Computational Methods for Differential Equations
http://cmde.tabrizu.ac.ir
Vol. 7, No. 4 (Special Issue), 2019, pp. 521-534

A priority-based service placement policy for Fog-Cloud computing systems

Sadoon Azizi[∗]
Department of Computer Engineering and IT,
University of Kurdistan, Sanandaj, Iran.
E-mail: `s.azizi@uok.ac.ir`

Fariba Khosroabadi
Department of Computer Engineering and IT,
University of Qom, Qom, Iran.
E-mail: `f.khosroabadi@stu.qom.ac.ir`

Mohammad Shojafar
Department of Computer Science,
Ryerson University, Toronto, Canada.
E-mail: `mohammad.shojafar@ryerson.ca`

Abstract Recent advances in the context of Internet of Things (IoT) have led to the emergence of many useful IoT applications with different Quality of Service (QoS) requirements. The fog-cloud computing systems offer a promising environment to provision resources for IoT application services. However, providing an efficient solution to service placement problem in such systems is a critical challenge. To address this challenge, in this paper, we propose a QoS-aware service placement policy for fog-cloud computing systems that places the most delay-sensitive application services as closer to the clients as possible. We validate our proposed algorithm in the iFogSim simulator. Results demonstrate that our algorithm achieves significant improvement in terms of service latency and execution cost compared to simulators built-in policies.

Keywords. Internet of Things (IoT), Fog-Cloud computing system, Quality-of-Service (QoS), Service placement, iFogSim.

2010 Mathematics Subject Classification. 68-06, 68M11, 68M14.

1. Introduction

Nowadays, the Internet of Things (IoT) is playing a significant role in humans daily life. IoT enables many smart applications in a variety of domains, including smart city, healthcare, smart grid, video surveillance, etc. Due to the popularity and growth of these applications, the number of IoT devices (e.g., sensors, cameras, actuators, and smart meters) are dramatically increasing. So it is expected that a tremendous amount of data is generated by these devices, which may require real-time processing. However, most IoT devices are limited in terms of computational power, storage, and

∗ Corresponding author.

battery life. Hence, more powerful devices are required for processing and storage services.

Cloud computing infrastructure can be used to host IoT applications. The integration of cloud computing and IoT is known as Cloud of Things model (CoT) [1]. In the CoT model, the cloud acts as a middleware between IoT applications/end-users and IoT devices by which the development of smart services, such as smart transportation systems [2] and pervasive healthcare [3], can be simplified. Nevertheless, the geographically distributed nature of the IoT applications does not match with the centralized nature of the cloud data centers. As IoT devices usually are very far away from the cloud data centers, high latency in service delivery and high bandwidth consumption is inevitable.

There are many IoT applications such as connected cars, augmented reality, remote healthcare monitoring systems, and video streaming, which require low response time. To meet such a requirement, it is necessary to place some computational and storage resources closer to IoT devices (i.e., sensors and actuators/users). To this end, the fog computing paradigm is introduced as a complement to cloud computing to fulfill the requirements of IoT applications. This computing model is known as a fog-cloud computing system. By having this system, the IoT applications with different requirements can be supported. More specifically, the Quality of Service (QoS) for latency-sensitive applications is improved.

Fog-cloud computing system is still in its infancy. Although the system comes with many benefits to IoT applications, there exist numerous challenges and difficulties which need more research and attention. Service placement problem is one of the key research issues which has attracted significant attention from the research community due to its impressive impact on the overall system performance and cost.

Motivated by these considerations, in this paper, we propose a QoS-aware algorithm to service placement on a fog-cloud computing system. The proposed algorithm takes into account the deadline requirement of each IoT application so that the most delay-sensitive applications are placed on the devices as closely as possible to the service consumer. Furthermore, to reduce the network bandwidth and the cost of execution in the cloud, the number of assigned application services to the fog environment is maximized. Our algorithm is implemented in iFogSim simulator [4] and compared with the simulator's built-in policies, edge-ward and cloud-only. Results demonstrate that our algorithm significantly improves the QoS for delay-sensitive applications and reduces the cost of execution.

The rest of this paper is organized as follows. In Section 2, related works are reviewed. The system model and problem formulation are described in Section 3. The proposed service placement algorithm is presented in Section 4. Section 5 is devoted to performance evaluation and simulation results. Finally, Section 6 concludes the paper.

2. Related Work

The service placement in fog-cloud computing systems is a new research area that has recently attracted considerable attention [3, 5–13]. In this section, some of the most relevant to our work are discussed.

