An IoT-based two-factor divide and conquer task scheduler and deep resource allocator for cloud computing

Sripriya Arun*

Alpha Arts & Science College, Porur,
No.30, Tundalam Road, Chettiyar Agaram Road,
Behind Ramachandra Hospital,
Porur, Chennai, Tamil Nadu 600116, India
Email: priyaarun02@gmail.com
*Corresponding author

Sundara Rajan

Government Arts College for Men,
No.329, Annasalai, Nandanam, Chennai – 600035, India
Email: drmsrajan23@yahoo.com

Abstract: The epidemic developing rate of the networking technologies has resulted in an impressive sizeable scope of the associated computing framework. Internet-of-things (IoT) is considered a substitute for acquiring high performance by the improved potentialities in task scheduling, resource allocations and information exchanges. However, the current IoT is experiencing the gridlock of the task scheduling and resource allocation due to the higher level of dependency while scheduling and convoluted service contributing frameworks. With task scheduling and resource allocation considered with salient characteristics of cloud computing (CC) environment, this paper proposes a method called two-factor task scheduler and deep resource allocator (TFTS-DRA) based on IoT. In this method, each task is processed before its actual allocation to the cloud resources by cost and time-based divide and conquer task scheduling model. The resources are allocated using deep resource allocation model, which considers the auto encoder (AE) and fully connected neural network (FCNN) with energy consumption and transmission delay of cloud resources as constraints. Simulation results show that the proposed TFTS-DRA method performs in an extensive manner with higher throughput rate. The numerical results shows that proposed deep resource allocator algorithm in an IoT-CC environment, both the bandwidth utilisation and energy consumption can be improved.

Keywords: internet of things; IoT; cloud computing; two factors; task scheduler; deep resource allocator; auto encoder; fully connected neural network.

Reference to this paper should be made as follows: Arun, S. and Rajan, S. (2022) 'An IoT-based two-factor divide and conquer task scheduler and deep resource allocator for cloud computing', *Int. J. Internet Technology and Secured Transactions*, Vol. 12, No. 1, pp.61–79.

Biographical notes: Sripriya Arun holds a PhD from VISTA in Computer Science and Master in Computer Applications from Bharathidasan University. She has made unique contribution into understanding the broad sweep of technical developments in computer cloud security and digital forensics. She has authored and published several articles on cloud security at various forums. She has also integral part of many discussion forums and contributed to various workshop related to cryptography and cloud forensics computing. She is currently working as a Vice-Principal in Alpha Arts and Science College, Chennai.

Sundara Rajan guided more than ten doctoral scholars of state and deemed university in computer science with 24 years of experience in teaching. He started his career as software consultant in the USA and moved to teaching, presented papers in international journals, served as member for academic board in colleges and university. He is a member of the Staff Selection Commission Regional Office, Southern Region, Chennai.

1 Introduction

With the developing popularity of the application of cloud computing, the needs for CC have also increased in height. Research analysis concentrating on CC has started using it due to its better performance along with the lower energy consumption model. Many science applications, like the internet of things (IoT), sequencing gene operations and prediction for earthquake modelling are becoming in high demand vulnerable to high-performance computing and storage in a distributed fashion. Hence, CC can be seen as a perspective where a flexible heterogeneous resource pool via network, and users on demand rent varied resources according to the needs and requirements.

Two IoT-aware multi-resource task scheduling algorithm was proposed in Lin et al. (2019) to attain both load balancing and minimise the task response time for each IoT device. The two tasks scheduling algorithms were resource load balancing and time balancing based on heuristics. The main contribution lies in the way the correct virtual mappings are being performed between virtual machines. During the pre-processing, two phases were included, task ordering by means of priority and task re-ordering according to the task category. The urgency was addressed by means of prioritising and segregation of task between IoT and common was done by means of categorisation.

Besides, two strategies, namely, selecting strategy and mapping strategy were also included via greedy algorithm. With this, both the energy consumption and IoT average task response time were found to be reduced. However, dependent task scheduling was not focused and therefore compromising both the bandwidth utilisation and average response time. In this work to improve both the bandwidth utilisation and average response time, a dependent task scheduling model based on cost and time-based divide and conquer task scheduling is designed. Here, by considering both the computation cost and communication time while performing divide and conquer while task scheduling, optimal task scheduling is said to be achieved.

A deadline and cost-aware genetic algorithm (DCGA) for scheduling of workflows in IoT applications were designed in Ma et al. (2019). With the objective of reducing and cut shorting the cost involved, the algorithm concentrated on the trivial aspects of the cloud, like, acquisition of resources on-demand, dynamics in a heterogeneous manner, delay incurred during the acquisition and variations of performance involved in the virtual machines.

Besides, heuristic operations were also utilised to allocate different tasks to the concerned VMs, therefore contributing to execution time and execution cost. However, the throughput was not focused. To address this issue, in this work, a deep resource allocator is designed that uses both auto encoder and fully connected neural network to achieve better throughput.

Under the above circumstances, in this paper we propose an integrated task scheduling and resource allocation method that not only evaluates the cost and time-based task scheduling, but also considers the clock cycle of each cloud computing machine during resource allocation. Furthermore, we consider the auto encoder and fully connected neural network for IoT applications in a dynamic cloud environment. In essence, the main contributions of this paper are as follows:

- We formulate a joint task scheduling and resource allocation problem in the downlink of the cloud computing environment with cost and time constraints. Therefore, we provide a divide and conquer framework to decompose the complex IoT device scheduling into manageable and recursively breakup of tasks to optimise the task to be scheduled for IoT applications.
- We formulate a two-side deep resource allocator via auto encoder and fully connected neural network to initiate the scheduled task association followed by task scheduling between the cloud computing infrastructures and the IoT devices. Furthermore, the two-factor divide and conquer-based analytical framework significantly enhances the performance of the task response time.
- We perform extensive numerical analysis to evaluate the performance of the proposed approach. The results show that the two-factor task scheduler and deep resource allocator (TFTS-DRA) based on IoT for joint task scheduling and resource allocation achieves higher utility gain for the users. In addition, the average task response time outperforms the conventional resource allocation for IoT-based cloud computing.

The remainder of the paper is organised as follows. In Section 2, an extensive literature based on the current task scheduling and resource allocation for IoT-based applications in cloud environment is presented. In Section 3, the system model and the proposed method with neat block diagram and algorithm is presented. In Section 4, numerical analysis is presented to validate the performance and efficiency of the proposed method. Finally, in Section 5 the paper is concluded with concluding remarks.

