Received: 5 February 2021 | Revised: 8 August 2021 | Accepted: 5 September 2021

DOI: 10.1002/ett.4368

RESEARCH ARTICLE

WILEY

DEEDSP: Deadline-aware and energy-efficient dynamic service placement in integrated Internet of Things and fog computing environments

Meeniga Sri Raghavendra[1] | **Priyanka Chawla**[1] | **Sukhpal Singh Gill**[2]

[1]School of Computer Science and Engineering, Lovely Professional University, Punjab, India

[2]School of Electronic Engineering and Computer Science, Queen Mary University of London, London, UK

Correspondence
Priyanka Chawla, School of Computer Science and Engineering, Lovely Professional University, Punjab, 144001, India.
Email:priyankachawla.cse@gmail.com

Abstract

Fog computing has become adaptable and also as a promising infrastructure for providing elastic resources at the edge of the network. Fog computing reduces the transmission latency and consumption of bandwidth while processing the incoming requests from various Internet of Things (IoT) devices. Moreover, fog computing can support and facilitate geographically distributed applications with low and predictable latency. However, this technology also has significant research issues in its current stage such as successful implementation of service location models. In this article, we propose a deadline-aware and energy-efficient dynamic service placement (DEEDSP) technique for fog computing that supports the placement of IoT based services. Further, hyper-heuristic algorithm based energy-efficient service placement technique is proposed to balance the energy-delay trade-off based on different service placement decision criteria (eg, minimum response time or energy consumption). The proposed algorithm is able to dynamically minimize the energy consumption of the system while ensuring that the response time satisfies a given time constraint. Finally, the proposed technique is evaluated in simulated fog computing environment and experimental results show that this technique performs better than state-of-the-art placement techniques in terms of energy and latency.

1 | INTRODUCTION

Within the next five years, the estimated number of Internet of Things (IoT) objects would have reached 50 billion.[1] These objects include power and computation-hungry devices such as wearable's, autonomous vehicles, drones, robots, augmented reality (AR), and virtual reality (VR) gadgets.[2] The seamless combination of all "things" connected in the network presents a challenge at a scale which it is never seen before. With the sheer increase in the amount of data traffic generated by billions of these IoT devices,[3] low latency and energy efficiency have become impractical for the traditional cloud computing model under time-critical requirements.[4] The idea is to add a new layer of networking, storage, computation and resources at the edge of the network or at end users which is now known as fog computing.[5]

Fog environment is needed for the optimization of Quality of Service (QoS) parameters such as minimizing service latency, increasing service efficiency and providing end user service with improved overall experience.[6] For example, sensing and actuation modules are the IoT services in an IoT application which interacts with the IoT or edge devices to exchange data.[6] These features are especially useful in real-time applications[7] such as intelligent light systems, vehicle

networks, smart grid, pipeline monitoring, wired trains, wind farming, applications in the petroleum and gas industry, and industrial loop control. The current central computer paradigm, where IoT system's control, data and intelligence are only accessible at the cloud,[8] is increasingly being transformed into a distributed computing paradigm. However, the achievement of those criteria requires a smart choice of hosting systems, that is, the applications must be properly assigned.

1.1 | Motivation and our contributions

The decision-making in application placement is an NP-hard problem.[9-11] The difficulty of placing the modules in fog computing environment when compared to cloud becomes more complicated.[12] The reasons for the increased complexity are: (i) diversity of devices (in terms of configuration of the devices, hardware and also software, such as operating systems), (ii) location of the IoT devices required to be considered in the IoT application placement, and a massive amount of these devices, further escalating the complexity in placement, and (iii) specific application's conditions must be satisfied, such as, computation and delay requirements.

The massive scale and rising demands for service, raise the power usage on cloud servers. The energy conservation of the fog computing system is also important and must be taken into consideration.[13] On the other hand, it is equally important to guarantee the QoS such as response time of the application within the given deadline.[14] We systematically inquire the trade-off between the power consumption and response delay in the fog computing system. Based on the most applicable and new techniques published in the field of IoT, cloud computing, and fog computing, a framework for deadline-aware and energy efficiency in service placement strategy is proposed in this article. The proposed method adds novelty to the fog-computing environment by making effective decisions dynamically.

The proposed hyper heuristic-based algorithm incorporates various heuristic techniques. This algorithm makes use of the fortes of heuristic algorithms, such as simulated annealing (SA),[15] genetic algorithm (GA),[16] particle swarm optimization (PSO),[17] and integrates all of them into a single algorithm. This proposed algorithm aims to look for service placement inside fog nodes, the fog controller node, neighbor controller node, and the cloud node. Minimizing the energy consumption with respect to the constraint; the application response time should not exceed the given deadline; is the main objective of the article. The *main contributions* of this article are:

- We proposed a hyper-heuristic based a deadline-aware and energy-efficient dynamic service placement (DEEDSP) technique for IoT devices which prioritize different applications of the user and place the application modules in fog environment.
- DEEDSP dynamically makes the decisions for mapping the application services in fog computing environment to satisfy the application deadline and reduce energy consumption.
- iFogSim simulator is used to evaluate the performance of DEEDSP in fog computing environment and demonstrated that there is a significant improvement in reducing the energy consumption and service response time as compared to existing techniques.

1.2 | Article organization

The rest of the article is structured as follows: In Section 2, we discuss the relevant work in the area of service placement in fog computing environment and the analysis of those works. In Section 3, we discuss how the system is modeled. We present the proposed technique in Section 4. Afterwards, we present the experimental results in Section 5. Finally, Section 6 concludes the article and highlights the future directions.

2 | RELATED WORKS

The positioning in fog nodes is a significant problem particularly in the case of limited resource devices. Several experiments were undertaken to address the issue of service placement in heterogeneous computing systems.[18,19] These experiments attempt to find the best location of modules in an optimized way. Parameters that an optimal placement

strategy must keep in mind are latency, reducing the energy consumption, minimize the cost and response time.[20,21] There are many existing works related to service placement in cloud/fog computing. Rahman et al[7] sum up and outline the architecture of the fog computing model, functionality, related paradigms, security challenges, and various real-time applications such as smart grid, traffic control and increased reality.

Qi et al[8] has developed a cloud, fog and edge computing based hierarchical architecture. The proposed fog system is composed of three levels of mobile-fog-cloud; mobile users are provided with service from fog servers via local Wi-Fi links and fog servers are upgraded to cloud content through cellular or wired networks. For IoT-based applications, Taneja and Davy[22] presented an efficient deployment of fog-cloud infrastructure. To allow application modules to be deployed on devices in fog layer close to the source, an image is distributed dynamically across the fog and cloud layers. The outcome of this study is a microbenchmark in IoT and fog computing observation. This article is intended to serve as a benchmark for IoT applications with Quality of Service (QoS).

Xuan and Huh[23] introduced an almost similar cloud-fog computing system that combines fog nodes and cloud nodes which are owned and leased from cloud providers. In order to benefit from this cloud-fog computing system, the processing nodes of each layer are strategically assigned with the computing tasks. The algorithm proposed by the author is not only guarantees the application efficiency, but also decreases the expense of accessing cloud services. Gupta et al[24] proposed the iFogSim simulator, to model and calculate the effect on latency, network usage, energy usage, and cost of the resources. In this article, the comparison of resource management policies and modeling of IoT environment are manifested. In many situations, with respect to computing size, storage and RAM consumption, the scalability of the simulation toolkit is assured.

Recent research works have revealed the significance and pertinence of fog computing.[20] However, in terms of implementation view, most of the works have not bestowed up on the technical aspect of the paradigm.

Pham et al[25] did the task offloading of directed acyclic graph (DAG)-based applications with the trade-off between cloud-based computing execution time and cost when placing applications in the cloud-fog environment. The proposed cost-aware placement algorithm satisfies the deadline restrictions of the application. Mahmud et al[26] focused on executing the applications of fog environment which can be decomposed into modules, which might independently be executed. Ensuring the QoS applications in completing before the deadline for service delivery and maximize the resource use in the fog environment is the main purpose of the proposed algorithm.

