC_3, TEC_4, ..., TEC_z\},\ \text{where}\, z \in \{N\}$$

and the computational time of those are represented as

$$\boldsymbol{CT}_z(d_z, D) = \{CT_1, CT_2, CT_3, CT_4, ..., CT_z\},\ \text{where}\, z \in \{N\}$$

The Fitness Function places all the two QoS parameters into a single one based on weighted-sum approached which is defined below.

$$fit(d_z,\ D) = \alpha_1.TEC_z(d_z) + \alpha_2.CT_k(d_z) \tag{20}$$

Thus the offloading objectives are aggregated in a single Fitness Function, which measures the degree of the optimal executing device. In Eq. (20), $fit_z(d_z,\ D)$ represents the Fitness solution of the selection of the suitable computing device $k \in \{N\}$ for each requested application. In the above equation, the coefficients α_1, and α_2 are the weights which are used to indicate the priority of the objectives whose value lies between [0, 1]. For minimum optimization problem, the light intensity of the computing server IS (represented as Firefly) at a particular location x can be chosen as $IS(x) \propto FF(x)$. The light intensity of the computing server position varies with the distance r and is expressed by the following equation.

$$IS_i(r) = IS_{0i}.e^{-\gamma r^2} \tag{21}$$

where IS_0 denotes the light intensity of the source computing server, S_i is an index of the computing server and γ represents a fixed light absorption coefficient. Usually, the light intensity for the minimization problem is associated with the inverse of the Fitness Function. The attractiveness AS depends on the distance r and is defined as follows.

$$AS_i(r) = AS_{0i}.e^{-\gamma r^2} \tag{22}$$

During finding an optimal executing device each objective of particle d_z is normalized based on the maximum and minimum values of the corresponding objective function. Such normalized objective function helps to eliminate the impact of various amplitudes on multiple objectives. The normalized objective $FN_r(d_i)$ (where r represents the total number of objectives) of d_i are obtained using

$$FN_r(d_i) = \frac{f_r(d_i) - f_r^{\min}}{f_r^{\max} - f_r^{\min}} \tag{23}$$

where ${f_r}^{max}$ and ${f_r}^{min}$ represent the maximum and minimum values of the rth objective which are obtained from the non-dominated solutions. Finally, the Firefly algorithm pointed the best-fit computing server in the plane with minimum ED value, is defined as

$$gbest(d_i) = \min_{j \neq i} \sqrt{\sum_{x=1}^{2} ED(f_r(d_i),\ f_r(d_j))} = \sqrt{\sum_{x=1}^{2} (f_r(d_i) - f_r(d_j))^2} \tag{24}$$

where ED value $ED(d_i,\ d_j)$ between two particles d_i and d_j in 2-D space is defined as

$$ED(f_r(d_i),\ f_j(d_i)) = \begin{cases} f_r(d_j) - f_r(d_i)\ if\ (f_r(d_j) > f_r(d_i)) \\ 0 \qquad\qquad Otherwise \end{cases} \tag{25}$$

The above total energy consumption and computational time are dynamically balancing two weight factors α_1 and α_2. Those factors are adaptively adjusted for different particles based on the objectives. For this purpose, the average value of all $TEC_{\mathbf{z}}(d_z,\ D)$ and $CT_{\mathbf{z}}(d_z,\ D)$ in the particle swarm d_z are calculated by

$$\overline{TEC}(d_z,\ D) = \frac{\sum_{i=1}^{Z} TEC(d_i, D)}{Z}\ and\ \overline{CT}(d_z,\ D) = \frac{\sum_{i=1}^{Z} CT(d_i, D)}{Z} \tag{26}$$

Finally, the proposed EES algorithm is shown in Fig. 2.

Thus, it is expected to enhance the convergence speed of selection of executing component of EES strategy and finds the suitable computational server for each IoT application which meets multiple QoS objectives.

6. Performance evaluation

In this section, we investigate the performance of the proposed EES strategy by evaluating various QoS parameters such as computational time, energy consumption, CO_2 emission, Temperature emission, and resource utilization. We further compare the proposed method with existing algorithms proposed in [18–20].

6.1. Simulation setup

The simulation parameters of the proposed EES strategy are summarized in Table 1. Here we consider 15 IoT and each the devices are capable to generate multiple tasks or applications simultaneously. Each IoT device has some processing and storage capacity which is randomly and uniformly taken as [1000–4000] MIPS and [256–512] MB, respectively. Each IoT device randomly generates multiple numbers of tasks with sizes lay in [7000–100,000] MI. We assume that each computing device is equipped with three different wireless interfaces, Long Term Evolution (LTE), Wifi and Bluetooth. The IoT and Non-IoT devices may use the LTE connection for long-term communication such as the cloud data center, while they use Wifi and Bluetooth interfaces to connect with the other IoT devices or Fog nodes. Here, we assume that the transmission rate of the LTE and Wifi interfaces are randomly and uniformly distributed over [4.85, 6.85] Mbps and [2.01, 4.01] Mbps, respectively. Here, we consider two cloud data center where each of them has enough processing and storage capacity. The CPU capacity and memory usage of each cloud server lay in [12,000–30,000] MIPS and [2018–4096] MB, respectively.