2 Related works

In recent years, to ensure quality of service (QoS) to internet of things (IoT), the cloud computing environment has transferred toward the edge pattern. A joint user association and resource allocation framework was designed in Abedin et al. (2019) with the aid of analytic hierarchy process (AHP), therefore ensuring stability of user allocation and higher utility gain in terms of resource allocation. A specific fog computing network was proposed in Zhang et al. (2017) to achieve optimal and stable performance via Stackelberg game. However, two important factors, like, energy consumption and application requests completion time were not analysed. To focus on this aspect, energy and time efficient computation offloading and resource allocation (ETCORA) algorithm was presented in Sun et al. (2019).

Security aspects were included in Ben Daoud et al. (2019) by means of distributed access control mechanism. With this high focus was made on privacy and security aspects along with the minimisation of administration complexity. However, with massive number of applications, data processing has to be performed between IoT devices in an efficient manner. Therefore, QoS was focused in Reddy and Krishna (2020) by means of optimised fuzzy scheduling approach. With this, makespan of time was reduced along with the improvement in QoS.

As far as disaster management is concerned, resource allocation is said to be one of the most crucial one to be addressed. A resource allocation algorithm was presented in Choksi and Zaver (2019) using multi-objective mechanism dealing with both under and over demand resource utilisation. Besides, a priority-based scheduling technique was also applied for dense network. A distributed resource allocation for IoT using generalised Nash equilibrium (GNE) was proposed in Abuzainab et al. (2017) that not only decreased the total energy consumption but also satisfied QoS constraints in an effective manner. A resource trading mechanism utilising iterative double sided auction scheme was presented in Li et al. (2017).

There have been several research articles combining both the cloud and sensors. However, combining both sensors and applications at the application layer are found to be new in number. The objective of the work done in Bose et al. (2019) was to present a sensor-cloud architecture that created virtual sensors to facilitate both computing and resource capabilities. A collaborative between fog and cloud was performed in Alsaffar et al. (2016) to improve resource allocation in an efficient manner and ensuring optimised big data distribution. Virtual experiments were conducted in Al-Zoubi and Wainer (2019) besides real experiments separately to select best discovered resource servers.

A student project allocation game was designed in Gu et al. (2018) for the postulated joint resource allocation problem that in turn improved user satisfaction. However, the reliability was not addressed. To focus on this aspect, the power of machine learning was combined with optimisation mechanism in Liao et al. (2019) that contributed network throughput. QoS for smart home healthcare applications was proposed in Hassan et al. (2017) using agent-based modelling (ABM) for ensuring optimal resource allocation. A comprehensive survey to enhance the IoT network performance was provided in Yu et al. (2018). A summary of IoT research related to smart technology, data mining, cellular communication was investigated in Ud Din et al. (2019).

However, due to the mismatch in service quality observed in networking and environment providing complicated services, the quality of experience were compromised. To address this issue, two reinforcement learning-based algorithms were utilised in Gaia and Qiub (2018) to improve the same. A game strategy that included profit to the providers and enforcing penalty cost to offenders were proposed in Choi and Lim (2016) therefore, concentrating on the success rate of job completion to a greater extent.

A cooperative resource scheduling was carried out in Al-Turjman et al. (2018) for energy-constrained application with high reliability performance. The designed process improved the quality of service with higher throughput and minimal delay. Genetic algorithm (GA) and ant colony optimisation (ACO) termed GAACO algorithm was introduced in Basu et al. (2018) to choose the best combination of tasks at every stage. GAACO algorithm guaranteed the suitable convergence and optimality.

A scheduling algorithm was introduced in Preethi and Jayavel (2018) to assign the static tasks with respect to resource availability efficiently. The designed algorithm managed the situation efficiently through inserting the jobs under different weightage queues. A new technique was designed in The et al. (2019) to optimise the task scheduling issue for bag-of-tasks applications in cloud with minimal execution time and operating cost.

As discussed in the literature survey, a number of materials and methods have been presented for scheduling tasks and using the optimal resources. With the availability of constrained resources and large number of devices to be allocated for IoT-based applications, the optimal resource allocation is very much necessary. In this context, there necessitates a method that considers both tasks and performs the allocation of resources by considering several objectives and benchmarks. There are a number of issues like, independent and dependent task scheduling, minimising the task response time, and accomplishing all IoT tasks with minimum cost. In this work, these issues are handled, which differentiates our proposed method from other methods reported in the literature. In this paper, factor task scheduler and deep resource allocator (TFTS-DRA) based on IoT for cloud computing environment is proposed. The proposed method is evaluated based on the average computation cost and average communication time. Overall, using our proposed method, the performance of the system is improved in terms of the number of parameters, namely, bandwidth utilisation, task response time and throughput rate.

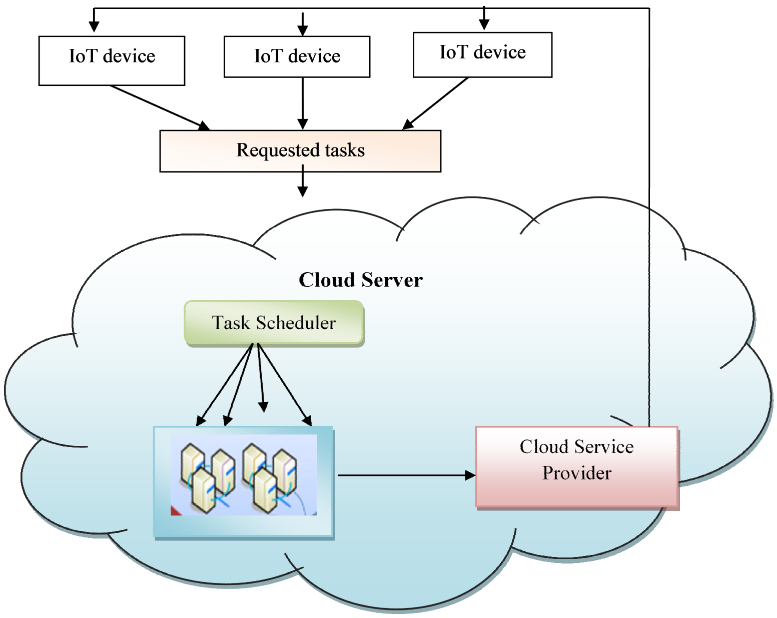
3 Two-factor task scheduler and deep resource allocator method

In this section, a method called, two factor task scheduler and deep resource allocator (TFTS-DRA) based on IoT for cloud computing environment with the objective of improving the throughput with minimum bandwidth and task response time is presented. The proposed method is split into two stages. They are task scheduling and resource allocation. In real time situation, different types and size of tasks arrive at the cloud computing stations. The proposed method obtains the requests from several IoT devices as input. To manage the tasks that enter into the cloud computing environment, a cost and time-based divide and conquer task scheduling model is first designed. Next, in the second stage, the proposed method also addresses the resource allocation by the cloud server via deep resource allocation model. The elaborate description of the TFTS-DRA method is given below.