The above existing algorithms are unable to minimize the response time. The reason for such inefficiency is, they executes the applications according to their arrival sequence. In order to cope with the placement of IoT applications over fog environment. The authors Huang et al[27] has presented a placement solution while minimizing the parameters latency and cost. This multiobjective problem has been solved with an ant colony optimization (ACO) based meta-heuristic approach. Service orchestration is done at the cloud node. Skarlat et al[28,29] proposed an optimization problem while taking into consideration the diversity of resources and applications. For addressing the application modules placement, the authors proposed two approaches to solve the optimization problem, one is using linear integer programming in Reference 28 and another is using the genetic algorithm (GA) in Reference 29. Choudhari et al[30] proposed a prioritized task scheduling algorithm to minimize overall cost and response time of the application; this algorithm will assign a priority based on its deadline. It is located in the fog layer using this determined task priority. In each fog layer, there are many computational fog nodes that can communicate with the other fog nodes of the same fog layer. If all the fog nodes in the fog layer are infused, the task would be shifted to the cloud. The algorithm decreases overall computing cost and response time with the increasing number of modules.

Sriraghavendra et al[31] proposed a deadline-aware service placement (DoSP) algorithm, which ensures the response time is satisfied for a given time constraint. The DoSP algorithm plans to place the application modules in fog-cloud architecture. A profitable application placement strategy for the integrated fog-cloud environment is proposed by Mahmud et al,[32] which maximize the providers revenue and decreases the deployment cost and application deadline.

The works,[27-31] and[32] do not take the energy efficient workflow execution into account. A placement algorithm needs to be built that can reduce response time and meet deadlines for sensitive latency applications, while minimizing energy consumption. Naranjo et al[33] introduces a penalty-aware bin packing (PABP) heuristic algorithm for the energy minimization. This is accomplished by scaling up or down the processing speeds of virtual processors. The final goal is to reduce the total energy per slot.

Ramirez et al[34] evaluated fog-to-cloud (F2C) systems efficiency to demonstrate the architecture's advantages which include edge as well as cloud devices. On the three separate architectural scenarios, the proposed policy evaluates the performance of response time, power and bandwidth usage. The proposed design reduces the power consumption than stand-alone cloud. Wu et al[35] proposed the energy minimization scheduling algorithm to place the IoT workflow

applications in fog environment to minimize the consumption of energy. The energy consumption is reduced when compare with the random, integer linear programming (ILP) for different workloads. The above existing works[29-31,33,34] and[35] suffer from service placement optimization while minimizing energy and prevent hitting the application deadline. To overcome the above disadvantages, we proposed a service placement algorithm that is deadline-aware and energy-efficient in this article.

Kim et al[36] has suggested a power-aware algorithm for independent application modules. The authors considers the time limit on dynamic voltage scaling (DVS)-enabled systems to lessen power usage and meet deadlines. Deng et al[37] tackled difficulties in the allocation of workload. These may be formulated to the minimum consumption of energy and the delay in service. Their methodology is intended to determine the best workload distribution between fog and cloud layers. The author concentrates on static service planning which are predefined as the collection of services that are to be allocated. The authors do not adequately address the necessary information sharing and communication overhead.

Sharma et al[38] suggests 4-tier energy consumption architecture and deliberates scheduling of delays in the fog environment. The author prioritizes the applications based on their deadline. The priority for execution is given to the nodes that are least loaded that are closer to the user. Mingfeng et al[39] proposed a task offloading scheme with service orchestration to enhance the cloud-MEC's energy consumption and latency among mobile edge computing (MEC) server and cloud resources. The works,[36-38] and[39] mentioned above, does not consider the intertask dependencies and fog cluster head cooperation in the environment. While providing a performance assessment of cloud-fog architecture, the work also aims at portraying its benefits. This analysis is undertaken with respect to application response time, understanding of deadlines and energy usage.

Our research work provides uniqueness in two perspectives, namely:

- A modeling perspective of constraints (operational and nonoperational) optimization metrics have also been taken into account to determine best application placement.
- An algorithm and contribution methodological approach is used to deploy the application modules in fog environment.

2.1 | Critical analysis

Table 1 compares proposed work (DEEDSP) with existing works based on important key parameters such architecture, application and placement approach.

- The IoT application section discusses each proposal's mode of dependence and how each proposal is modeled according to the number of applications and modules.
- In the architectural section, we studied the number of fog layers and fog cluster head cooperation.
- The placement properties, specifies the prioritized placement of application (or) node (or) module, and considered the application placement approach for fog environment.
- To the best of our knowledge, no hyper heuristic algorithmic approach with fog cluster co-operation was proposed.[6] This framework also jointly considers the deadline of IoT applications and also energy consumption of the system.

3 | MODELING

This section discusses the proposed technique for effective application placement in fog computing.

3.1 | DEEDSP: System model

In the architecture shown in Figure 1, the fog environment's computational nodes are ordered in a three-tier hierarchical layered architecture.

The IoT devices such as sensors and actuators, are connected to lower-level fog nodes which have the capability of computing, storage and networking. In the fog environment, there is a fog node that performs the controlling functionality. This node is called fog controller node. The activities of the fog controller node are: (1) receive the user requests;

TABLE 1 Comparison of proposed technique (DEEDSP) with existing works

Application properties			Architectural properties		Placement properties			
							Optimization parameters	
Work	No. of apps & No. of modules	Module type	Fog layers	Fog cluster cooperation	Prioritized	Approach	Deadline	Energy
[22]	Single and multiple	Dependent	Multiple	No	-	Heuristic	No	No
[23]	Single and multiple	Dependent	Single	No	-	Heuristic	No	No
[24]	Single and multiple	Dependent	Single	No	-	Heuristic	No	No
[25]	Single and multiple	Dependent	Single	No	Task	Heuristic	No	No
[26]	Multiple and multiple	Dependent	Multiple	No	App	Heuristic	Yes	No
[27]	Single and multiple	Dependent	Single	No	-	Heuristic	Yes	No
[29]	Multiple and multiple	Dependent	Multiple	Yes	App	Heuristic	Yes	No
[30]	Multiple and single	Independent	single	No	App	Heuristic	Yes	No
[31]	Multiple multiple	Dependent	Multiple	Yes	App	Heuristic	Yes	No
[32]	Multiple and single	Independent	Single	No	-	Linear programming	Yes	No
[33]	Single and multiple	independent	single	Yes	-	Heuristic	No	Yes
[34]	Multiple and single	Independent	Multiple	No	-	Heuristic	No	Yes
[35]	Multiple and single	Independent	Single	No	-	Linear programming	No	Yes
[36]	Multiple and single	Independent	Single	No	App	Heuristic	Yes	Yes
[37]	Multiple and single	Independent	Single	No	-	Linear programming	Yes	Yes
[38]	Multiple and single	Independent	Multiple	No	App	Fuzzy logic	Yes	Yes
[39]	Multiple and single	Independent	Single	No	App	Heuristic	Yes	Yes
DEEDSP	Multiple and multiple	Dependent	Multiple	Yes	App	Hyper heuristic	Yes	Yes