6.2. Dataset used

In this section, we discuss the dataset used to evaluate the performance of the proposed method. Here we generate two different real-time datasets where each of the datasets generates the energy consumption and the computational time

Table 1
Simulation parameters.

Parameter description	Values
Number of IoT devices	15
Number of IoT applications	100
Number of Fog nodes	8
Number of Servers	4
Number of cloud data centers	1
CPU capacity of IoT devices	1000–4000 MIPS
RAM size of IoT devices	256–512 MB
CPU capacity of Fog nodes	4000–8000 MIPS
RAM size of Fog nodes	512–1024 MB
CPU capacity of each cloud server	12,000–30,000 MIPS
RAM size of each cloud server	2048–4096 MB
Storage capacity of each cloud server	100 GB
Sizes of the IoT applications	4000–100,000 MI

EES Algorithm:
Input: Set the objectives of the computational servers; Set the user defined constants: n, I_0, α, γ
Output: Find the optimal computational servers

```
1:  Begin
2:   For each j: 1 to N do                         // r = Maximum number of iterations
3:       $FN_j$ = Initial Solution()
4:   End For
5:   While ($FN_j <= \Delta$) do                  // $\Delta$= Threshold value
6:       min = arg min $FN_j$
7:         For each j: 1 to N do
8:             For each l: 1 to N do
9:                 \\ Calculate the modified attractiveness function
                        $FN_r(d_i) = \frac{f_r(d_i) - f_r^{\min}}{f_r^{\max} - f_r^{\min}}$
10:                If ($FN_j < FN_l$)
11:                    \\ Find the Euclidean distance between the servers
                        $ED(d_i, d_j) = \sqrt{\sum_{x=1}^{2} (f_r(d_i) - f_r(d_j))^2}$
12:                    Compare the Pareto dominance relationship between each pair of
                       the servers and put the non-dominated ones.
13:                    Randomly select one server from the set as the attractive one
14:                    \\ Update the position of the computational server
                        $d_{i+1} = d_i + \beta_0 e^{-\gamma r_{ij}^2}(FF_i - FF_j) + \alpha\varepsilon_i$
15:                End if
16:            End for
17:        End for
18:        $FN^{min} = FN^{min} + \alpha$. Rand(1/2)
19:        Find the optimal server
20:    End for
21:  End
```

Fig. 2. The pseudo code of the EES algorithm.

for the executing devices. For real-time analysis, we have considered the Raspberry Pi and mobile devices as Fog nodes and various types of IoT sensors as the IoT devices which generate the applications or tasks at a regular time interval. The important parameters of the Firefly algorithm are α, β, and γ and we assumed their possible values are 0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85 and 1.0. In the experiment, the size of the population ranges from 20, 30, 40, 50, 60, 70, 80, 90 and 100. An empirical test is performed to fix the values of the parameter. The proposed EES algorithm is simulated with a fixed value of each parameter for 100 iterations and minimum error was recorded with 1000 independent runs.

A fitness value of each pattern in a dataset is calculated based on two QoS parameters, shown in Eq. (20) and performs a significance test between the dataset to measure the reality of the data. The significant level of the datasets is measured based on *P*-value analysis with an unpaired *t*-test. The unpaired *t*-test assumes that the data have been sampled from the normally distributed population. The *P*-value is the probability of finding a result equal to or more extreme than the observed data when the null hypothesis (H_0) is true. *P*-value is less than the selected significance level then the null hypothesis is rejected and supported the alternative hypothesis with proper evidence. The choice of significance level at which we reject H_0 is arbitrary. Conventionally the 5%, 1% and 0.1% ($P < 0.05$, 0.01 and 0.001) levels have been used. Most researchers refer to statistically significant $P < 0.05$ and statistically highly significant $P < 0.001$. From Table 2, it is clearly observed that all the datasets are highly significant. So we can conclude that all the datasets are valid.

Table 2
Significant test of the datasets (DSs) based on *P*-value.

	DATASET- 1	DATASET- 2	DATASET- 3	DATASET- 4	DATASET- 5	DATASET- 6
DS-1	0	4.16421E−08	3.5324E−08	6.64E−07	0.000287	3.1346E−07
DS-2	1.2539E−07	0	2.35E−07	2.46E−07	7.37E−10	6.64E−07

Table 3
Minimum and mean error of β.