3.1 System model

Let us consider a network consisting of a set of IoT devices '$D = D_1, D_2, \ldots, D_n$', such as smart phones, cameras for surveillance, vehicles, fire alarming devices and so on. These IoT devices or IoT users may discharge certain type of storage tasks to the cloud service providers (CSPs), that are represented as '$CSP = CSP_1, CSP_2, \ldots, CSP_n$'. These '$CSPs$' can meet several IoT devices or users with distinct computing requirements both in terms of data size '$Data_{size}$' and service delay 'S_{delay}' respectively. The overall network topology is illustrated in Figure 1.

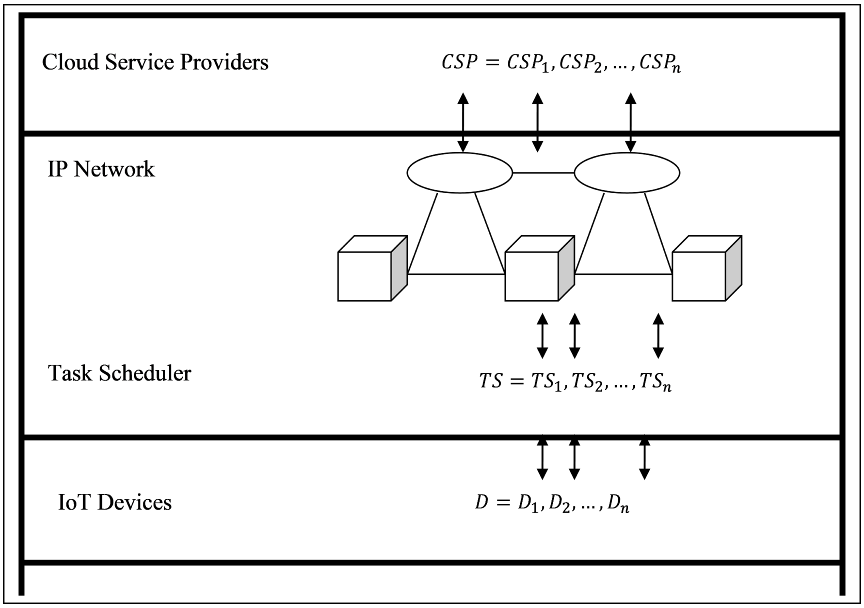
Figure 1 Overall network topology (see online version for colours)



To be specific, certain sensitive IoT devices like, fire alarming devices are frequently more delay sensitive. On the other hand, the IoT devices like smart phones are frequently more adaptable concerning the service delay essentials. Let us consider the fire alarm system as a real-time case scenario. Fire department or fire brigade (i.e., cloud server) is a public organisation that provides the predominantly emergency fire fighting and rescue services for particular geographic area (i.e., municipality, county, or fire protection district). Alarms are considered as the first line of defence in event of fire occurrence and act as early warning sign giving the chance to escape from any fire-related danger. Each building/establishment in geographic area acts as an IoT network. A fire department includes one or more fire stations (i.e., cloud service provider) within boundaries and staffed by career firefighters, volunteer firefighters, or combination. When an alarm warning sign is received in fire department, task scheduler schedules the nearby fire stations for the preventing from fire accidents. By this way, we tell that IoT devices are not delay sensitive, the computing are sent to the cloud and for those IoT devices with precise delay essentials, the requirements, the Task Scheduler '$TS = TS_1, TS_2, \ldots, TS_n$' will assign one of the '*CSPs*' to discharge the computation task. However, the '*CSPs*' that are adjacent to the IoT devices classically result in smaller delay. But, the topological factor is not the only element that influences the entire service delay.

The service delay comprises of three time periods, which are transmission time, CPU processing time and the reception time. The transmission time and reception time are defined as the time utilised for sending data to task schedulers '*TS*' and the time utilised for obtaining the processed results, depending on the data size. The CPU processing time refers to the CPU rate of each '*TS*'. Thus for any cloud service provider '*CSP*', while selecting the task scheduler from the set '$TS^j = \{ts_1^j, ts_2^j, \ldots, ts_i^j\}$' for each IoT device or user will cooperatively assign its bandwidth '$BW^j = bw_1^j, bw_2^j, \ldots, bw_m^j$' and CPU clock speed '$CPU^j = cpu_1^j, cpu_2^j, \ldots, cpu_n^j$' respectively. Figure 2 shows the system model.

Figure 2 Proposed system model


Cloud Service Providers
$CSP = CSP_1, CSP_2, \ldots, CSP_n$
IP Network
Task Scheduler
$TS = TS_1, TS_2, \ldots, TS_n$
IoT Devices
$D = D_1, D_2, \ldots, D_n$

From the IoT devices' or users' outlook, who have delay sensitive data to be processed, are sent to the cloud service provider to content for finer resources (both bandwidth and CPU clock speed). In addition, IoT devices' or users' will take the data sizes into consideration. This is due to the reason that more data to be processed necessitates extensive transmission time and also extensive CPU processing time.

Let us further assume, the cooperative bandwidth and CPU clock cycle to be treated as the scaling between the IoT user or device sets '*D*' and the (cooperative bandwidth and CPU clock cycle) resource duality sets '$RD^j = \{(bw_m^j, cpu_n^j) | bw_m^j \in BW^j, cpu_n^j \in CPU^j\}$' possessed by each cloud service provider. Then, the scaling correlation '*SC*' is mathematically derived as given below:

$$SC_{mn}^{ij} = 1, \quad \text{if } d_i \text{ is discharged to } TS \tag{1}$$

$$SC_{mn}^{ij} = 0, \quad \text{if } d_i \text{ is not discharged to } TS \tag{2}$$

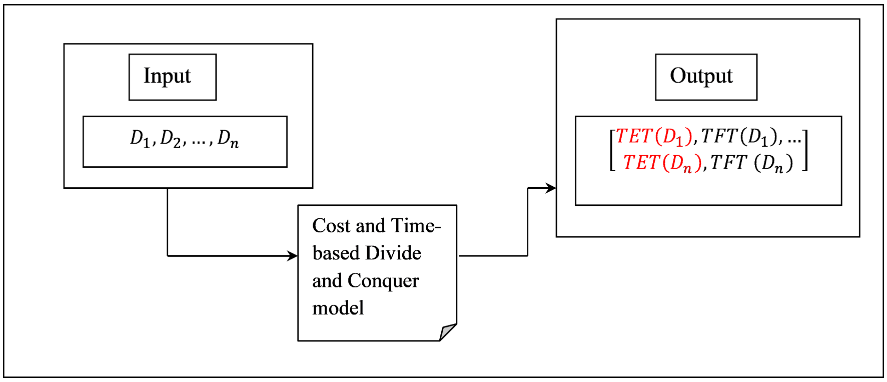
With the purpose of optimising the integrated resource allocation, both the benefit of IoT devices' '*D*' and the cloud service providers '*CSPs*' are considered in this work that are analysed in the forthcoming sections.