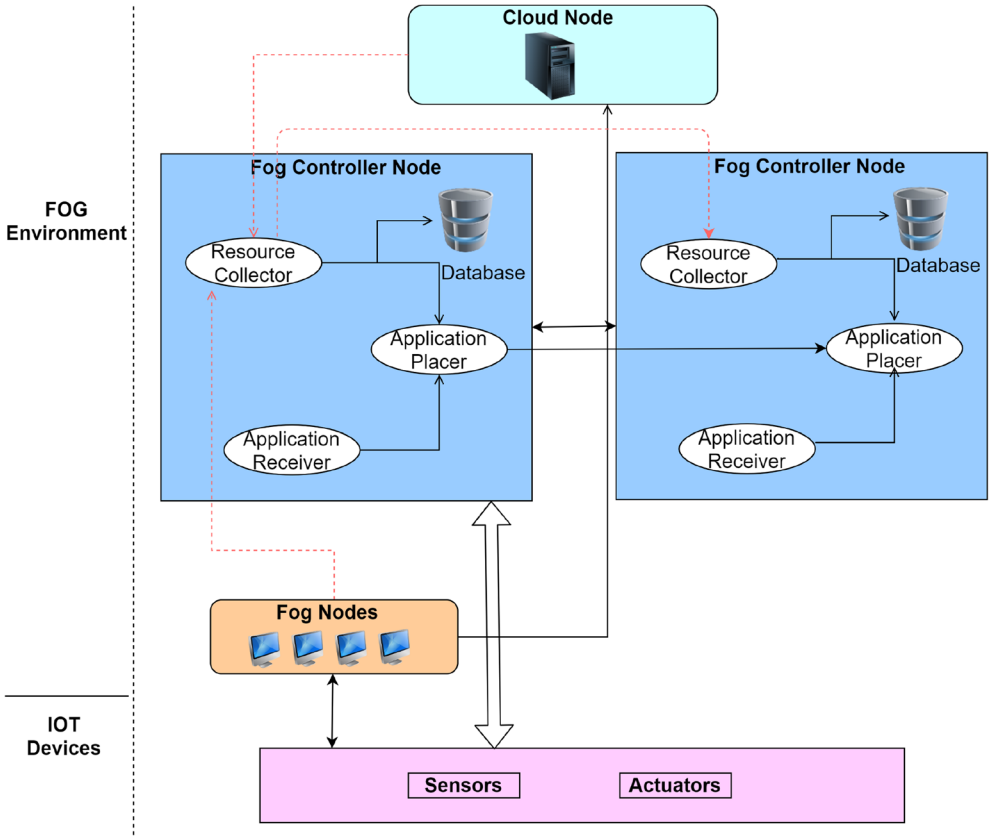


FIGURE 1 DEEDSP architecture

(2) control the available resources on fog nodes and cloud node, such as computing power, memory and storage; and (3) it identifies the most suitable placement for an application. It determines which application module will be installed on which computing node. The fog controller node connects the cloud node, the fog nodes and the neighbor controller node in order to run the modules for utilizing the resources. The main components of fog controller node are described below.

The responsibility of receiving the application request from the user is taken by a component called application receiver. The parameters and details such as module count, workload, and data input size are defined for each application reaching the controller node. Fog controller node (FCN) administers to locate the appropriate positioning of module in the fog computing infrastructure.[41]

Resource collector takes responsibility for collecting and managing information about the current system state of all processing nodes and storing it in resource database. Frequent state information changes along with the resource addition or elimination on the computing infrastructure.[41] This ensures that the final application placement produced by fog controller node is matched with the current updates on the resource consumption of computing nodes and therefore results in the greater precision.

The application placer analyses the application, decides the required location and distributes the modules of each application to a suitable computing node based on knowledge about processing power and delay of communications of all the computing nodes. The computing nodes that are considered in the proposed architecture are fog, fog controller node, neighbor controller node, and cloud. The frequency of the sensors is seen as the same for convenience, and the fog nodes of the same organizational levels are assumed to be of the same kind.

Application response time:

The optimization problem is formulated to minimize the response time and meet requirements such as deadline and energy usage.

$Res_{\text{time}}(n)$, is the response time of an nth application. The application response time is calculated as follows:

$$Res_{\text{time}}(n) = MKspan_{\text{time}}(n) + Total_{\text{time}}(n). \tag{1}$$

Where

- The application makespan time, $MKspan_{\text{time}}(n)$, consists of the time needed to execute all application modules, $Exec_{\text{time}}(n)$, and the time required for communication, $Comm_{\text{time}}(n)$, with the application modules from the controller node.
- The total deployment time of an application $Total_{\text{Dept}}(n)$ takes into account the time elapsed before the proper location of each service on the computational framework resource sources or the computing nodes.

$$MKspan_{\text{time}}(n) = Exec_{\text{time}}(n) + Comm_{\text{Dept}}(n). \tag{2}$$

$$Exec_{\text{time}}(n) = \frac{Module\ Capacity_n}{Node\ Capacity_n}. \tag{3}$$

The execution time, $Exec_{\text{time}}(n)$, is the module capacity over node capacity. Module capacity is "the required CPU power for the moduleε and the Node capacity is "the CPU power of the computational node where the application service is placedε. The communication time is the sum of undeniable communication delays from the nodes where each module in an application is placed. The variables $dist_{\text{fog}}$, $dist_{\text{nbr}}$, $dist_{\text{ctrl}}$, and $dist_{\text{cloud}}$ represent the distances from the FCN to the fog node (FN), the neighbor control node (NFCN), the controller node (FCN), and the cloud node (CN) respectively. The fog controller node acts as manager, which places the modules in the suitable nodes. The results from the nodes are returned back to the controller node. Hence, the communication happens twice between the nodes.

$$MKspan_{\text{time}}(n) = \sum_{i=0}^{m} \{Exec_{\text{time}}(n_i) + dist_{\text{fog}} * x_{\text{fog}} + 2 * dist_{\text{nbr}} * x_{\text{nbr}} + dist_{\text{ctrl}} * x_{\text{ctrl}} + 2 * dist_{\text{cloud}} * x_{\text{cloud}}\}. \tag{4}$$

In Equation (4), m is the number of modules in each application. Total deployment time of the application $Total_{\text{Dept}}(n)$ consists of $Dept_{\text{time}}(n)$ and the estimated extra time when an application module n_i is sent to the adjacent controller node. Here n_i represents i^{th} module in n^{th} application. The $x_{\text{fog}}, x_{\text{nbr}}, x_{\text{ctrl}}, and x_{\text{cloud}}$ are binary decision variables. The value of the these variables will be 1 if the n_i is placed in the appropriate node, else the value is 0. The additional deployment time comprises of delay for propagation $P_{\text{Delay}}(n)$ and expected deployment time $Expt_{\text{Dept}}(n)$ in neighbor controller node. We depend on the variable $F(n)$ to cover the additional deployment time. If the neighboring controller node has at least one application module of n^{th} application propagated then $F(n) = 1$ otherwise $F(n) = 0$. If $F(n) == 1$, we add $P_{Delay}(n)$ and $Expt_{\text{Dept}}(n)$ to $Dept(n)$. Else we add nothing to $Dept(n)$. We formalize the total deployment time of an application $Total_{\text{Dept}}(n)$ as follows:

$$Total_{\text{Dept}}(n) = Dept(n) + \begin{cases} P_{\text{Delay}}(n) + Expt_{\text{Dept}}(n) & : F(n) = 1 \\ 0 & : F(n) = 0 \end{cases}. \tag{5}$$

3.2 | Energy consumption model

In this article, we are considering the energy consumption of computational nodes only and not considering the communication energy and cooling energy.

$$E_{\text{node}} = E_{\text{static}} + E_{\text{dynamic}}, \tag{6}$$

$$E_{\text{dynamic}} = (E_{\text{max}} - E_{\text{static}}) * Util, \tag{7}$$

$$E_{\text{total}} = \sum_{i=0}^{m} E_{\text{FN}} + E_{\text{FCN}} + E_{\text{NFCN}} + E_{\text{CN}}. \tag{8}$$

The goal of our article is to minimize overall energy consumption, that is, the total energy consumption in working conditions and energy consumed in idle condition. The static energy is depleted when the computation node is inactive,

and the dynamic energy refers to the host usage of resources. *Util* illustrates the host node's consumption point. E_{max} is the nominal power implied by dissipating the full power unit. Here m is the number of modules in each application.

3.3 | Application model

An application should be prior partitioned into modules, before it is needed to be executed on the fog environment. The collections of interdependent modules promote the distributed application concept. Each module in the application has importance and certainly performs at least some functionality in the application. The connection between two application modules is defined as application edges; if applications have an edge between them it indicates they both are dependent on each other. Distributed data flow model is a way of representing the application dependencies and data flow in a directed graph form; either sequential or unidirectional.

Application data flow model for the placement policy:

As illustrated in Figure 2, the IoT applications considered in this article consists of five major modules (tasks) which perform processing—sense module, process module 1, process module 2, process module 3, and actuate module. Every module in an application is characterized as a set of dependent tasks. They are run in a sequential mode. In Figure 1, the sensor transmits data from the Sense module to the application module. This module manages authentication, data receiving frequency standardization, and multisensor data aggregation. The data coming from Sense module is processed in process modules. The application incorporates multiple functionalities. Therefore, each processing module can perform at least single functionality on the data. The actuate module determines and manages the activities of the associated actuator. The app modules are simulated with an AppModule class in iFogSim simulator. As shown in Figure 2, data dependency between modules were developed using the AppEdge class in iFogSim simulator.