Skarlat et al. [6] provided a conceptual framework for service provision in fog computing. Their proposed framework is defined based on the concept of fog colony which has been considered in many related works [8,11,14]. A fog colony is a microdata center that is composed of an arbitrary number of fog cells. A fog cell is a software module that runs on IoT devices and provides virtualized resources for other IoT devices connected to it.

The authors in [7] have modeled the service placement problem in fog as Integer Linear Programming (ILP) optimization. The goal of their optimization is to maximize the utilization of the fog environment while applications QoS requirements are considered. Simulation results indicate that their model reduces processing cost and does not violate application deadlines. But ILP optimization is an NP-hard problem which is almost impossible to solve it on a large scale.

In [8], authors have modeled the service placement problem in the fog as an optimization problem, where the goal function is to maximize resource efficiency in the fog while satisfying each application's deadline. To solve this problem, they have implemented a genetic algorithm. The results of their study show that in their proposed approach, the deadlines are not violated. However, GA runs a large number of services in the cloud, which increases the cost of execution.

The work presented in [3] suggests an energy-efficient algorithm for application tasks placement. The main thesis of the algorithm is to put each task on the fog device, which has the least increase in energy consumption. In addition, their policy tries to use fog devices in a balanced manner. They have used the remote patient monitoring scenario as a case study to evaluate the performance of their proposed algorithm. Compared to the iFogSim default policies, edge-ward and cloud-only, their proposed policy reduces overall system power consumption. The main disadvantage of this method is that priority is not considered for applications.

Yousefpour et al. [9] formulated application services placement on the fog nodes in the form of Dynamic fog Service Provisioning (DFSP) problem. The authors have presented two heuristics to solve this problem. The first of them is the Min-Viol heuristic, which its goal is to minimize SLA violations while the second is the Min-Cost heuristic, which aims to minimize cost. The main thesis is based on a centralized approach that suffers from a single point of failure and decision making delay.

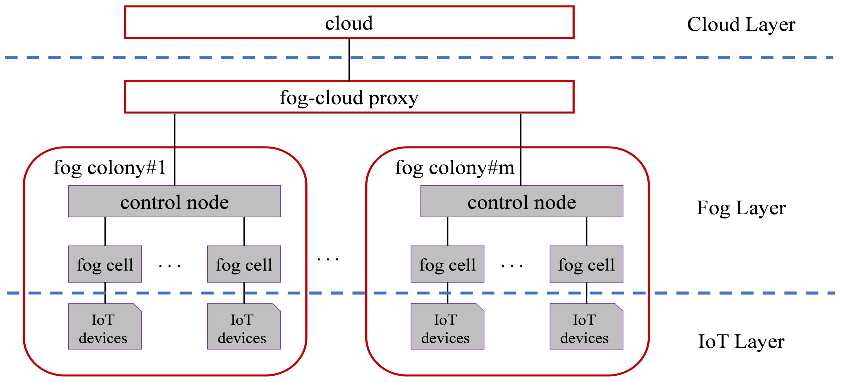
Recently, in [10], authors proposed a decentralized service placement approach for the fog computing landscape based on context-aware information. Their solution uses context information such as the location of IoT devices, network conditions, service type, and expected QoS for each application to provide resources efficiently for IoT applications. They used IBM CPLEX solver to solve their optimization problem. Their experimental results confirm that the proposed approach is useful in maximizing the efficiency of fog landscapes and reducing delay. According to this fact that their optimization model is of type Integer Linear Programming, when the number of services and fog nodes increases, the computation time of the problem-solving increases exponentially.

3. The system model and problem statement

In this section, we present our proposed architecture and form our problem statement.

3.1. **Assumed Architecture.** Our assumed architecture is based on the fog computing framework proposed by the authors in [6]. A view of this architecture has been depicted in Figure 1, where three layers can be defined: IoT layer, fog layer, and cloud layer. In this architecture, IoT devices are located on the lowest layer, which are distributed geographically in different locations. In the fog layer, fog computing is placed as the middle layer between IoT devices and cloud. In this layer, the fog devices are partitioned into some subsets (or clusters), that each of them is called a colony. Each colony contains a set of fog cells and a control node, that details of which are given in the next sub-section. In addition, the connection between fog computing and cloud computing is done by a proxy. In the cloud layer, the highest layer, the cloud servers are placed that provide more powerful computing than the fog devices.

Figure 1. Assumed fog computing architecture.