3.2 *Cost and time-based divide and conquer task scheduling model*

In this section, a cost and time-based divide and conquer task scheduling model is designed with the objective of arranging the incoming requests or tasks in an optimal manner by considering the dependent tasks also during scheduling. Let '$D = D_1, D_2, \ldots, D_n$' represent a set of IoT devices on the IoT layer connected to the task scheduler '$TS = TS_1, TS_2, \ldots, TS_n$'. Besides, a duplex communication model has been utilised in our work to ensure task flow, data transmission and data reception in a parallel fashion.

The input of the heterogeneous IoT task scheduling model includes a schedule of finite directed graph with task scheduler and IoT devices. The output is a schedule constituting the allocation of incoming requests or tasks in a certain manner so that the resources available in hand are said to be utilised optimally. On the other hand, the task scheduling comprises of each task in each application being allocated with Task establishment time '$TET(D_i)$' and task finishing time '$TFT(D_i)$' respectively. Figure 3 shows the block diagram of cost and time-based divide and conquer task scheduling model.

Figure 3 Block diagram of cost and time-based divide and conquer task scheduling model (see online version for colours)


Input
$D_1, D_2, \ldots, D_n$
Output
$\begin{bmatrix} TET(D_1), TFT(D_1), \ldots \\ TET(D_n), TFT(D_n) \end{bmatrix}$
Cost and Time-based Divide and Conquer model

As illustrated in Figure 3, the implementation of cost and time-based divide and conquer task scheduling model is given below. The tasks to be scheduled are recursively break down into two or more tasks and this process is repeated until the tasks are said to be scheduled in a direct pattern. With the arrival of 'n' number of tasks, the tasks are broke down based on the average computation cost and average communication time and accordingly, the process of scheduling is performed.

To start with the average computation cost 'ACC' and average communication time 'ACT' required for each IoT device 'D_i' that has placed its request as input to the task scheduler 'TS' is formulated and is mathematically expressed as given below:

$$ACC(t_i) = D_i = \sum_{i=1}^{n} \frac{CPU_i}{R_i / i} \tag{3}$$

From equation (3), the average computation cost 'ACC' for each task 't_i' to corresponding device 'D_i' is measured by means of the CPU clock cycle 'CPU_i' and processing rate 'R_i' respectively.

$$ACT(t_i) \to D_{ij} = \sum_{i=1}^{n} \frac{DM_{ij}}{BW_i / i} \tag{4}$$

Besides, the average communication time '*ACT*' for each task 't_i' is measured via data migration between device 'D_i' and 'D_j' and the bandwidth 'BW_i' respectively. In order to schedule a task for each IoT device on a processor by the task scheduler, tasks have to be accommodated on idle time slots (ITS). For that purpose, in this work, the task establishment time '*TET*' and task finishing time '*TFT*' are defined. Here '$TET(t_i, P_n)$' and '$TFT(t_i, P_n)$' represents the task establishment time and task finishing time of task 't_i' for device 'D_i' on processor 'P_n' respectively. They are mathematically formulated as given below:

$$TET(t_i, P_n) = MAX\{Time(P_n), [ACC(t_i) + ACT(t_i)]\} \tag{5}$$

$$TFT(t_i, P_n) = TET(t_i, P_n) + Time(t_i, P_n) + ArrivalTime(Prec(t_i)) \tag{6}$$

From equation (5), the task establishment time '*TET*' is obtained based on the execution time on processor 'P_n' potential of scheduling task 't_i', average computation cost '*ACC*', average communication time '*ACT*' and the arrival time of the parent task of '$Prec(t_i)$' respectively. In a similar manner the task finishing time '*TFT*', is measured based on the task establishment time '*TET*' and execution time of task 't_i' on processor 'P_n' respectively. The pseudo code representation of two-factor divide-and-conquer dependent task scheduler is given below.

Algorithm 1 Two-factor divide-and-conquer dependent task scheduler

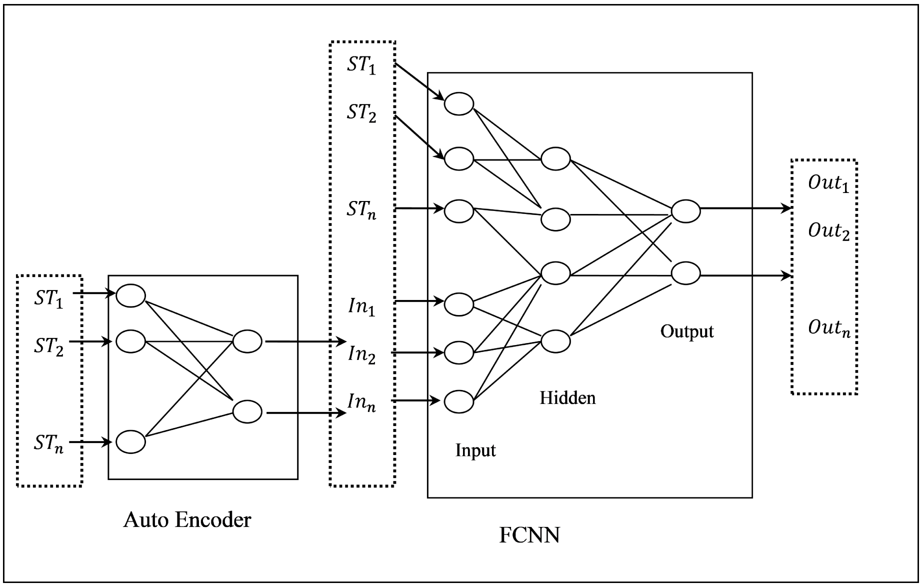
Input: IoT devices '$D = D_1, D_2, \ldots, D_n$', Cloud Service Provider '$CSP = CSP_1, CSP_2, \ldots, CSP_n$', Task Scheduler '$TS = TS_1, TS_2, \ldots, TS_n$'

Output: Optimal Scheduled Task '$ST = ST_1, ST_2, \ldots, ST_n$'

```
1: Begin
2:     For each IoT devices 'D' and Task Scheduler 'TS'
3:         Evaluate the average computation cost using (3)
4:         Evaluate the average communication time using (4)
5:         Measure task establishment time using (5)
6:         Measure task finishing time using (6)
7:         Return (optimal scheduled tasks)
8:     End for
9: End
```

As given in the above two-factor divide-and-conquer dependent task scheduler algorithm, as the name implies two different factors is considered. They are cost and time. First, the incoming tasks are recursively break down. The solution to the problem is arrived at based on the two different factors. The average computation cost for each IoT devices for scheduling particular task is obtained. Next, the average communication time incurred in scheduling the task is measured. Finally, the task establishment and task finishing time are evaluated. Accordingly, the tasks are said to be scheduled in an optimal manner, therefore contributing to both bandwidth utilisation and task response time.