Sequence flow of application service:

Figures 3 illustrates the concept where application modules are placed to VMs in fog node or cloud node or fog broker 1 or fog broker 2. Then, after processing the application modules, they are sent back to fog broker 1 to merge and send them to user.

Step 01: The mobile user requests for the application, this request is handled by the fog node which it is connected.
Step 02: The fog node forwards the request to the fog broker 1.
Step 03: Fog broker 1 extracts and sequences the modules of the requested application.
Step 04: Fog broker 1 finds the required resources of the modules for execution.
Step 05: Fog broker 1 finds the resource availability information from itself, fog node, fog broker 2, and cloud node.
Step 06: Fog broker 1 runs the placement algorithm to find the optimal module placement.
Step 07: The module assign nodes are responsible for processing the assigned module.
Step 08: The assigned nodes send the results of the modules back to the Fog broker 1.
Step 09: Fog broker 1 combines the results which are send by the module assigned nodes.
Step 10: The response send to the mobile user through the fog node which it is connected.

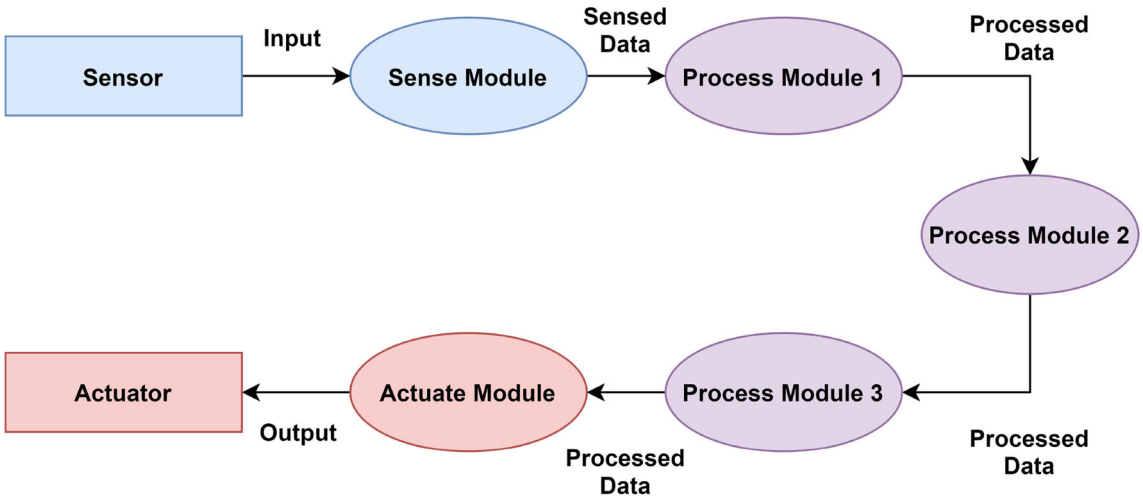


FIGURE 2 Application Dataflow model

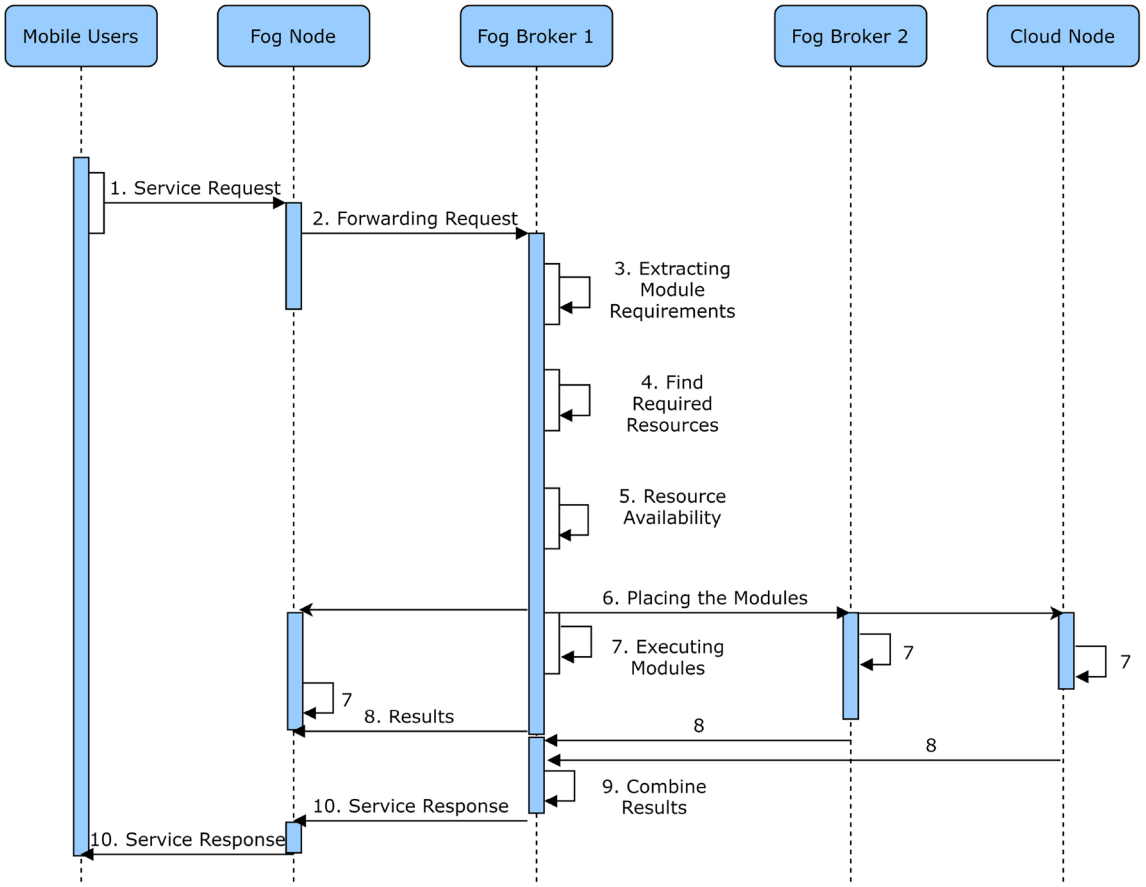


FIGURE 3 Application dataflow model

Application process model:

As shown in Figure 4, applications are buffered in the application queue at the fog controller node to achieve the objectives of improving response time and energy efficiency of all the applications. The application buffering policy can be performed on the application queue. We propose a parallel virtual queuing model at the fog controller node that

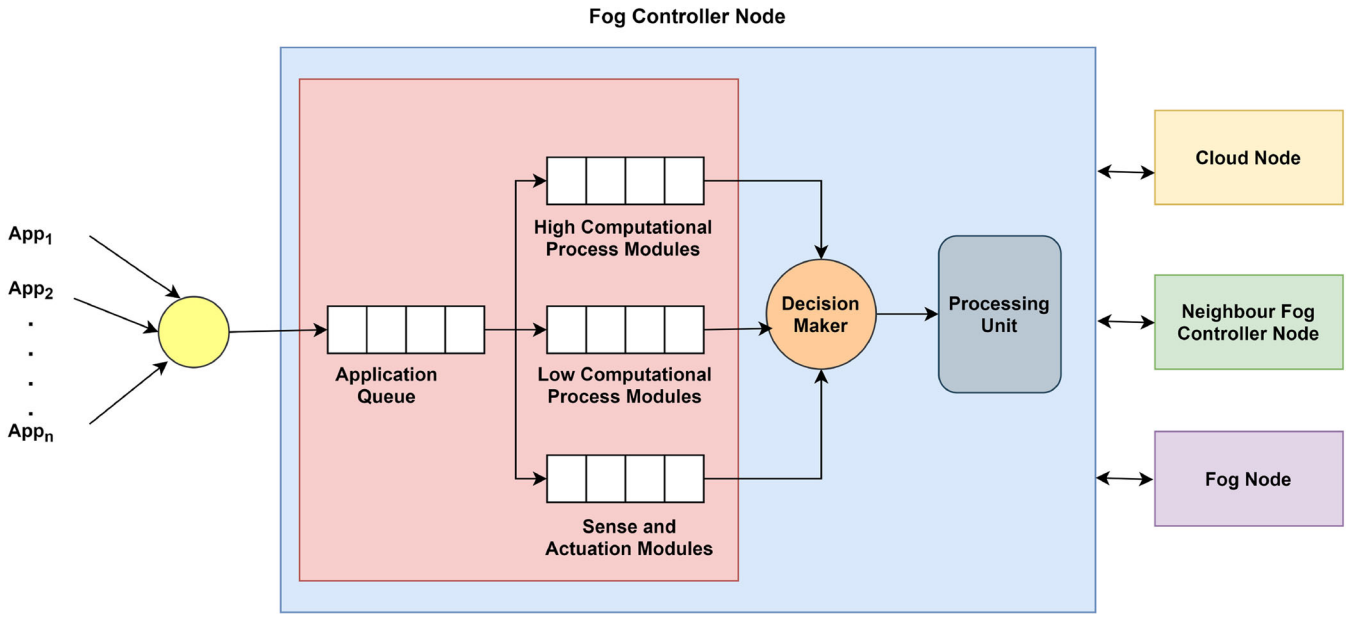


FIGURE 4 Process model

buffers the arrival application modules of the same type into a separate virtual queue as shown in Figure 1. Based on the framework, multiple IoT applications exist in the fog environment system.