Dataset	Minimum error								
	$\beta=0.15$	$\beta=0.25$	$\beta=0.35$	$\beta=0.45$	$\beta=0.55$	$\beta=0.65$	$\beta=0.75$	$\beta=0.85$	$\beta=1.0$
DS-1	2.729E−06 (0.026502)	1.68E−05 (0.01091)	9.22E−06 (0.02273)	8.13E−07 (0.02069)	1.03E−05 (0.015191)	**2.1E−06 (0.007381)**	2.42E−06 (0.01006)	3.82E−06 (0.01143)	2.78E−05 (0.01476)
DS-2	3.88E−05 (0.008896)	3.7E−06 (0.01786)	7.78E−06 (0.02295)	5.14E−06 (0.01856)	1.31E−05 (0.028192)	**1.08E−06 (0.02665)**	1.66E−05 (0.02642)	1.47E−06 (0.02051)	4.36E−06 (0.01977)

6.3. Parameter analysis and discussion

In this section, we analyze the values of the parameters of the proposed EES strategy for observing the better quality of the solutions. For fixing the values of the parameters of the EES strategy, we calculate the minimum, mean and maximum error of the populations. Here, we use Euclidean distance for calculating the minimum error ($MinErr\ (D^*, d_z)$) and maximum error ($MaxErr\ (D^*, d_z)$) between the best-position of a particle (D^*) and the other particles (d_z), $z \in Z$ in the dataset, which is shown below.

$$MinErr(D^*, d_z) = \min(ED_i(D^*, d_z)) = \min\left(\sqrt{\sum_{i=1}^{2}(D^* - d_z)}\right)$$
$$MaxErr(D^*, d_z) = \max(ED_i(D^*, d_z)) = \max\left(\sqrt{\sum_{i=1}^{2}(D^* - d_z)}\right) \quad (27)$$

The mean error ($\overline{Err}(D^*, d_z)$) of the a, $a \in N$ number of the computational server is calculated as follow.

$$\overline{Err}(D^*, d_z) = \frac{(ED_z(D^*, d_z)}{a} = \frac{\sqrt{\sum_{i=1}^{2}(D^* - d_z)}}{a} \quad (28)$$

Analysis of β value: Beta parameter of EES algorithm is one of the most important parameters and the performance of the proposed algorithm varies for the different values of the parameter. The fluctuation in the performance due to the different values of the beta parameter can be easily observed from Table 3. In Table 3, the minimum error for different values of Beta is shown. All the dataset used for the experiment with the proposed method produces the minimum value if the beta value is set to 0.65. Hence, for the rest of the experiment, the beta value is fixed at 0.65.

Analysis of α value: Alpha parameter of EES algorithm is one of the most important parameters and the performance of the proposed algorithm varies for the different values of this parameter. The fluctuation in the performance due to the different values of the Alpha parameter can be easily observed from Table 4. In Table 4, the minimum error for different values of Alpha is shown. All the dataset used for the experiment with the proposed method produces the minimum value if the alpha value is set to 0.15. Hence, for the rest of the experiment, the alpha value is fixed at 0.15.

Analysis of γ value: Gamma parameter of EES algorithm is another important parameters and the performance of the proposed algorithm varies for the different values of this parameter. The fluctuation in the performance due to the different values of the gamma parameter can be easily observed from Table 5. In Table 5, the minimum error for different values of gamma is shown. All the dataset used for the experiment with the proposed method produces the minimum value if the gamma value is set to 0.65. Hence, for the rest of the experiment, the gamma value is fixed at 0.65.

Table 4
Minimum and mean error of α.

Dataset	Minimum error								
	$\alpha=0.15$	$\alpha=0.25$	$\alpha=0.35$	$\alpha=0.45$	$\alpha=0.55$	$\alpha=0.65$	$\alpha=0.75$	$\alpha=0.85$	$\alpha=1.0$
DS-1	**2.52E−06 (0.01274)**	1.701E−05 (0.01554)	2.542E−05 (0.01751)	4.34E−05 (0.01505)	2.64E−06 (0.014222)	3.63E−05 (0.02454)	1.9162E−05 (0.013341)	2.28E−05 (0.015994)	4.68E−05 (0.0370)
DS-2	**1.27E−06 (0.01815)**	1.08E−05 (0.01572)	1.58E−05 (0.02983)	4.34E−06 (0.013121)	2.63E−06 (0.015777)	8.16E−06 (0.037883)	8.36E−05 (0.025652)	3.44E−05 (0.027234)	3.83E−06 (0.0190)

Table 5
Minimum and mean error of γ.