Figure 4 Block diagram of deep resource allocator



3.3 Deep resource allocation

To improve the performance and efficiency in terms of throughput, in this work, a deep resource allocation model is designed. This deep resource allocation model considers both the auto encoder and fully connected neural network to the scheduled task to improve the rate of throughput. Let us consider an encode function '$EF(.)$' and decode function '$DF(.)$' for the AE and is mathematically formulated as given below:

$$EF(.) = \log\left[1 + EXP\left(WM_E(.) + B_E(.)\right)\right] \tag{7}$$

$$DF(.) = WM_D(.) + B_D(.) \tag{8}$$

From equations (7) and (8), encode and decode functions is arrived at based on the weight matrix of encode function '$WM_E(.)$', weight matrix of decode function '$WM_D(.)$', bias of encode function '$B_E(.)$' and bias of decode function '$B_D(.)$' respectively. In addition, there exists 'n' layers in FCNN, where '$WM^{(n,n-1)}$' represents the weight matrix between '$(n-1)$th' layer and 'nth' layer with 'B^n' representing the bias for neurons in the 'nth' layer. Besides, the cloud energy consumption and transmission delay are also used as input. Energy consumed is measured based on the CPU clock cycle of each cloud computing machine. Then, energy consumption based on the CPU clock cycle of each cloud computing machine is measured as given below:

$$E_j^{cloud} = \alpha_j N_j * \left[E(D_i)\right] \tag{9}$$

As given in equation (9), the cloud energy consumption of the 'jth' cloud server is obtained by multiplying the on/off state of cloud server 'j', 'α_j', number of turned on machines at cloud server 'N_j' and each device energy consumption value '$E(D_j)$' respectively. Let 'd_{ij}' denote the delay of IoT device transmission path from task scheduler '*i*' to the cloud server '*j*' and 'λ_{ij}' represent the traffic arrival rate from task scheduler '*i*' to the cloud server '*j*', then, the transmission delay is measured as given below:

$$TD_{ij} = d_{ij} * \lambda_{ij} \tag{10}$$

Figure 4 shows the block diagram of deep resource allocator.

The inputs of the deep resource allocation model for AE consists of the known parameters, i.e., scheduled task '*ST*' and the input of the FCNN consists of the scheduled task and the ratio of the scheduled task and number of requests made by the IoT devices, that is rewritten as given below:

$$In(AE) = \sum_{i=1}^{n} ST_i \tag{11}$$

$$In(FCNN) = ST_i \cup In_i, \qquad \text{where } In_i = \sum_{i=1}^{n} \frac{(ST_i)}{Req_i} \tag{12}$$

The output of AE is the union of the scheduled task and the ratio of scheduled task to the requests made. Finally, the output of FCNN is written as given below with the consideration of the energy consumption 'E_j^{cloud}' and transmission delay 'TD_{ij}'.

$$Out = \sum_{i=1}^{n} (In)\left(E_j^{cloud}\right) TD_{ij} \tag{13}$$

The pseudo code representation of deep resource allocator is given below:

Algorithm 2 Deep resource allocator

Input: IoT devices '$D = D_1, D_2, \ldots, D_n$', Cloud Service Provider '$CSP = CSP_1, CSP_2, \ldots, CSP_n$', Task Scheduler '$TS = TS_1, TS_2, \ldots, TS_n$'

Output: Improved throughput

```
1:  Begin
2:      Initialise scheduled tasks 'ST'
3:          For each IoT devices 'D' and Cloud Service Provider 'CSP'
4:              Evaluate encode and decode function for AE using (7) and (8)
5:              Measure activation function based on energy consumption and transmission
                delay using (9) and (10)
6:              Obtain input for AE and FCNN using (11) and (12)
7:              Measure output for FCNN using (13)
8:              Return (optimal resource allocation)
9:          End for
10: End
```

As given in the above algorithm, for each IoT devices and cloud service provider as input, the scheduled tasks arrived is first initialised. Next, with the objective of improving the throughput, a deep resource allocator algorithm, considering both AE and FCNN is utilised that in addition to the conventional model also utilises the energy consumption and transmission delay into consideration for managing the resources to fulfil the requests generated by the IoT devices in a timely manner.

4 Implementation and analysis

In this section, the numerical experiments results are presented to examine the efficiency of storage and resource allocation as well as comparing our method performance with other methods in terms of bandwidth utilisation, task response time and throughput to allocate the resources to user smart devices via task scheduler and cloud server. The comparison method uses IoT-aware multi-resource task scheduling (Lin et al., 2019) and deadline and cost-aware genetic algorithm (DCGA) (Ma et al., 2019) towards resource scheduling via task scheduler using active personal cloud measurement obtained from Personal Cloud Datasets, NEC Personal Cloud Trace (Gracia-Tinedo et al., 2013).

4.1 Experiment settings

The characteristics of our target system are illustrated in Table 1. In our PC, one Intel Core^{TM} i7 965 and 8 GB RAM is used. The proposed method was simulated on CloudSim (https://code.google.com/p/cloudsim/downloads/list), a framework specifically utilised for modelling and simulation of infrastructures and services in Java.

Table 1 Characteristic of the target system

Parameter	*Value*
Network	LAN
Topology	Connected
Number of smart devices	10
Bandwidth	10 – 512 Mbps

4.2 Comparative analysis of bandwidth utilisation

Bandwidth utilisation refers to the data rate (i.e., for each IoT device) that is supported by the network connection (i.e., cloud server) or the interfaces that connect to the network. Bandwidth utilisation denotes both the volume and time. In other words, it refers to the amount or frequency of data that are said to be transmitted between two ends in a stipulated time period and expressed in terms of bits per second (b/s) or kilo bits per second (Kb/s). This is mathematically formulated as given below:

$$BU = \frac{Avg\_D}{Avail\_NBW} * D \tag{14}$$

From equation (14), the bandwidth utilisation 'BU' is measured based on the average utilisation required by specific application (i.e., IoT devices) 'Avg_D', available network bandwidth '$Avail\_NBW$' and the number of IoT device's request 'D' respectively. Table 2 shows the comparative analytical representation results of bandwidth utilisation for three different methods, TFTS-DRA, IoT-aware multi-resource task scheduling (Lin et al., 2019) and DCGA (Ma et al., 2019).