4 | PROPOSED ALGORITHM

Depending on the application deadline and energy usage, a hyper heuristic is used to put IoT applications in a fog environment. In reality, existing methods used to process the modules in the fog layer. Our strategy targets to position the delay-tolerable applications in the cloud, and neighbor node, whereas delay-sensitive applications are placed in low computational fog nodes. Comparing cloud with fog, the cloud has greater computational capacity resources, with high communication latency between cloud and device layer. On the other hand, fog has limited computational resources, with lower communication latency between the fog and the device layer. Furthermore, during the placement process, our approach takes application deadlines and energy consumption into account. The data considered in running the proposed algorithm is synthetic.

$$Objective\_function = \min \left\{ \sum_{n=1}^{N} \{Res_{\text{time}}(n) + E_{\text{cons}}(n)\} \right\}, \tag{9}$$

where, n is the index of modules in each application and N is the number of applications. The aim of the proposed algorithm is to minimize the objective function presented as above. There are three steps of the planned placement strategy:

1. Choosing an application
2. Choosing the module
3. Choosing the node

These phases are explained in the following subsections.

4.1 | Application selection phase

Step 1: Prioritization of applications is done according to the deadline and deployment time. The application that has least deadline-deployment time has the highest priority. If multiple ready tasks have the same priority, then an application with the first come first serve (FCFS) strategy is selected. The application priorities A_{pri} is given by

$$A_{\text{pri}} = D(n) - Dept(n). \tag{10}$$

4.2 | Module selection phase

As soon as the application is selected, the modules of the application are extracted. Later, we check how the application modules are to be categorized. This is achieved as follows:

Step 2: Selected application of sense and actuation modules are placed in sense and actuation modules queue.
Step 3: Place the process modules of the application either in high computational process modules or low computational process modules queue, based on the CPU value of the process modules.

The step 2 and step 3 as discussed above are presented in the Algorithm 1. The selected heuristic algorithm H_i will evolve the solution Z for specified number of iterations by using the Govern function (H_i; Imp_Flag; Div_Flag), as defined in Algorithm 2 and Algorithm 3.

Algorithm 1. Deadline-aware and Energy-Efficient Applications Placement

```
input : Z{}, H{}, APP[N][M]
Initialize Threshold W.
 Calculate Application Priorities, AppsortedbasedonA_pri[].
 Construct MinHeap(App). ;
 for (i=0;i<N;i++) do
    EnQueue(SA_Queue,App[i][0],App[i][M-1]);
     for (j=1;i<M-1;j++) do
        if (App[i][j] > W) then
            EnQueue(HC_Queue,App[i][j]);
        else
            EnQueue(LC_Queue,App[i][j]);
call Module_Placement(Z,H);
```

4.3 | Node selection phase

Step 4: Algorithm 1 calls the *ModulePlacement* function to place the modules in the selected nodes. The *ModulePlacement*() choose the nodes as per satisfying the constraints shown below for application module placement.

Let $place^{\text{fog}}(n_i)$, $place^{\text{ctrl}}(n_i)$, $place^{\text{nbr}}(n_i)$, $place^{cloud(ni)} \in \{0,1\}$ be binary factors that specifies whether the module n_i is deployed on a fog node ($place^{\text{fog}}(n_i) = 1$), or a fog controller node ($place^{\text{ctrl}}(n_i) = 1$), or a neighbor controller $Nodeplace^{\text{nbr}}(n_i) = 1$), or on the cloud node ($place^{\text{cloud}}(ni) = 1$). Since the module is deployed only once, in the $constraint_1$ is held:

$$constraint_1 \; : \; \{place^{\text{fog}}(n_i) + place^{\text{ctrl}}(n_i) + place^{\text{nbr}}(n_i) + place^{\text{cloud}}(n_i)\} \tag{11}$$
$$\forall n_i \in n, n \in N.$$

The $constraint_1$ value will be 1, because the module will be placed in at least one of the available nodes. The module placement strategy must be assured that $Res_{\text{time}}(n)$ as specified in the $constraint_2$, application does not compromise its deadline, $D(n)$.

$$constraint_2 \; : \; \{Res_{\text{time}}(n) < D(n), \forall n \in N\}. \tag{12}$$

The $constraint_3$ in this article is that, available resources, $Avail_{\text{res}}(r, nodes)$, such as processing power (CPU), storage space (STR), memory capacity (MEM) must be sufficient to required resources, $Req_{\text{res}}n_i$, of the application in the deployment node. This can be defined as:

$$constraint_3 \; : \; \left\{ \sum_{n}^{N} \sum_{n_i}^{n} Req_{\text{res}} n_i <= Avail_{\text{res}}(r, \text{nodes}) \right\} \tag{13}$$
$$r \in \{\text{CPU, MEM, STR}\}, \text{nodes} \in \{\text{fog, ctrl, nbr, cloud}\}.$$

The $constraint_4$ in this article represents the sensing (x_i) and actuation (y_i) modules of the application that should be placed in lower fog nodes only. It is defined as below:

$$constraint_4 \; : \; \{place^{\text{fog}}(n_{x_i}) \;\; \&\& \;\; place^{\text{fog}}(n_{y_i})\} = 1. \tag{14}$$

The above $constraints_{1,2,4}$ are applicable independently for each application.

Step 5: The Algorithm 2, called module placement is shown below. First two of the input parameters of the algorithm are population of the solution and the candidate pool. The population of the solution is specified the placement

plans of the application modules in the computational nodes. The module placement algorithm sequentially selects the heuristic algorithm H_i from the candidate pool H. The set H comprises three hyper heuristic algorithms, genetic algorithm, particle swarm optimization, simulated annealing, $H = \{GA, PSO, SA\}$. The selected heuristic algorithm H_i will then be performed repeatedly until the maximum iterations is reached or if there is no improvement, as shown in Algorithm 2.

The *Module Placement* algorithm uses the *Diversity_Detection*() and the *Improvement_Detection*() functions. During the convergence phase it uses those two functions to balance the strengthening and variegation in the search of the solutions. The flags *Improvement_Detection*, *Imp_Flag*, *Div_Flag*, and *Diversity_Detection* determine if a new heuristic algorithm is to be chosen. *Imp_Flag* is alone used for heuristic algorithms that are based on single solution, while heuristic algorithms that are based on population both these *Imp_Flag* and *Div_Flag* are used.

Step 6: The *Module Placement* algorithm will randomly select the next algorithm H_i from H whenever the *Govern*() function returns *True*. Then the solution Z is obtained, as shown in Algorithm 3, to refine the solution.