Each IoT application contains collections of modules that they are also known as *services* or *tasks*. These modules make the processing elements that are modeled based on Distributed Data Flow (DDF) [15]. Throughout this paper, the terms of the module, service, and task are used interchangeably. We also consider the dependencies between the services of each application that can be modeled by a directed graph in which the nodes stand for the application services and the edges represent the flow of data between services. Additionally, the Sense-Process-Actuate Model is used for IoT applications. Based on this model, first, data is collected by the sensors; next, it has been sent to the computing nodes in the higher layer (fog and/or cloud) to process and finally, the resultant commands are forwarded to the actuators. Such a model is completely considered in iFogSim [4].

3.2. **Problem Statement.** Consider a fog colony with a control node and m fog cells. We use the notation F for the control node and the set $R^F = \{f^1, f^2, \ldots, f^m\}$ for the fog cells that are inside the colony. Similar to [8], for each fog colony, we consider a neighbor fog colony which its control node has the lowest delay with F, and use the notation N to identify its control node. Also, the C notation represents cloud servers. Therefore, the total set of computational devices in the system model can be defined by $D = \{F, R^F, N, C\}$.

Each of the devices in the set D has different resources, such as CPU and RAM. C_F^{cpu} and C_F^{ram} are used to indicate the capacity of the CPU and RAM of the control node F, respectively. Such notations can be defined for other computing devices within the set D. We show the utilization of CPU and RAM of the control node F, by U_F^{cpu} and U_F^{ram}, respectively. Similar notations can be defined for other computing devices within the set D, too. The latency of the communication link between a particular fog cell f^j and the control node F is identified by d^j. Analogously, d^N and d^C indicate the delay between the control node F with the neighbor control node and the cloud, respectively.

Consider $A = \{A_1, A_2, \ldots, A_n\}$ representing IoT applications that are submitted by the IoT devices for processing to the control node of a fog colony. Each $A_i \in A$ contains a set of services $A_i = \{S_{i,1}, S_{i,2}, \ldots, S_{i,k}\}$, where $S_{i,j}$ represents the j-th service of the application i. Each service $S_{i,j}$ has different resource requirements which in this work, we focus on its computational resources, i.e., CPU and RAM. For each service $S_{i,j}$, its CPU and RAM requirements are defined by $S_{i,j}^{cpu}$ and $S_{i,j}^{ram}$, respectively. Based on the assumed model, i.e. Sense-Process-Actuate Model, each service is belonging to one of types sensing, processing or actuating. Finally, application A_i has a specific deadline D_{A_i} that is determine by the users of the application.

Regarding the above described model, the service placement problem in the fog-cloud computing system can be presented as follows:

Consider a set of applications $A = \{A_1, A_2, \ldots, A_n\}$ in which each application A_i has a deadline D_{A_i} and consists of the number of dependent services. On the other hand, there is a set of computational devices in the system including $D = \{F, R^F, N, C\}$.The process of mapping services into computational devices is known as the service placement problem.

The response time of application A_i is obtained by the summation of the deployment time W_{A_i}, and the makespan M_{A_i}, that can be calculated by the following equation [8]:

$$R_{A_i} = W_{A_i} + M_{A_i} \tag{3.1}$$

The deployment time of application A_i is determined by the two factors: waiting time in the queue and the required time it takes to put all of its services on the computational devices. When IoT applications are submitted to the control node F for execution, they enter the application queue until the time of problem-solving by F will come. For example, if application A_i arrives at time t_x while the time of problem-solving is t_y, in this case, the application should wait for the duration $t_y - t_x$ in the queue. This time is known as the waiting time for the application A_i. Additionally, it takes a while to run the service placement algorithm and provides the required resources for each of the services of application A_i. This time also should be added to the application deployment time.

The time it takes the application A_i services are deployed on the computational resources until the results are returned to the user, is known as the makespan of the application A_i. Based on the assumed model, i.e., the dependency between services, the makespan of application A_i can be calculated by the sum of the time required to run all of its services and the delay of the communicational link between the two services that are interdependent and placed on different devices.