Table 2 Comparative results of TFTS-DRA, IoT-aware multi-resource task scheduling (Lin et al., 2019) and DCGA (Ma et al., 2019)

Number of IoT devices	*Bandwidth utilisation (Mb/s)*		
	TFTS-DRA	*IoT-aware multi-resource task scheduling*	*DCGA*
25	500	400	300
50	600	450	350
75	700	530	420
100	900	620	500
125	1,200	750	630
150	1,300	900	840
175	1,400	1,200	990
200	1,800	1,400	1,200
225	1,850	1,550	1,450
250	2,000	1,700	1,650

Figure 5 Bandwidth utilisation comparisons (see online version for colours)

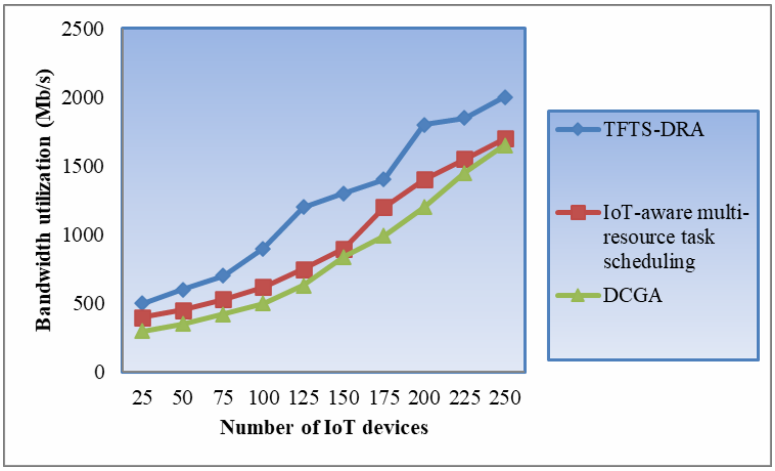


Figure 5 shows the comparative analysis results of bandwidth utilisation for three different methods, TFTS-DRA, IoT-aware multi-resource task scheduling (Lin et al., 2019) and DCGA (Ma et al., 2019). From the figure it is inferred that increasing the number of IoT devices cause an increase in the bandwidth utilisation. It is evident from the simulation experiments conducted with the requests of the numbers of IoT devices. With the assumed available network bandwidth being ‘250 Kb/s’, average utilisation by specific IoT device for TFTS-DRA, IoT-aware multi-resource task scheduling (Lin et al., 2019) and DCGA (Ma et al., 2019) were found to be ‘125 Kb/s’, ‘100 Kb/s’ and ‘75 Kb/s’ respectively.

From this it is inferred that the bandwidth utilisation was found to be better when applied with TFTS-DRA. This is due to the application of two-factor divide-and-conquer dependent task scheduler algorithm. By applying this algorithm, two different factors, like, average computation cost and average communication time were considered during the scheduling of tasks. With this, the average utilisation required by the IoT devices were said to be satisfied with the aid of TFTS-DRA and therefore improving the bandwidth utilisation by 32% compared to Lin et al. (2019) and 5% compared to Ma et al. (2019).

4.3 Comparative analysis of task response time

The task response time refers to the time consumed in responding to the IoT devices' tasks. Higher the number of tasks being requested by the IoT device, greater is the task response time. This is mathematically evaluated as given below:

$$TR_{time} = \sum_{i=1}^{n} D_i * Time[TR = TFT - TET] \tag{15}$$

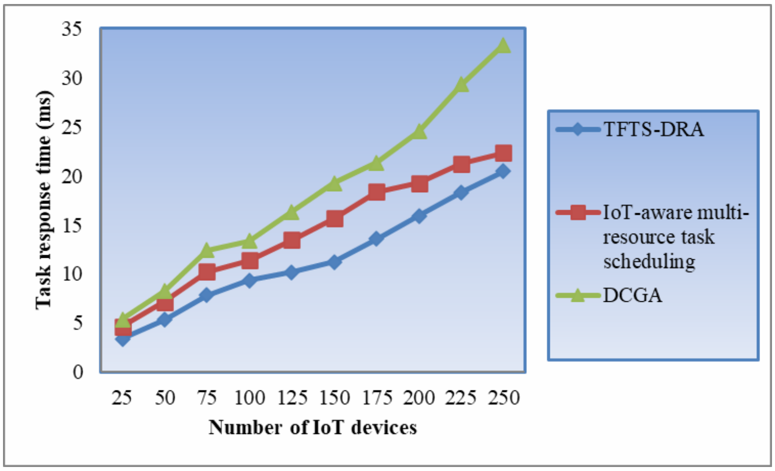
From equation (15), the task response time is measured based on the IoT devices' requests 'D_i' being placed in the cloud server and the time consumed in responding the task '*TIME*[*TR*]', which is the difference between the task finishing time '*TFT*' and task establishment time '*TET*' respectively. Table 3 shows the comparative analytical representation results of average task response time for three different methods, TFTS-DRA, IoT-aware multi-resource task scheduling (Lin et al., 2019) and DCGA (Ma et al., 2019).

Table 3 Comparative results of TFTS-DRA, IoT-aware multi-resource task scheduling (Lin et al., 2019) and DCGA (Ma et al., 2019)

Number of IoT devices	*Task response time (ms)*		
	TFTS-DRA	*IoT-aware multi-resource task scheduling*	*DCGA*
25	3.375	4.625	5.375
50	5.325	7.135	8.235
75	7.855	10.255	12.445
100	9.325	11.355	13.355
125	10.155	13.455	16.352
150	11.235	15.675	19.255
175	13.555	18.355	21.355
200	15.895	19.235	24.555
225	18.325	21.225	29.355
250	20.455	22.355	33.325

Figure 6 given above shows the response time comparison in the y axis and number of IoT devices' requests placed by IoT devices in the cloud server in the x axis. The number of IoT devices is varied in the range of 25 to 250 with an average 10 simulation run at different time intervals. Increasing the number of IoT devices requests in the cloud server obviously results in the large number of tasks to be scheduled by the cloud server in cloud computing environment. This is evident from the simulation with '15' IoT devices request placed in the cloud server, the tasks response time for scheduling single IoT device using TFTS-DRA was found to be ‘0.135 ms’, ‘0.185 ms’ using (Lin et al., 2019) and ‘0.215 ms’ using (Ma et al., 2019) respectively.