Algorithm 2. Module Placement

input : $Z\{\}, H\{\}$, *Max_iterations*, *Not_improve*
Select a Heuristic Algorithm H_i from set H.
while *($Max_{iterations}$ reached or $Not_{improve}$ is reduced to zero)* **do**
  Using H_i, improve the population of solutions Z ;
  Imp_Flag = Improvement_Detection() ;
  Div_Flag = Diversity_Detection() ;
  Flag = Govern(H_i, *Imp_Flag*, *Div_Flag*) ;
  if *(Flag == TRUE)* **then**
    Select next H_i*from set* H;
    $Z = Fine\_Tune(Z)$;

Step 7:

Algorithm 3. Govern

input : H, *Imp_Flag*, *Div_Flag*
S = Single Solution Based Heuristic Algorithms
P = Population Based Heuristic Algorithms
if *($(H \in S$ && Imp_Flag == TRUE) || $(H \in P$ && Imp_Flag, Div_Flag == TRUE))* **then**
  return FALSE ;
else
  return TRUE ;

In Algorithm 4, while selecting the next heuristic algorithm H_i from the candidate pool H a sequential selection method is employed. The Algorithm *Module_Placement*, suggests when to change to the next heuristic algorithm from the list H_i. If after a defined number of iterations, the selected H_i's Not_{improve}, cannot better the fitness value (which is obtained by BSDEE value) then we pick a new heuristic algorithm by returning a *False* value to the calling function. The *Improvement_Detection* function will suggest choosing a new heuristic algorithm in three situations: when iterations exceed the $Max_{\text{iterations}}$, the flag Not_{improve} has been met, and If the condition of termination is satisfied.

Algorithm 4. Improvement_Detection

BSDEE = Best so far Deadline and Energy Efficient
if *(BSDEE is not Improved)* **then**
  return FALSE
else
  return TRUE

Step 8: In Algorithm 5, the *Diversity_Detection*() function determines when to change the heuristic algorithm from the list H. Diversity of the initial solution $D(Z)$ is considered as the threshold value. (ie, fitness values of the initial solution). Average of the fitness function is determined as the diversity of the current solution $D(Z)$. If the threshold value is less than the diversity of the current solution $D(Z)$ and *Module_Placement*() algorithm will then select the next new Heuristic algorithm.

Algorithm 5. Diversity_Detection

CS = Current Solution
if *(CS>Threshold)* **then**
 ∟ return *FALSE*
else
 ∟ return *TRUE*

Algorithm 6. Fine_Tune

input : Z
Send the solutions obtained by H_i to the next chosen H_i as input

Step 9: In Algorithm 6, the newly selected hyper heuristic algorithm receives the fine-tuned solutions obtained by H_i from the *Fine_Tune*() algorithm. This means that the candidate solutions created by the heuristic algorithm H_i can be used to balance the intensification and variegation of the search. More precisely, the change in the solutions of the population is performed by the *Fine_Tune*() algorithm.

Complexity analysis: The running time complexity depends on the size of the application list, which is N. The scheduler needs $O(NlogN)$ time complexity to arrange this list of applications and the space complexity is estimated to be $O(N)$. The time complexity of each heuristic algorithm in list H is $O(SP^2)$, here S specifies the number of sub solutions of each solution of the problem and P indicates the population size. So, if all heuristic algorithms are limited to SP^2 and r is the number of iterations the algorithm takes, then $O(rSP^2)$ is the time complexity of the hyper heuristic algorithm. $O(NlogN + rSP^2)$ is the total computational complexity of the proposed algorithm.

5 | PERFORMANCE EVALUATION

This section discusses the experimental methodology and results.

5.1 | Experimental methodology

There must be three primary dimensions of representation that need to be addressed for the system benchmarking: computational platform, algorithm, and analyzing on set of data. In the proposed work, we have done an extensive evaluation of system with simulation tool and analytics algorithm. There are various simulation tools with its own advantages such as iFogSim,[24] FogNetSim++,[42] MyiFogSim,[43] and YAFS.[44] In this article, we used iFogSim simulator,[24] which is most popular fog computing simulator.[40] Further, iFogSim simulator is the most appropriate simulator for simulating IoT, fog, and cloud computation nodes in a hierarchical architecture. It is ideal choice for evaluating multiple key elements, such as response time and energy consumption of IoT applications.

We used fog nodes, fog controller node, neighbor controller node, and cloud node as the computing nodes for benchmarking. The performance metrics of the proposed work are satisfying applications deadline by minimizing response time and energy efficiency. The overall experimental framework is shown in Figure 5.

For the analytics algorithm, we present hyper heuristic algorithm for applications placement. We have evaluated the proposed work for benchmarking experiments with different existing algorithms.

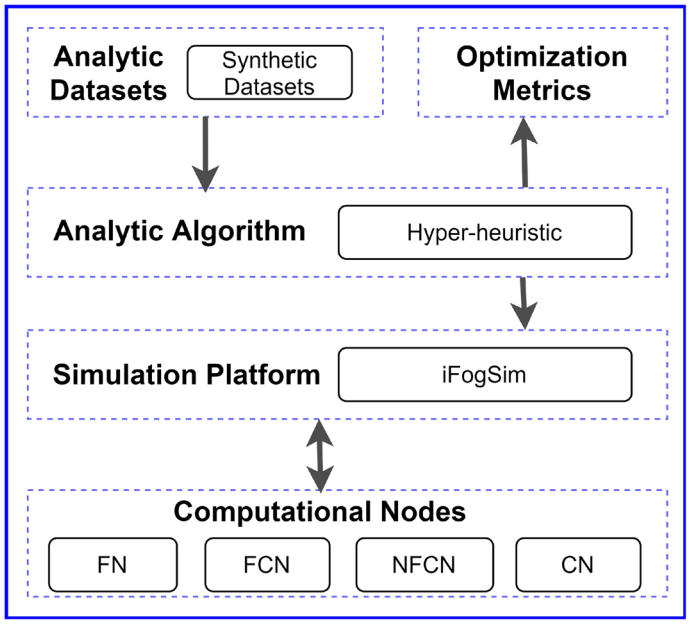


FIGURE 5 Experimental framework

5.2 | Experimental environment

Table 2 provides the experimental environment for the evaluation process of our model. Java with JDK 1.8 is used for developing fog infrastructure with iFogSim simulator. In our simulations, the processing power of processors is expressed by MIPS (million instructions per second).

Network topologies and resource description:

In terms of cloud Network Connectivity from the controller node and applications, the suggested solution proposes module placement plans for various scenarios. The structure of the network topology is as follows: Sensors and Actuators are at lowest level in the hierarchy, the next level contains several fog nodes (FN) controlled by a fog orchestration controller node (FCN), which resides in the next level, at the same level, one neighbor cluster fog controller node (NFCN) is also placed, and a cloud node (CN) is at the top most level. Table 3 shows a setup framework consisting of 1 cluster, with 10 fog nodes, controlled by the FCN and connecting to NFCN and the CN node.

Module configuration and resource demands:

We have analyzed applications such as motion, video, sound, temp, and humidity. These application deadlines, deployment times are mentioned in Table 4.

Each application contains 5 modules. They include sensing, data aggregation, data analysis, and decision making and actuate. These module's processing power (CPU), storage space (STR), and memory capacity (MEM) are given in the Table 5. Concrete examples for these descriptions are presented in the next subsection that describes different application scenarios. Different parameters have been varied to ensure the diversity of the simulations.

Table 6 presents the parameters that influence the results and their values considered in the algorithms GS, DoSP, GA, PSO, SA, and proposed DEEDSP. The GS[29] and DoSP[31] algorithms are GA based algorithms. So, the population size, crossover, mutation and elitism rate of GS, DoSP, GA algorithms are same.