Dataset	Minimum error								
	$\gamma = 0.15$	$\gamma = 0.25$	$\gamma = 0.35$	$\gamma = 0.45$	$\gamma = 0.55$	$\gamma = 0.65$	$\gamma = 0.75$	$\gamma = 0.85$	$\gamma = 1.0$
DS-1	2.73E−06 (0.02650)	1.68E−05 (0.01091)	9.22E−06 (0.02270)	8.13E−07 (0.02069)	1.03E−05 (0.01509)	**4.1E−07 (0.00738)**	2.42E−06 (0.00999)	3.82E−06 (0.01143)	2.78E−05 (0.01476)
DS-2	3.88E−05 (0.008896)	3.7E−06 (0.01786)	7.78E−06 (0.02295)	5.14E−06 (0.01856)	1.31E−05 (0.02796)	**1.08E−06 (0.02665)**	1.66E−05 (0.02618)	1.47E−06 (0.02051)	4.36E−06 (0.01977)

Table 6
Minimum and mean error of Pop_Size parameter.

Dataset	Minimum error								
	Pop_Size PS = 20	Pop_Size PS = 30	Pop_Size PS = 40	Pop_Size PS = 50	Pop_Size PS = 60	Pop_Size PS = 70	Pop_Size PS = 80	Pop_Size PS = 90	Pop_Size PS = 100
DS-1	2.56E−05 (0.17029)	4.39E−05 (0.05867)	1.02E−05 (0.03204)	1.69E−05 (0.05117)	**1.16E−05 (0.060961)**	1.74E−05 (0.05819)	6.81E−05 (0.04274)	4.31E−05 (0.04783)	4.49E−05 (0.0309)
DS-2	3.38E−05 (0.11809)	6.71E−06 (0.09632)	4.27E−05 (0.05754)	1.7E−05 (0.03495)	**3.83E−06 (0.03390)**	4.65E−06 (0.02170)	7.57E−06 (0.03088)	1E−05 (0.01380)	2.325E−05 (0.05682)

Table 7
Minimum and mean error over datasets.

Datasets	Error of datasets	
	Minimum error	Mean error
DS-1	3.03644E−07	0.020608693
DS-2	2.99E−07	0.038654

Analysis of population size: Population Size parameter of EES algorithm is another important parameter and the performance of the proposed algorithm varies for the different values of this parameter. The fluctuation in the performance due to the different values of the *Pop_Size* parameter can be easily observed from Table 6. In Table 6, the minimum error for different values of *Pop_Size* is shown. All the dataset used for the experiment with the proposed method produces the minimum value if the *Pop_Size* value is set to 60. Hence, for the rest of the experiment, the Pop_Size value is fixed to 60.

Result analysis: The experiment was conducted with the fixed parameter values of alpha, beta, gamma, and population size which are decided by the analysis. The experiment was conducted to prove the convergence speed, stability and solution quality of the proposed method. The convergence of the proposed method over different synthetic dataset is presented in Table 7. The convergence is the minimum Euclidean distance between the target result and the optimized result produced by the proposed method. So, the main target of all the conducted experiment is to achieve the minimum error. It is observed from the experiment and the data presented in Table 7 that the error of the proposed algorithm is almost close to zero for all datasets. It is self-explanatory that the stability in the performance is achieved through considered parameter values which are very helpful for the detection of the best-fit computational server for each IoT application. The performances of the proposed method over the extremely significant dataset are less than the predefined threshold value 0.0001 and the proposed method achieved the target level for all the datasets considered for the experimental purpose.

6.4. Comparison analysis

In this section, we evaluate the proposed EECS strategy via simulation runs over the two real-time datasets and compare with the existing algorithms such as OLD [18], MFC [19] and COG [20]. Here, we consider various performance metrics to evaluate the proposed strategy such as computational time, energy consumption, CO_2 emission, and Temperature emission.

6.4.1. Computational time

Computational time of a task is defined as the amount of time requires to complete its execution within a computational server. The computational time depends on the executing time of a task in the assigned resources and the time requires to transmit the task from the IoT devices to the Fog nodes or the cloud data center. As in Section 4, we already discussed that the computational time of the IoT devices. The proposed EES strategy finds a suitable computational server based on Firefly algorithm for each task and assigns the task to that server. However, the existing multi-objective scheduling strategies deploy the tasks to suitable Fog devices without considering the multiple objectives of the IoT applications or tasks. The comparative analysis between the EES algorithm and the existing state-of-art-algorithms in term of computational time over various numbers of tasks are shown in Fig. 3. The average computational time of EES strategy is better than the OBJ 18%, the MFC algorithm by 23%, and the COG algorithm by 27%. Based on the experiments, the EES strategy outperforms than other scheduling algorithms for different synthetic datasets.