Figure 6 Task response time comparisons (see online version for colours)



From these results it is inferred that the average task response time is comparatively lesser using TFTS-DRA when compared to Lin et al. (2019) and Ma et al. (2019). This is because of the application of cost and time-based divide and conquer task scheduling model. By applying this model, the IoT devices’ tasks to be scheduled are partitioned into smaller elements and the elements are scheduled optimally via two cost and time factors. Besides, the IoT devices’ request to be accommodated on idle time slots (ITS), the task establishment time ‘*TET*’ and task finishing time ‘*TFT*’ are defined. With this, the average task response time with TFTS-DRA is said to be reduced by 21% compared to Lin et al. (2019) and 37% compared to Ma et al. (2019).

4.4 Comparative analysis of throughput

Finally, the throughput refers to the percentage ratio of IoT devices’ request of task addressed with the required resources ‘D_i*addressed*’ to the IoT devices’ request placed in the cloud server ‘D_i*request place*’.

$$T = \frac{D_i addressed}{D_i request\ placed} * 100 \tag{16}$$

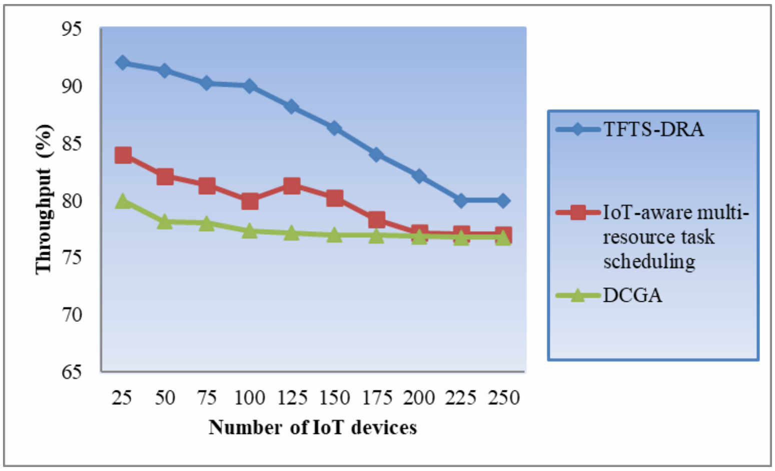
From equation (16), with higher number of devices’ tasks being allocated with the required resources, more efficient the method is said to be and is measured in terms of percentage (%). Table 4 shows the comparative analytical representation results of throughput rate for three different methods, TFTS-DRA, IoT-aware multi-resource task scheduling (Lin et al., 2019) and DCGA (Ma et al., 2019).

Figure 7 shows the comparative result analysis of throughput rate for three different methods. As illustrated in the above figure, increasing the IoT devices results in larger number of IoT devices requests for allocating the scheduling tasks and therefore larger number of requests being handled to allocation in an optimal manner, the throughput obviously will get reduced. However, from the figure it is inferred that the throughput rate using the proposed method is less compared to the Lin et al. (2019) and Ma et al. (2019). This is because of the application of deep resource allocator algorithm. In this algorithm, both the auto encoder and fully connected neural network are considered. Besides, the activation function is generated based on the cloud energy consumption and transmission delay. With this, resource allocation for the scheduled task is made only based on energy being consumed and the delay incurred during transmission. By considering all these factors, the throughput rate using TFTS-DRA is improved by 8% when compared to Lin et al. (2019) and 11% when compared to Ma et al. (2019).

Table 4 Comparative results of TFTS-DRA, IoT-aware multi-resource task scheduling (Lin et al., 2019) and DCGA (Ma et al., 2019)

Number of IoT devices	*Throughput (%)*		
	TFTS-DRA	*IoT-aware multi-resource task scheduling*	*DCGA*
25	92	84	80
50	91.35	82.15	78.15
75	90.25	81.35	78
100	90	80	77.35
125	88.15	81.35	77.15
150	86.35	80.25	77
175	84	78.35	76.95
200	82.15	77.15	76.85
225	80	77.06	76.75
250	80	77.02	76.77

Figure 7 Throughput rate comparisons (see online version for colours)



5 Conclusions

In this paper, we have focused on ensuring the quality of service in terms of bandwidth utilisation, task response time and throughput for end users by efficiently allocating the available resources to IoT-based applications in cloud environment. Therefore, we have proposed a two-factor-based resource allocator model that is scalable and well applicable to the cloud computing environment. Unlike traditional resource allocation models for IoT, we have efficiently mapped the resources to the IoT applications by considering computation cost and communication time for the IoT applications. In addition, we have investigated the throughput rate by means of combining the auto encoder and fully connected neural network on the outcomes of the scheduled tasks through extensive analysis. The simulation results show that, the proposed method is able to address task response time and bandwidth utilisation and we have observed significant performance gains compared to the other traditional IoT-based resource allocation methods for cloud computing environment.

References

Abedin, S.F., Alam, M.G.R., Kazmi, S.M.A., Tran, N.H., Niyato, D. and Hong, C.S. (2019) 'Resource allocation for ultra-reliable and enhanced mobile broadband IoT applications in fog network', *IEEE Transactions on Communications*, January, Vol. 67, No. 1, pp.489–502.

Abuzainab, N., Saad, W., Hong, C.S. and Poor, H.V. (2017) 'Cognitive hierarchy theory for distributed resource allocation in the internet of things', *IEEE Transactions on Wireless Communications*, December, Vol. 16, No. 12, pp.7687–7702.

Alsaffar, A.A., Pham, H.P., Hong, C-S., Huh, E-N. and Aazam, M. (2016) 'An Architecture of IoT service delegation and resource allocation based on collaboration between fog and cloud computing', *Mobile Information Systems*, August, Vol. 2016, pp.1–16, Hindawi Publishing Corporation.

Al-Turjman, F., Hasan, M.Z. and Al-Rizzo, H. (2018) 'Task scheduling in cloud-based survivability applications using swarm optimization in IoT', *Transaction on Emerging Telecommunication Technologies*, November, Vol. 30, No. 8, pp.1–20, Wiley Online Library.

Al-Zoubi, K. and Wainer, G. (2019) 'Fog and cloud collaboration to perform virtual simulation experiments', *Simulation Modelling Practice and Theory*, October, Vol. 101, pp.1–20, Elsevier.

Basu, S., Karuppiah, M., Selvakumar, K., Li, K-C., Islam, S.K.H., Hassan, M.M. and Bhuiyan, M.Z.A. (2018) 'An intelligent/cognitive model of task scheduling for IoT applications in cloud computing environment', *Future Generation Computer Systems*, November, Vol. 88, pp.254–261, Elsevier.

Ben Daoud, W., Obaidat, M.S., Meddeb-Makhlouf, A., Zarai, F. and Hsiao, K-F. (2019) 'TACRM: trust access control and resource management mechanism in fog computing', *Human-centric Computing and Information Sciences*, July, Vol. 9, No. 28, pp.1–18, Springer Open.