TABLE 2 Simulation setup

System	Intel Core i5-2430 CPU,2.40GHz
Memory	8 GB
Simulator	iFogSim
Operating system	Windows 7 professional
Topology model	Hierarchy

TABLE 3 Characteristics of the computation node

					Placement	
Node type	**Number of nodes**	**Delays (s)**	**CPU (MIPS)**	**RAM (GB)**	**Power max**	**Power idle**
Fog	10	0.2	100	0.5	88.57	80.23
Fog controller	1	0	1000	2	109.33	85.47
Neighbor controller	1	0.5	1000	2	109.33	85.47
Cloud	1	5	10000	4	170.62	110.82

TABLE 4 Applications configuration

Applications	**Deadlines**	**Deployment time**
Motion	120	60
Video	300	0
Sound	300	60
Temp	360	60
Humidity	240	0

TABLE 5 Required resources of application modules

Service module	**Scenario 1 CPU (MIPS)**	**Scenario 2 CPU (MIPS)**	**MEM (MB)**	**STR (MB)**
Sensing module	50	50	30	10
Data aggregation module	200	300	10	30
Data analysis module	200	300	20	30
Decision making module	100	300	30	30
Actuate module	50	50	20	10

TABLE 6 Parameters of GS, DoSP, and DEEDSP algorithms for IoT applications placement

Algorithm	**Parameter values**
GA	Population size = 50, crossover rate = 0.5, mutation rate = 0.02, elitism = 0.2, generations = 10
PSO	Swarm size = 50, acceleration rate = 2
SA	Starting temperature = 10, cooling rate = 0.05
DEEDSP	Max iteration of low-level algorithm = 50, nonimproved iteration threshold = 5

5.3 | Performance metrics

We believe that by placing the application modules will improve application response time, deadline-aware and energy-efficiency. Hence, our criteria to evaluate the proposed approach are deadline aware placement and minimizing the energy consumption. Our null hypothesis and alternate hypothesis are given below:

H0: There will be no difference in meeting the application deadline and energy consumption of the computation nodes after placing, using hyper heuristic algorithm.
H1: The proposed technique (DEEDSP) with placement using hyper-heuristic will have minimized energy consumption and satisfying deadlines of the application.

5.4 | Evaluation scenarios

The aim of this assessment is to analyze in contrast with different current methods how successful framework module placement plans are. Those approaches are cloud only placement, collaborative task offloading scheme with service orchestration (CTOSO),[39] edge-ward placement (EWP),[24] genetic scenario (GS),[29] deadline oriented service placement (DoSP),[31] genetic algorithm,[16] particle swarm optimization,[17] and simulated annealing.[15] In applying for different service placement policies, we observe the response time, resource utilization, deadlines, and energy use. All application modules are placed on a cloud node in the cloud only policy. This policy illustrates the advantages of fog cluster decentralization. Collaborative task offloading scheme with service orchestration policy, gives the higher priorities to the shorter delay modules requirements. The application modules are placed closed to the edge fog nodes and if it is not possible then it placed in cloud. In the edge-ward algorithm, modules will be positioned at the network edge and will move from the fog nodes toward the cloud. The genetic scenario is based on the genetic algorithm investigate a large search space for providing the qualitative solution for QoS placement. The DoSP policy is a modified genetic algorithm, this algorithm places the sensing and actuation modules in the FN, high prioritized processing modules in FCN, and the remaining processing modules in NFCN or CN.

5.5 | Analysis of results

This section analysis the results of time frame-conscious application placement and reduction of the energy consumption as the functions with different application complexities (ie, latency sensitive, latency tolerable applications with various computational capacities). They have been thoroughly evaluated through iFogSim simulator on existing state-of-art approaches and proposed approach.

The results are divided into two scenarios. Scenario 1 has applications with low computation. Scenario 2 has applications with high computation. The computations are differentiated by comparing them with FN computations.

5.6 | Response time

5.6.1 | Scenario 1

Deadline satisfaction:

Each application's response time, Res_{time} is determined by the Equation (5). The execution time ($Exec_{\text{time}}$) and the total time of deployment ($Total_{\text{Dept}}$)is computed with (1) and (2) with the respective data given in Tables 3–5. After computations, the corresponding deadline $D(n)$ of the application in Table 4 shall be compared to Res_{time} with various cloud distances. The results for the first scenario are shown in Figures 6–8.

The cloud-only policy exceeds the application deadline for motion, video, sound, temp, and humidity applications as shown in Figures 6,7, and 8. It exceeds deadline with an average cloud distance of 25 but in case of motion application it exceeds at 10, as it has short deadline.

The CTOSO policy exceeds the deadline in video and temp applications at the cloud distance of 50 as shown in Figure 6B and 7B, respectively. This policy does not exceed the deadline for short deadline application even at high cloud distances, because it places the application in FN. The long deadline applications will be given priority to place in CN, hence the applications exceed the deadline. At a cloud distance 50, two latency tolerable applications (ie, video and temp) exceed the deadline. The EWP policy exceeds the deadline for the motion application at the distance 20. The previous applications may have exhausted the resources, FN and FCN, hence the forth coming applications are left with resources, NFCN and CN. When the TAU and CN distance is high there may be a chance to exceed the deadline. The GS, DoSP, GA, PSO, SA, and DEEDSP policies does not exceed the deadlines in any scenario. The methods cloud Only, CTOSO, and EWP exceeds the application deadline. CTOSO exceeds the deadline for latency tolerable applications at high cloud distances and for latency sensitive applications with high computation. EWP policy exceeds the deadline for latency sensitive applications with low and high computation. For short deadline applications, DoSP gives better response time whereas for high cloud distances DEEDSP gives better response times. GA, PSO, SA, and DEEDSP policies produced low "combined response time" than GS and DoSP. Applications like video and temp whose deadlines are far enough, EWP has proven to be the best, while at a point, 50, it is observed that the proposed method outperforms EWP. In low and high

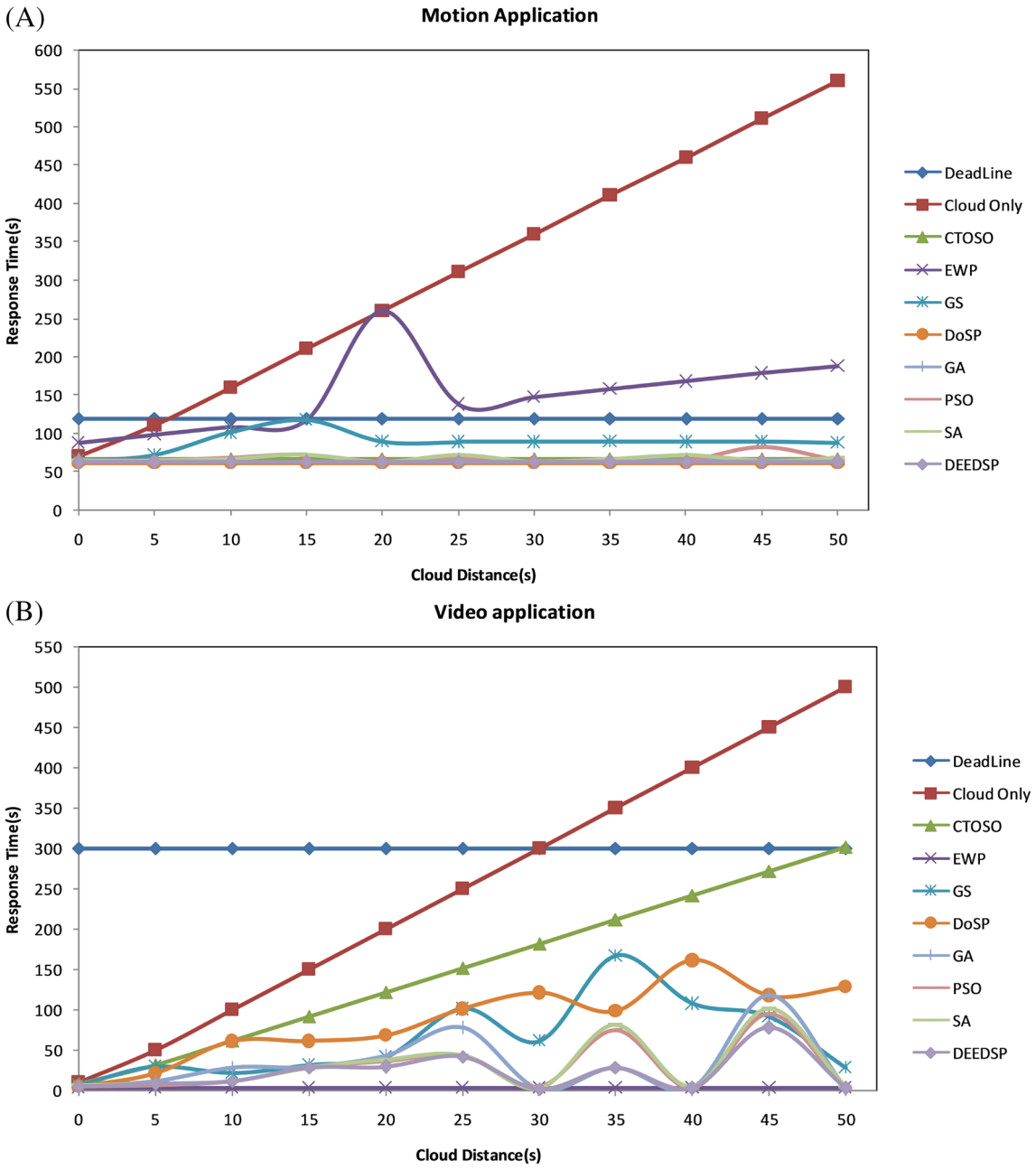


FIGURE 6 Performance comparison of motion and video applications

computational tasks DoSP provides least response times for latency sensitive applications. Comparatively in high computational modules the DEEDSP provides equal response times for latency sensitive applications.