4. Problem Formulation

There are several decision variables that we use to describe the service placement policy. In general, we use the variable $x_{i,j}^{dev}$ to indicate that the service $S_{i,j}$ on which dev computing device is located, where $dev \in D$. For example, if $x_{i,j}^{f^k} = 1$, then the service $S_{i,j}$ is located on the fog cell $f^k, 1 \leq k \leq n$.This can also be defined for the variables $x_{i,j}^F, x_{i,j}^N$ and $x_{i,j}^C$. In mathematical terms:

$$x_{i,j}^{f^k} = \begin{cases} 1, & \text{if } s_{i,j} \text{ is placed on the fog cell } f^k \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

$$x_{i,j}^{F} = \begin{cases} 1, & \text{if } s_{i,j} \text{ is placed on the fog control node} \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

$$x_{i,j}^{N} = \begin{cases} 1, & \text{if } s_{i,j} \text{ is placed on the neighbor colony} \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

$$x_{i,j}^{C} = \begin{cases} 1, & \text{if } s_{i,j} \text{ is placed on the cloud} \\ 0, & \text{otherwise} \end{cases} \tag{4.4}$$

4.1. **Optimization Formula.** To provide quick response time to users and reduce costs, it is imperative that the resources provided by the fog landscape can be utilized well. To this end, our goal is to solve the problem of module placement in a fog-cloud computing system in such a way that the number of assigned services to fog nodes be maximized. So the problem can be formulated as follows:

$$maximize \quad \sum_{i=1}^{n} \sum_{j=1}^{|A_i|} \left(\sum_{k=1}^{m} x_{i,j}^{f^k} + x_{i,j}^{F} + x_{i,j}^{N} \right) \tag{4.5}$$

Subject to the following constraints:

$$\sum\nolimits_{k=1}^{m} x_{i,j}^{f^k} + x_{i,j}^{F} + x_{i,j}^{N} + x_{i,j}^{C} = 1, \quad \forall S_{i,j} \tag{4.6}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{|A_i|} x_{i,j}^{D} S_{i,j}^{R} \leq C_{D}^{R}, \quad \forall k \in \{1, ..., m\}, D = \{R^F, F, N, C\}, R = \{cpu, ram\} \tag{4.7}$$

$$x_{i,j}^{f^k}, x_{i,j}^{F}, x_{i,j}^{N}, x_{i,j}^{C} \in \{0,1\} \tag{4.8}$$

$$R_{A_i} \leq D_{A_i} \quad \forall A_i \in A \tag{4.9}$$

Constraint (4.6) indicates that each service $S_{i,j}$ can be located only on a specific computing resource. Constraint (4.7) indicates that the total CPU and RAM of all services hosted on a computing device should not exceed the amount of CPU and RAM capacity of that device. The domain of decision variables is defined by constraint (4.8). Finally, constraint (4.9) ensures that the deadline for each application must be met. With considering the optimization formula (4.5) and its constraints, it is clear that the proposed model for the service placement problem is an Integer Linear Programming (ILP) problem. As a result, proposing an accurate solution to this problem in large scale is almost impossible. In the next section, we propose a simple, but the efficient heuristic algorithm for solving this problem that can find a promising answer in the polynomial time.

5. Proposed Algorithm

To deal with the issue of service placement in the fog-cloud computing environment, in this work, we propose an efficient heuristic algorithm called $Most\ Delay-sensitive\ Application\ First$ $(MDAF)$. It is worth noting that our proposed algorithm runs on the controller component of each fog control node. The details of this algorithm are given below.

The main idea behind MDAF is that the delay-sensitive application services are placed on resources that are closer to the client as much as possible. Given that the fog resources are closer to clients than the cloud resources, the first computational devices in the fog are used. We use the deadline of each application to determine its delay sensitivity.

Based on the assumed architecture, each IoT device is located inside a colony and is directly connected to a specific fog cell. When an IoT device (client) initiates an application, usually its sensing service by default is placed on the corresponding fog cell. Additionally, the actuating service of each application also has a specific destination which is selected from one of the fog cells. All processing services are submitted to the fog control node of the corresponding colony. The MDAF algorithm is run there to place the processing services of each application on appropriate computational devices.

In this work, we assume that processing services can run on one of the three computational devices F, N, and C. The pseudocode of the proposed algorithm is presented in Algorithm 1.

In Algorithm 1, the proposed algorithm first sorts applications in ascending order based on their deadlines (line 1). Computational devices are sorted according to their proximity to users in the corresponding colony (line 2). In other words, at first, the resources of the fog control node are utilized, after which the resources of the nearest neighbor colony and finally, the cloud resources are being used. This ensures that application services that have a short deadline will be placed on devices that have less response time for them. So, for each application A_i, its services are first put on the fog control node as far as possible. If this node does not have enough resources to allocate one of the services of this application, its services will be sent to the next computational device (lines 3-11). Furthermore, the algorithm tries to place services as far as possible on the fog computing resources to not only providing QoS but also reducing the cost of execution in the cloud.