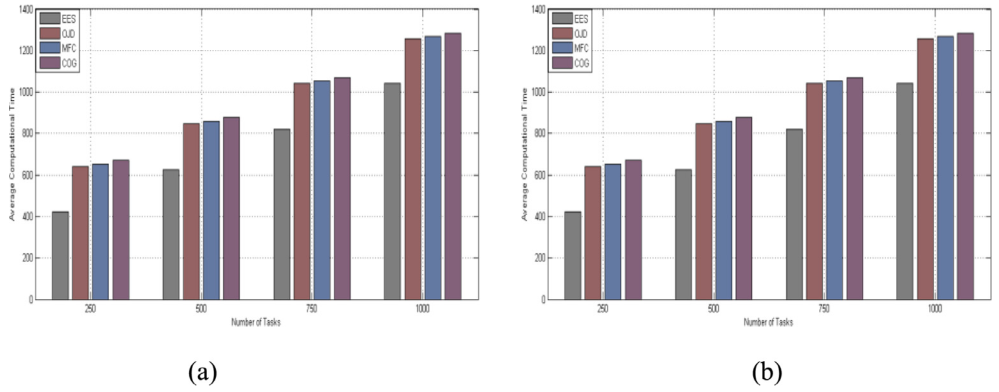

Fig. 3. Average computational Time of different state-of-arts-algorithms: (a) DS-1 and (b) DS-2.

Table 8
Statistical analysis of computational time.

Datasets		Computational Time for IoT and Non-IoT based tasks			
		EECS	MOBFOA	PSO_COGENT	PSO
DS- 1	**Minimum**	10	21	24	29
	Mean	35.12	47.17	49.34	55.45
	Maximum	65	91	94	104
	SD	5.495657	10.43240	11.34250	14.56231
DS- 2	**Minimum**	11	24	27	33
	Mean	37.54	49.23	51.92	58.12
	Maximum	71	92	97	110
	SD	4.215431	11.43245	12.43516	16.32187

The statistical analysis of the proposed EES strategy and the existing multi-objective scheduling algorithms in a cloud environment is shown in Table 8. This may prove the efficiency of the EES strategy over the existing ones in term of computational time. The metrics of minimum value, mean, maximum value and standard deviation (SD) are specified and computed for the algorithms based on the results of the computational time of various datasets. It was found that the OBJ, MFC and COG algorithms all performed significantly worse than the proposed EECS, especially when the algorithms need to schedule and execute a maximum number of tasks in CDC.

6.4.2. *Energy consumption*

Energy consumption of a task is defined as the amount of energy consumed by the resources of the computational server which is assigned for that task (defined in Eq. (15)). The energy consumption of a task depends on the energy consumption of the transmitting channel and the energy consumption of the resources during processing a task in a computing server. Here, we considered that most of the IoT application may execute locally or the Fog devices which minimize the total energy consumption. The proposed EES strategy finds a suitable computational server for each task and assigns the task to the selected server. However, the existing multi-objective scheduling strategies deploy the tasks to a computational server which may consume the maximum amount of resources due to single object optimization and consume maximum energy for running the assigned task. The comparative analysis between the EES algorithm and the existing state-of-art-algorithms in term of energy consumption over various numbers of tasks are shown in Fig. 4. The average energy consumption of EES strategy is better than the OJD 21%, the MFC algorithm by 24%, and the COG algorithm by 28%. Based on the experiments, the EES strategy outperforms than other scheduling algorithms for different synthetic datasets.

The statistical analysis of the proposed EES strategy and the existing multi-objective scheduling algorithms in a cloud environment is shown in Table 9. This may prove the efficiency of the EECS strategy over the existing ones in term of energy consumption. The metrics of minimum value, mean, maximum value and standard deviation (SD) are specified and computed for the algorithms based on the results of the energy consumption of various datasets. It was found that the OJD, MFC and COG algorithms all performed significantly worse than the proposed EES, especially when the algorithms need to schedule and execute a maximum number of tasks in CDC.