Bose, S., Sarkar, D. and Mukherjee, N. (2019) 'A framework for heterogeneous resource allocation in sensor-cloud environment', *Wireless Personal Communications*, April, Vol. 108, pp.19–36, Springer.

Choi, Y. and Lim, Y. (2016) 'Optimization approach for resource allocation on cloud computing for IoT', *International Journal of Distributed Sensor Networks*, January, Vol. 2016, pp.1–6, Hindawi.

Choksi, M. and Zaver, M.A. (2019) 'Multiobjective based resource allocation and scheduling for postdisaster management using IoT', *Wireless Communications and Mobile Computing*, March, Wiley, Vol. 2019, pp.1–16.

CloudSim, *A Framework for Modeling and Simulation of Cloud Computing Infrastructure and Services* [online] https://code.google.com/p/cloudsim/downloads/list.

Gaia, K. and Qiub, M. (2018) 'Optimal resource allocation using reinforcement learning for IoT content-centric services', *Applied Soft Computing*, May, Vol. 70, pp.12–21, Elsevier.

Gracia-Tinedo, R., Sánchez Artigas, M., Moreno-Martínez, A., Cotes, C. and López, P.G. (2013) 'Actively measuring personal cloud storage', *6th IEEE International Conference on Cloud Computing (Cloud'13)*, 27 June–2 July, pp.301–308.

Gu, Y., Chang, Z., Pan, M., Song, L. and Han, Z. (2018) 'Joint Radio and computational resource allocation in IoT fog computing', *IEEE Transactions on Vehicular Technology*, August, Vol. 67, No. 8, pp.7475–7484.

Hassan, M.M., Albakri, H., Al-Dossari, H. and Mohamed, A. (2017) 'Resource provisioning for cloud-assisted body area network in a smart home environment', *IEEE Access*, July, Vol. 5, pp.13213–13224.

Li, Z., Yang, Z. and Xie, S. (2017) 'Computing resource trading for edge-cloud-assisted internet of things', *IEEE Transactions on Wireless Communications*, December, Vol. 16, No. 12, pp.3661–3669.

Liao, H., Zhou, Z., Zhao, X., Zhang, L., Mumtaz, S., Jolfaei, A., Ahmed, S.H. and Bashir, A.K. (2019) 'Learning-based context-aware resource allocation for edge computing-empowered industrial IoT', *IEEE Internet of Things Journal*, December, Vol. 7, No. 5, pp.4260–4277.

Lin, W., Peng, G., Bian, X., Xu, S., Chang, V. and Li, Y. (2019) 'Scheduling algorithms for heterogeneous cloud environment: main resource load balancing algorithm and time balancing algorithm', *Journal of Grid Computing*, November, Vol. 17, pp.699–726.

Ma, X., Gao, H., Xu, H. and Bian, M. (2019) 'An IoT-based task scheduling optimization scheme considering the deadline and cost aware scientific workflow for cloud computing', *EURASIP Journal on Wireless Communications and Networking*, July, Vol. 249, pp.1–19, Springer.

Preethi, M. and Jayavel, K. (2018) 'IoT based visualization of weightage based static task scheduling algorithm in datacenter', *International Journal of Engineering and Technology*, Vol. 7, No. 2, pp.439–443, Science Publishing Corporation.

Reddy, D.A. and Krishna, P.V. (2020) 'Feedback-based fuzzy resource management in IoT using fog computing', *Evolutionary Intelligence*, March, pp1–13, Springer.

Sun, H., Yu, H., Fan, G. and Chen, L. (2019) 'Energy and time efficient task offloading and resource allocation on the generic IoT-fog-cloud architecture', *Peer-to-Peer Networking and Applications*, July, Vol. 13, pp.548–563, Springer.

The, A.T., Thanh, B.H.T., Bao, S.D. and Minh, N.B. (2019) 'Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment', *Applied Sciences*, Vol. 9, No. 9, pp.1–20.

Ud Din, I., Guizan, M., Hassan, S., Kim, B-S., Khan, M.K., Atiquzzaman, M., Ahmed, S.H. (2019) 'The internet of things: a review of enabled technologies and future challenges', *IEEE Access*, January, Vol. 7, pp.7606–7640.

Yu, W., Liangi, F., He, X., Hatcher, W.G., Lu, C., Lin, J. and Yang, X. (2018) 'A survey on the edge computing for the internet of things', *IEEE Access*, March, Vol. 6, pp.6900–6919.

Zhang, H., Xiao, Y., Bu, S., Niyato, D., Yu, R. and Han, Z. (2017) 'Computing resource allocation in three-tier IoT fog networks: a joint optimization approach combining Stackelberg game and matching', *IEEE Internet of Things Journal*, October, Vol. 4, No. 5, pp.1204–1215.

Nomenclatures

List of symbols	*Description*
$D = D_1, D_2, \ldots, D_n$	Set of IoT devices
$CSP = CSP_1, CSP_2, \ldots, CSP_n$	Cloud service providers
$Data_{size}$	Data size
S_{delay}	Service delay
$TS = TS_1, TS_2, \ldots, TS_n$	Task scheduler
$TS^j = \{ts_1^j, ts_2^j, \ldots, ts_i^j\}$	Task scheduler selection
$BW^j = bw_1^j, bw_2^j, \ldots, bw_m^j$	Bandwidth
$CPU^j = cpu_1^j, cpu_2^j, \ldots, cpu_n^j$	CPU clock speed
$RD^j = \{(bw_n^j, cpu_n^j) \mid bw_m^j \in BW^j, cpu_n^j \in CPU^j\}$	Resource duality sets
$TET(D_i)$	Task establishment time
$TFT(D_i)$	Task finishing time
ACC	Average computation cost
ACT	Average computation time
n	Number of tasks
t_i	Task
P_n	Processor
$Prec(t_i)$	Arrival time of parent task
$EF(.)$	Encode function
$DF(.)$	Decode function
$WM_E(.)$	Weight matrix of encode function
WM_D	Weight matrix of decode function
$B_E(.)$	Bias of encode function
$B_D(.)$	Bias of decode function
N_j	Number of turned on machines
B^n	Bias for neurons in 'n^{th}' layer
$E(D_j)$	Each device energy consumption value
d_{ij}	Delay of IoT device transmission path
λ_{ij}	Traffic arrival rate
E_j^{cloud}	Energy consumption
TD_{ij}	Transmission delay
BU	Bandwidth utilisation
Avg_D	Average bandwidth utilisation by IoT devices
$Avail\_NBW$	Available network bandwidth
$Time[TR]$	Time consumed in responding tasks
D_i *addressed*	IoT device request of task addressed with required resources
D_i *request placed*	IoT devices request placed