Algorithm 1 Service Placement Algorithm

```
1:  Sort application list $A$ in ascending order of deadline;
2:  Sort computational devices in the list $D$ in order of closeness to the client;
3:  for each application $A_i$ in list $A$ do
4:      $curdev = F$;
5:      for each processing service in $A_i$ do
    say service $S_{i,j}$
6:          while service $S_{i,j}$ has not been placed do
7:              if $S_{i,j}^{cpu} \leq C_{curdev}^{cpu}$ && $S_{i,j}^{ram} \leq C_{curdev}^{ram}$ then
8:                  place $S_{i,j}$ on $curdev$;
9:                  $C_{curdev}^{cpu} = C_{curdev}^{cpu} - S_{i,j}^{cpu}$;
10:                 $C_{curdev}^{ram} = C_{curdev}^{ram} - S_{i,j}^{ram}$;
11:                 break ;
12:             else
13:                 $curdev$ = next device in the list $D$;
```

Table 1. Application deadlines

Application	D_{A_k} (ms)
A_1	3000
A_2	2500
A_3	3000
A_4	4500
A_5	2700

6. Performance Evaluation

We have implemented our proposed algorithm on the iFogSim simulator [4] and compared its performance with edge-ward and cloud-only strategies. The edge-ward strategy emphasizes on the implementation of services near the edge of the network. Devices on the edge of the network may not have sufficient computational resources to serve all the modules of an application. In such situations, higher-level devices are checked in terms of computational resources to find the appropriate device for services. In cloud-only strategy, all application services run on cloud data centers. In such applications, the sense-process-actuate loop works in this way that sensors send the received data to the cloud to run and after running services in the cloud, the results are returned to the user via actuators.

6.1. Simulation Settings.

To evaluate the proposed algorithm, we consider five applications with different deadline requirements (see Table 1). In this work, the deployment time of applications is equal to zero. Each application consists of four services, where the required resources for each service and the execution time of each service has been listed in Table 2. ($U[a, b]$ indicates a uniform random number between a and b).

We have considered a fog colony which includes five fog cells connected to the fog control node. The latency of the communication link between the fog control node and the cloud, the

Table 2. The resource demand of each service

Service	$S^{cpu}_{i,j}$ (MIPS)	$S^{cpu}_{i,j}$ (MB)	Execution Time (ms)
Sense	50	30	$U[100, 1000]$
Process1	200	10	$U[100, 1000]$
Process2	200	20	$U[100, 1000]$
Actuate	50	20	$U[100, 1000]$

Table 3. Characteristics of computational devices

Device Type	CPU(MIPS)	RAM(MB)
cloud	20000	20000
fog control node	1000	1024
fog cells	250	256

Table 4. Performance comparison

Scenario	R_{A_i}**(ms)**					$D_{A_i} - R_{A_i}$**(ms)**					**Placement (%)**				**Cost**
	A_1	A_2	A_3	A_4	A_5	A_1	A_2	A_3	A_4	A_5	f_j	F	N	C	**($)**
MDAF	2147	2126	2247	4191	2430	853	374	753	309	270	50	25	15	10	8.8
edge-ward	2047	2126	4147	4091	4430	953	374	-1147	409	-1730	50	25	0	25	14
cloud-only	4047	4126	4147	4091	4430	-1047	-1626	-1147	409	-1730	50	0	0	50	23

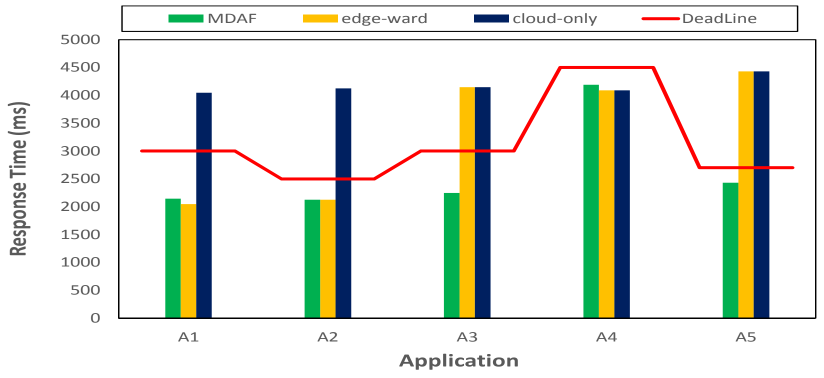
control node and the nearest neighbor node, the control node, and fog cells are respectively 1000, 50, and 10 ms. The characteristics of computational devices are shown in Table 3.