6.4.3. *CO_2 emission*

CO_2 emission of a task is defined as the amount of CO_2 emitted by the resources of executing device which is assigned for that task (defined in Eq. (17)). The CO_2 emission of a task depends on the CO_2 emitted of the transmitting channel and the CO_2 emitted of the resources during processing a task in a computing server. The proposed EES strategy finds a suitable computing server for each task and assigns the task to the selected server. However, the existing multi-objective scheduling

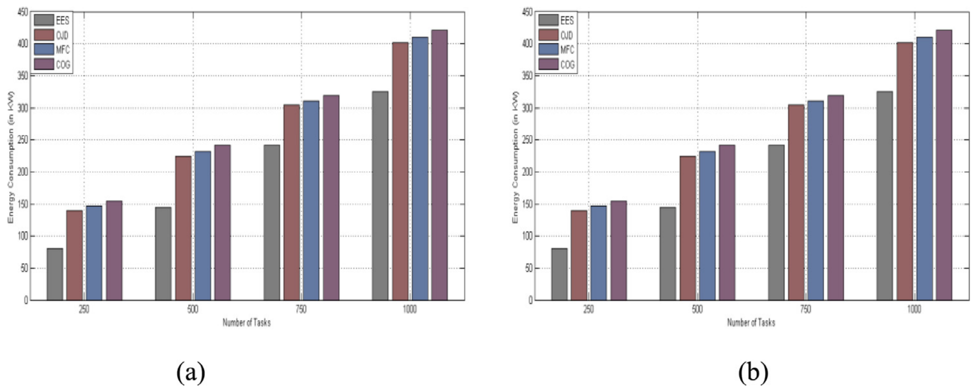

Fig. 4. Average energy consumption of different state-of-arts-algorithms: (a) DS-1 and (b) DS-2.

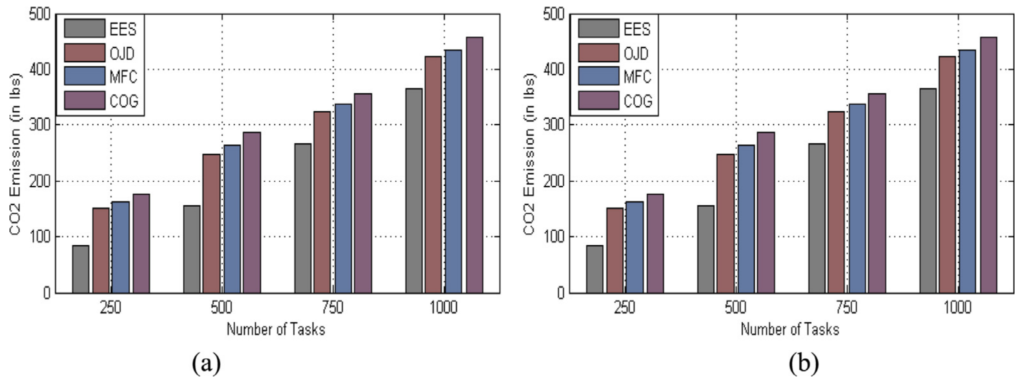

Fig. 5. Average CO_2 emission of different state-of-arts-algorithms: (a) DS-1 and (b) DS-2.

Table 9
Statistical analysis of energy consumption.

Datasets		Energy Consumption for IoT and Non-IoT based tasks			
		EECS	MOBFOA	PSO_COGENT	PSO
DS-1	**Minimum**	7.12	15.34	18.32	21.53
	Mean	25.32	37.65	39.51	42.56
	Maximum	42.87	57.23	65.02	71.98
	SD	3.145657	3.84320	3.94520	4.14234
DS-2	**Minimum**	8.37	17.45	20.11	22.76
	Mean	26.54	38.76	40.12	42.95
	Maximum	47.54	57.76	60.43	67.13
	SD	3.215431	4.33245	4.87516	5.13187

strategies deploy the tasks to a computational server which may consume the maximum amount of resources and emit maximum CO_2 for running the assigned task. The comparative analysis between the EES algorithm and the existing state-of-art-algorithms in term of CO_2 emission over various numbers of tasks are shown in Fig. 5. The average CO_2 emission of EECS strategy is better than the OJD 22%, the MFC algorithm by 27%, and the COG algorithm by 30%. Based on the experiments, the EES strategy outperforms than other scheduling algorithms for different synthetic datasets.

The statistical analysis of the proposed EECS strategy and the existing multi-objective scheduling algorithms in a cloud environment is shown in Table 10. This may prove the efficiency of the EES strategy over the existing ones in term of CO_2 emission. The metrics of minimum value, mean, maximum value and standard deviation (SD) are specified and computed for the algorithms based on the results of the CO_2 emission of various datasets. It was found that the OJD, MFC and COG algorithms all performed significantly worse than the proposed EES, especially when the algorithms need to schedule and execute a maximum number of tasks in CDC.

Table 10
Statistical analysis of CO_2 emission.

Datasets		Energy consumption for IoT and non-IoT based tasks			
		EECS	MOBFOA	PSO_COGENT	PSO
DS-1	**Minimum**	8.12	16.64	19.72	23.63
	Mean	25.32	38.65	40.53	43.58
	Maximum	47.87	59.23	68.02	73.98
	SD	3.245657	3.74320	3.94520	4.14234
DS-2	**Minimum**	9.47	16.55	21.01	23.75
	Mean	27.54	38.98	41.17	41.75
	Maximum	47.84	59.76	61.73	66.53
	SD	3.205431	3.71245	3.97516	4.13187

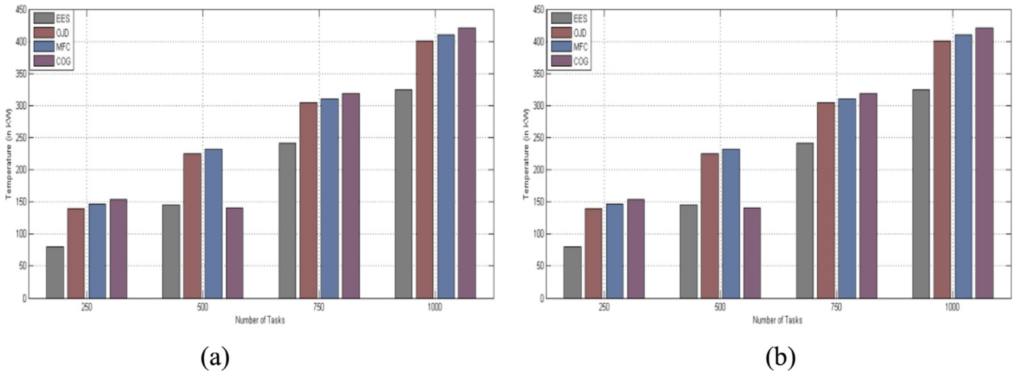

Fig. 6. Average CO_2 emission of different state-of-arts-algorithms: (a) DS-1 and (b) DS-2.

6.4.4. Temperature emission

Temperature emission of a task is defined as the amount of heat emitted by the resources of executing device which is assigned for that task (defined in Eq. (19)). The Temperature emission of a task depends on the amount of heat emitted of the transmitting channel and the heat emitted of the resources during processing a task in a computing server. The proposed EES strategy finds a suitable computing server for each task and assigns the task to the best-fit server. However, the existing multi-objective scheduling strategies deploy the tasks to a computing server which may consume the maximum amount of resources emit maximum heat for running the assigned task. The comparative analysis between the EES algorithm and the existing state-of-art-algorithms in term of Temperature emission over various numbers of tasks are shown in Fig. 6. The average Temperature emission of EES strategy is better than the OJD 21%, the MFC algorithm by 24%, and the COG algorithm by 27%. Based on the experiments, the EES strategy outperforms than other scheduling algorithms for different synthetic datasets.

The statistical analysis of the proposed EES strategy and the existing multi-objective scheduling algorithms in a cloud environment is shown in Table 11. This may prove the efficiency of the EES strategy over the existing ones in term of Temperature emission. The metrics of minimum value, mean, maximum value and standard deviation (SD) are specified and computed for the algorithms based on the results of the Temperature emission of various datasets. It was found that the OJD, MFC and COG algorithms all performed significantly worse than the proposed EECS, especially when the algorithms need to schedule and execute a maximum number of tasks in CDC.

Table 11
Statistical analysis of temperature emission.

Datasets		Energy consumption for IoT and non-IoT based tasks			
		EECS	MOBFOA	PSO_COGENT	
DS-1	**Minimum**	7.12	15.34	18.32	21.53
	Mean	25.32	37.65	39.51	42.56
	Maximum	42.87	57.23	65.02	71.98
	SD	3.145657	3.84320	3.94520	4.14234
DS-2	**Minimum**	8.37	17.45	20.11	22.76
	Mean	26.54	38.76	40.12	42.95
	Maximum	47.54	57.76	60.43	67.13
	SD	3.215431	4.33245	4.87516	5.13187

7. Conclusion

In this paper, we have proposed a meta-heuristic based offloading strategy in a Fog-Cloud environment. The proposed offloading strategy selects a suitable computing device for each IoT application using two QoS parameters such as computation time and energy consumption of the devices. The Firefly algorithm is used to solve the multi-objective optimization strategy and increases the accuracy of the problem. This strategy helps to assign the delay-intensive applications on the Fog devices and the resource-intensive applications are assigned to the cloud data center. This may minimize the average computation time of the IoT application and also minimizes the overall energy consumption for processing the IoT applications. The performance of the proposed algorithm is evaluated over two real-time datasets. The experimental results show the superior performance of the proposed offloading strategy over the existing state-of-arts-algorithms in terms of average computation time, average energy consumption, CO_2 emission and temperature emission of the computing devices. In our future, we will consider various SLA constraints in the Fog-Cloud environment for better accuracy and performance.