6.2. **Evaluation Parameters.** One of the parameters for evaluating a service placement plan is to check whether this plan complies with the applications deadline or not. To do this, you must calculate the response time of the applications. The difference between response time and application deadline $(D_{A_i} - R_{A_i})$ determines how the service placement plan performs in response to an application deadline, and whether it violates the application deadline. Another evaluation parameter is the utilization of devices (fog or cloud) resulting in the ratio of the number of services placed on the device to the total number of services. Whatever an algorithm can increase the utilization of a device, or in other words, whatever the policies of an algorithm go to place more services on a single device, the cost, and energy consumption of the devices are reduced. The philosophy of using the fog environment is to use the maximum capacity of the fog devices, and as far as possible, services will be sent less to cloud resources.

6.3. **Results.** To compare the performance of our algorithm, MDAF, with edge-ward and cloud-only strategies, the response time of each application, the amount of applications deadline violation, resource utilization and the cost of execution in the cloud is collected that a summary of them is shown in Table 4.

6.3.1. *Violation of applications deadline and services delay.* In Table 4, the edge-ward policy violates application A_3 deadline at 1,147 ms and application A_5 at 1,730 ms. In the cloud-only policy, the deadlines for applications of A_1 , A_2 , A_3 and A_5 are violated in 1,047, 1,626, 1,147, and 1,730 ms, respectively. But in MDAF policy, no application is violated. Because our algorithm has a special interest in the applications deadline and it applies the module placement policy from the minimum deadline to the maximum deadline for applications. The response time of applications and their deadlines was obtained in three policies is shown in Figure 2.

FIGURE 2. Response times of applications.



In Figure 2, the cloud-only policy violates the deadlines for applications of A_1, A_2, A_3 and A_5. The edge-ward policy does not comply deadlines of the A_3 and A_5 applications, but in the MDAF scenario, all application deadlines have complied. Because in the MDAF scenario, applications are sorted in ascending order based on their deadlines. For this reason, applications which have less deadline, are served faster than those with higher deadlines. But this does not comply in edge-ward and cloud-only, because of this, the response time of the services violates specified deadline.

We conduct another experiment to show the effect of the Application parameters on response times. In this experiment, we changed the deadline parameter of the Application A_4, and we observed the response time of all the Applications, the results of which are shown in Figure 3. Application A_4 was selected for this experiment since it has the highest response time. In Figure 3, when the deadline is less than 2500 milliseconds, the A_4 application services are assigned only to the fog control node because in this case, the A_4 deadline is lower than other applications. Above 2500 ms and less than 3000 ms, A_4 processing services are placed in the control node and neighbor control node. Above 3000 ms, A_4 processing services are placed in cloud data centers.

6.3.2. *Utilization of the fog landscape.* In all scenarios, the sensing and actuating services are placed on fog cells within the fog colony. However, in the MDAF, 5, 3 and 2 services are placed on the control node, the closest neighbor colony, and cloud, respectively. In the edge-ward, 5 services are placed on the control node and 5 services are placed on the cloud. In the cloud-only scenario, 10 services are placed on the cloud. Figure 4 shows the utilization of fog landscape in different scenarios.

FIGURE 3. Impact of application A_4 deadline on response times.

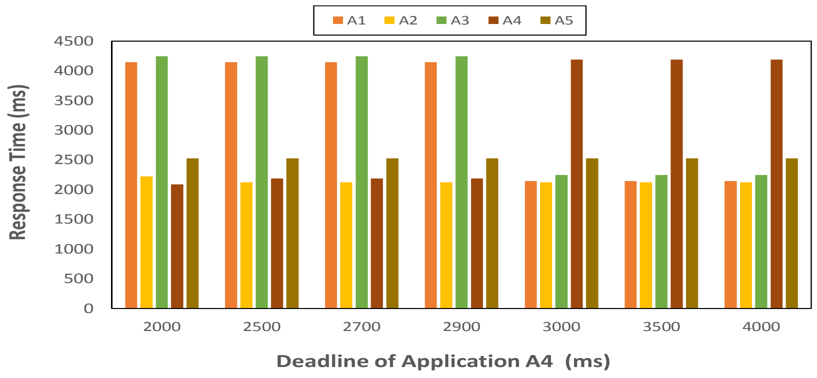
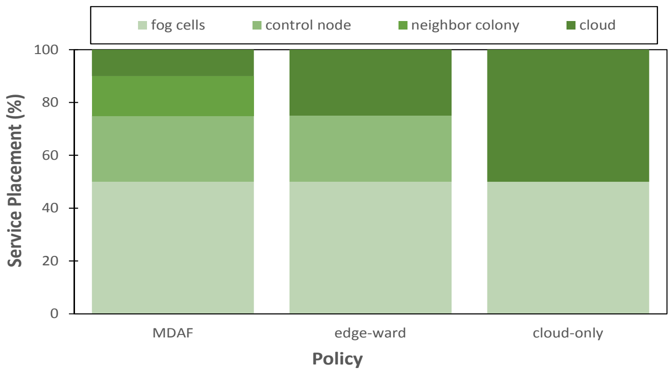