Conflict of interest

The author declares no conflict of interest in this paper.

References

[1] J. Stankovi, Research directions for the internet of things, IEEE Internet Things J. 1 (1) (2014) 3–9.
[2] A. Zanella, Internet of things for smart cities, IEEE Internet Things J. 1 (1) (2014) 22–32.
[3] C. Perera, P.P. Jayaraman, A. Zaslavsky, P. Christen, D. Georgakopoulos, Context-aware Dynamic Discovery and Configuration of Things in Smart Environment, Springer International Publishing, Cham, 2018, pp. 215–241.
[4] C. Perera, Y. Qin, J.C. Estrella, S. Reiff-Margnaiec, A.V. Vasilakos, Fog computing for sustainable smart cities: a survey, ACM Comput. Surv. 50 (3) (2017) 1–43.
[5] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the Internet of Things, in: Proceeding in ACM MCC Workshop on Mobile Cloud Computing, Helsinki, Finland, 2012.
[6] M. Chiang, T. Zhang, Fog and IoT: an overview of research opportunities, IEEE Internet Things J. 3 (6) (2016) 854–864.
[7] A. Brogi, S. Forti, QoS-aware deployment of IoT applications through the fog, IEEE Internet Things 4 (5) (2017) 1113–1116.
[8] International Series in Operations Research & Management Science K. Miettinen, F.S. Hillier (Ed.), Nonlinear Multiobjective Optimization, 12 Springer, Boston, MA, 1998. International Series in Operations Research & Management Science.
[9] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, 1st ed., John Wiley & Sons, Chichester, New York, 2001 Wiley-Interscience Series in Systems and Optimization.
[10] Q. Zhang, H. Li, MOEA/D: a multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (2007) 712–731.
[11] K. Liang, L. Zhao, X. Chu, H.-H. Chen, An integrated architecture for software-defined and virtualized radio access networks with fog computing, IEEE Netw. 31 (1) (2017) 80–87.
[12] C. Fricker, F. Guillemin, P. Robert, G. Thompson, Analysis of an offloading scheme for data centers in the framework of fog computing, ACM Trans. Model. Perform. Eval. Comput. Syst. 1 (4) (2016) 1–16.
[13] R. Craciunescu, A. Mihovska, M. Mihaylov, S. Kyriazakos, R. Prasad, S. Halunga, Implementation of fog computing for reliable e-health applications, in: Proceedings of the 2015 49th Asilomar Conference on Signals, Systems, and Computers, IEEE, 2015, pp. 459–463.
[14] R. Hasan, M. Hossain, R. Khan, Aura: an incentive-driven ad-hoc IoT cloud framework for proximal mobile computation offloading, Futur. Gener. Comput. Syst. 86 (2017) 821–835.
[15] Y. Wang, M. Sheng, X. Wang, L. Wang, J. Li, Mobile-edge computing: partial computation offloading using dynamic voltage scaling, IEEE Trans. Commun. 64 (10) (2016) 4268–4282.
[16] G. Orsini, D. Bade, W. Lamersdorf, Computing at the mobile edge: designing elastic android applications for computation offloading, in: Proceeding in 8th International Conference on IFIP Wireless and Mobile Networking Conference, WMNC, IEEE, 2015, pp. 112–119.
[17] K. Habak, M. Ammar, K.A. Harras, E. Zegura, Femto clouds: leveraging mobile devices to provide cloud service at the edge, in: Proceeding in IEEE 8th International Conference on Cloud Computing, CLOUD, IEEE, 2015, pp. 9–16.
[18] H. Shah-Mansouri, V.W.S. Wong, Hierarchical fog-cloud computing for IoT systems: a computation offloading game, IEEE Internet Things J. 5 (4) (2018) 1–12.
[19] J. He, J. Wei, K. Chen, Z. Tang, Y. Zhou, Y. Zhang, Multi-tier fog computing with large-scale IoT data analytics for smart cities, IEEE Internet Things J. 5 (2) (2018) 1–10.
[20] H. Tan, Z. Han, X.-Y. Li, F.C.M. Lau, Online job dispatching and scheduling in edge-clouds, in: Proceeding in IEEE Conference on Computer Communications, INFOCOM 2017, 2017, pp. 1–9.
[21] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison, ACM Comput. Surv. 35 (2003) 268–308.
[22] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[23] X.S. Yang, Firefly algorithms for multimodal optimization, in: O. Watanabe, T. Zeugmann (Eds.), Stochastic Algorithms: Foundations and Application, SAGA 2009, Lecture Notes in Computer Science, 5792, Springer-Verlag, Berlin, 2009, pp. 169–178.