FIGURE 4. Utilization of resources.



6.3.3. *Cost of execution in cloud.* The cost of execution in the edge-ward and cloud-only is 14$ and 23$, respectively. The execution cost in the MDAF is just 8.8$, which constitutes 62% of edge-ward and 38% of cloud-only strategies. The results are shown in Figure 5. As we can see from Figure 5, our proposed algorithm significantly reduces the execution cost.

6.3.4. *Energy consumption.* The amount of energy consumed by mobile devices, fog nodes, and cloud data centers is compared using the three MDAF, edge-ward, and cloud-only scenarios in Figure 6. As shown in Figure 6, the MDAF algorithm reduces the energy consumed by fog devices by 1.34% compared to the edge-ward algorithm. The total energy consumed by various devices considering the proposed algorithm, edge-ward and cloud-only policies is presented in Figure 7. As we can observe, the proposed algorithm is more energy-efficient than the other two policies. Indeed, it saves approximately 0.038% of the energy compared to cloud-only and 0.66% of the energy compared to the edge-ward.

FIGURE 5. Cost of Execution in Cloud.

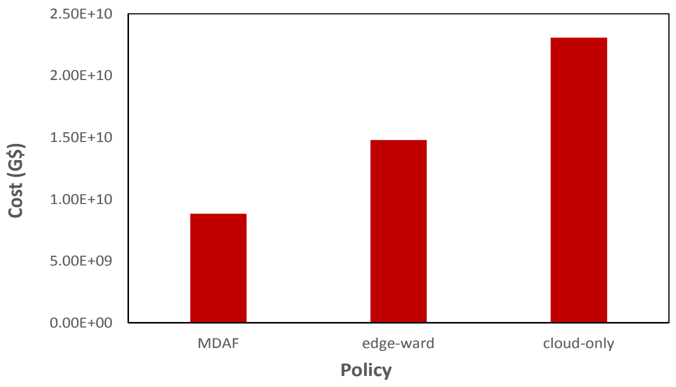
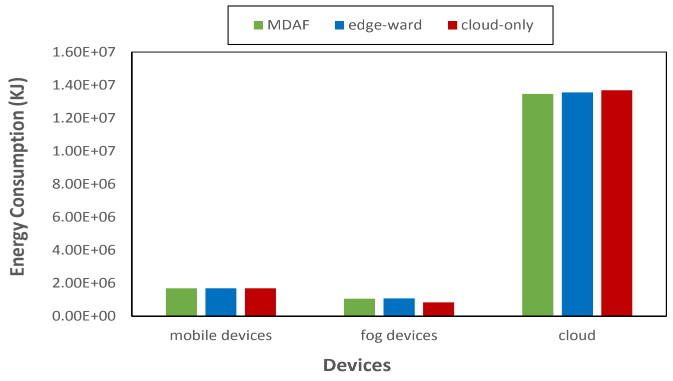


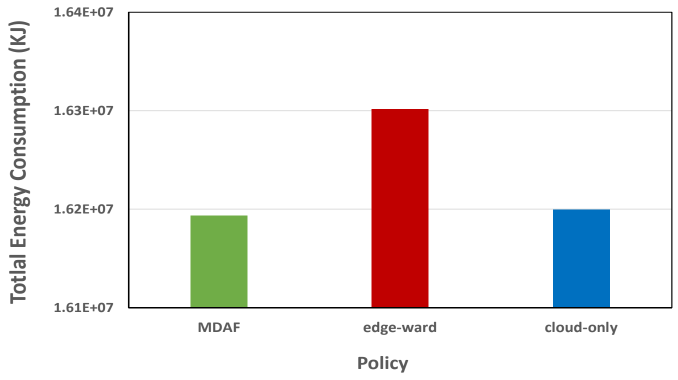
FIGURE 6. The total energy consumed by different devices (mobile phones, edge devices, and the cloud) under various policies (MDAF, edge-ward and the cloud-only policy).



7. CONCLUSION AND FUTURE DIRECTION

In this paper, we propose an efficient heuristic algorithm to solve the service placement problem in fog-cloud computing environments. Our algorithm places the most delay-sensitive application services closer to the IoT devices. We compare the performance of the proposed algorithm with two baselines, edge-ward and cloud-only. Unlike the baselines, the proposed algorithm does not violate deadlines of applications. Furthermore, our algorithm is more effective utilizing fog landscape resources. More specifically, the proposed approach reduces the execution cost by about 37% and 62% compared to edge-ward and cloud-only policies, respectively. In the future, we plan to concentrate on the energy efficiency aspect of the problem to provision resources more effectively in fog-cloud computing paradigm.

Figure 7. The total energy consumed by different devices (mobile devices, fog devices, and the cloud) undervarious policies (MDAF, edge-ward and the cloud-only policy).



References

[1] M. Aazam, I. Khan, A. A.Alsaffar, and E. N. Huh, *Adaptive finite volume methods for hyperbolic problems*, Proceedings of 2014 11th International Bhurban Conference on Applied Sciences & Technology (IBCAST) Islamabad, Pakistan, 14th-18th January, 2014 (Uxbridge, 1993), Wiley, Chichester, 1994, 289–297.

[2] F. Bonomi, *The smart and Connected Vehicle and the Internet of Things*, Invited Talk, Workshop on Synchronization in Telecommunication Systems, 2013.

[3] M. M. Mahmoud, J. J. Rodrigues, K. Saleem, J. Al-Muhtadi, N. Kumar, and V. Korotaev, *Towards energy-aware fog-enabled cloud of things for healthcare*, Computers & Electrical Engineering, vol. 67, pp. 58-69, 2018.

[4] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, *iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments*, Software: Practice and Experience, *47*(9) (2017), 1275–1296.

[5] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, *Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption*, IEEE Internet of Things Journal, *3*(6) (2016), 1171–1181.

[6] O. Skarlat, S. Schulte, M. Borkowski, and P. Leitner, *Resource provisioning for IoT services in the fog*, 2016 IEEE 9th international conference on service-oriented computing and applications (SOCA), (2016), 32-39.

[7] O. Skarlat, M. Nardelli, S. Schulte, and S. Dustdar, *Towards qos-aware fog service placement*, 2017 IEEE 1st international conference on Fog and Edge Computing (ICFEC), (2017), 89-96.

[8] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner,*Optimized IoT service placement in the fog*, Service Oriented Computing and Applications, *11*(4) (2017), 427-443.

[9] A. Yousefpour, A. Patil, G. Ishigaki, I. Kim, X. Wang, H. C. Cankaya, Q. Zhang, W. Xie and J. P. Jue, *Qos-aware dynamic fog service provisioning*, arXiv preprint, arXiv:1802.00800.

[10] M.-Q. Tran, D. T. Nguyen, V. A. Le, D. H. Nguyen, and T. V. Pham, *Task Placement on Fog Computing Made Efficient for IoT Application Provision*, Wireless Communications and Mobile Computing, *2019* (2019), Article ID 6215454, 17 pages.

[11] Q. T. Minh, D. T. Nguyen, A. Van Le, H. D. Nguyen, and A. Truong, *Toward service placement on Fog computing landscape*, 4th NAFOSTED conference on information and computer science, (2017), 291-296.

[12] S. K. Mishra, D. Puthal, J. J. Rodrigues, B. Sahoo, and E. Dutkiewicz, *Sustainable Service Allocation Using a Metaheuristic Technique in a Fog Server for Industrial Applications*,IEEE Transactions on Industrial Informatics, *14*(10) (2018), 4497-4506.

[13] R. Mahmud, K. Ramamohanarao, and R. Buyya, *Latency-aware application module management for fog computing environments*, ACM Transactions on Internet Technology (TOIT), *19*(1) (2018), Article No. 9, 21 pages.

[14] C. Guerrero, I. Lera, and C. Juiz, *On the Influence of Fog Colonies Partitioning in Fog Application Makespan*, 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud), (2018), 377-384.

[15] N. K. Giang, M. Blackstock, R. Lea, and V. C. Leung, *Developing iot applications in the fog: A distributed dataflow approach*, 2015 IEEE 5th International Conference on the Internet of Things (IOT), (2015), 155-162.