In Figure 8, it is observed that GS and DoSP policies placed the application such that they are close to its deadline. Figure 9 displays the response times of motion application, for TAU = 25, cloud distance = 20, and varying FN MIPS. In this situation, the cloud only and EWP policies exceed the deadline at 200 MIPS. The proposed method produced low "combined response times" than all other policies.

5.6.2 | Scenario 2

The results for the second scenario are presented in the Figures 10–13. In this scenario at the cloud distance of 10, all the policies that were used for comparison do not exceed the deadline in all the applications except in Motion

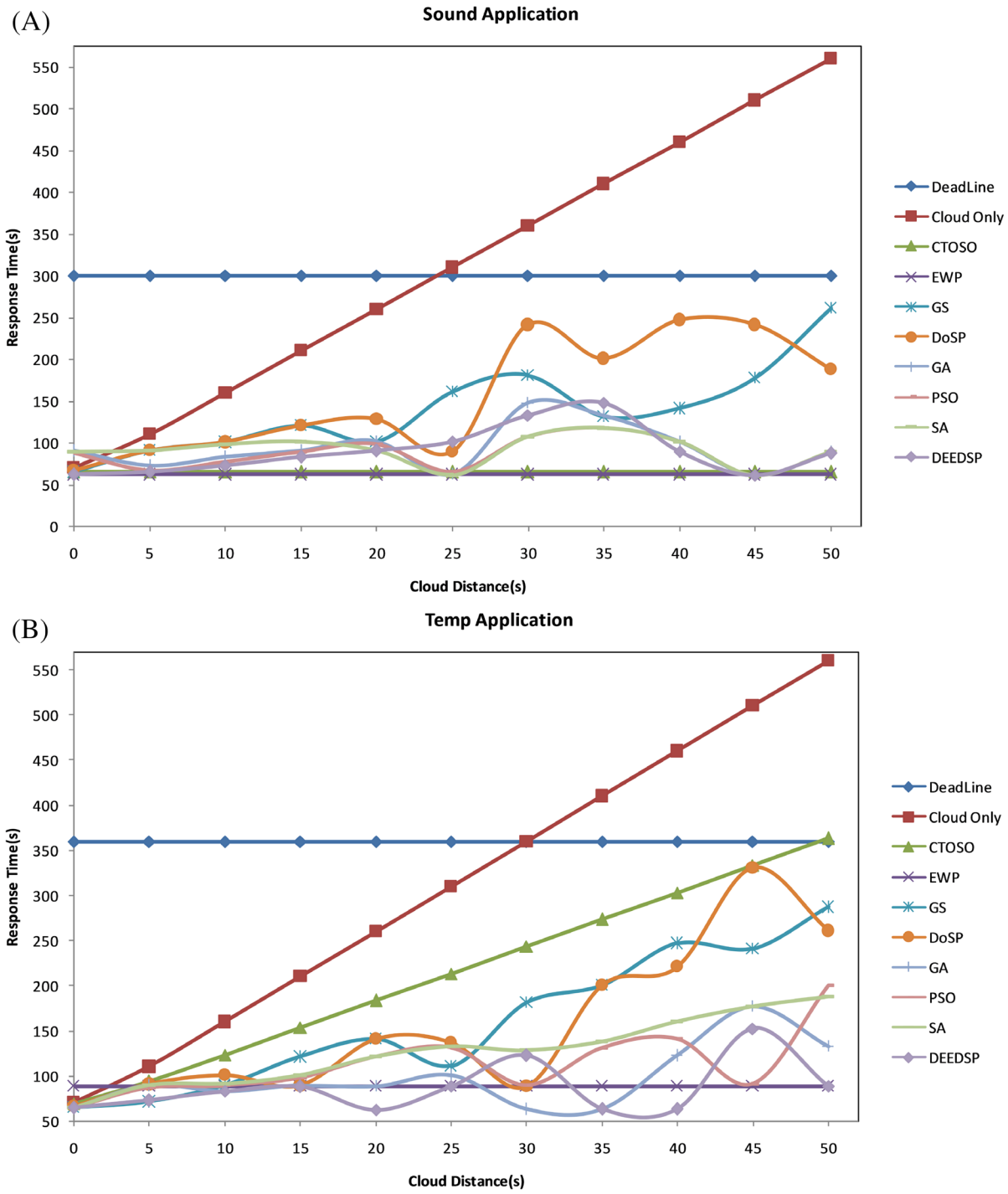


FIGURE 7 Performance comparison of sound and temp applications

application. In this motion application cloud only, CTOSO, and EWP policies exceeds the deadline. CTOSO exceeds the deadline as there is no possibility to place the high computational modules in FN. The proposed DEEDSP algorithm places this application in FCN, so it is observed from the results that DEEDSP performs equally well as the DoSP.

Figure 13 shows the response times of motion application, for TAU = 25, cloud distance = 20, and varying FN MIPS. In this situation, the cloud only, EWP, and CTOSO policies exceed the deadline up to the MIPS reaches 250. Until MIPS 250, DEEDSP scheme, and DoSP performed equally well, for the higher MIPS values DoSP performs slightly better than the proposed DEEDSP algorithm.

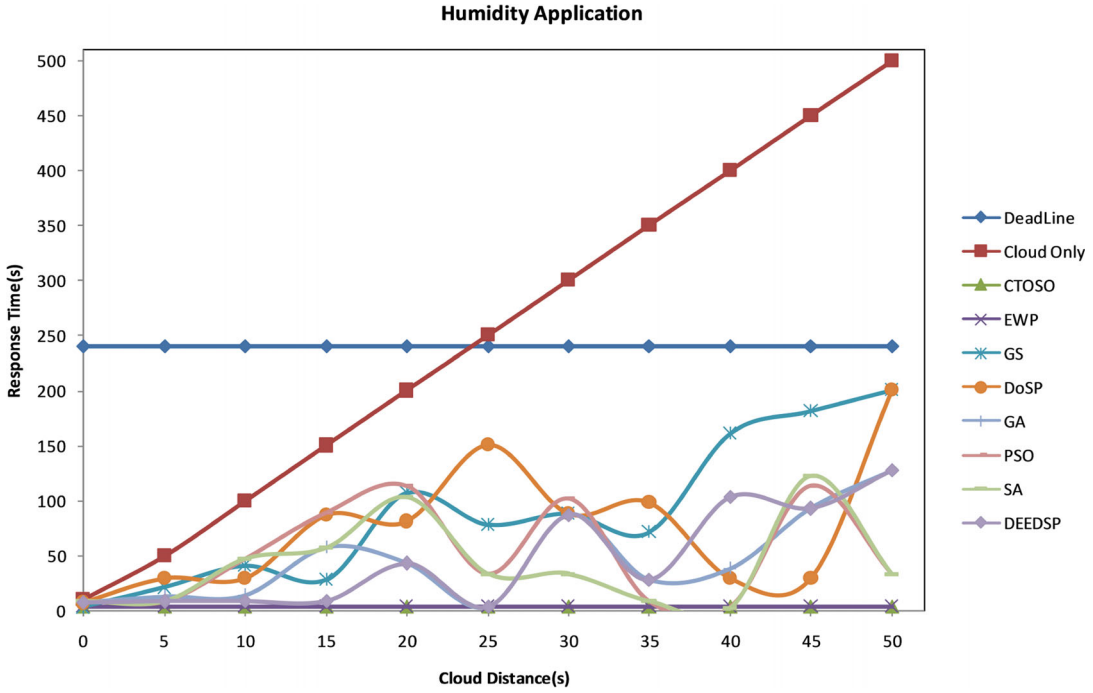


FIGURE 8 Performance comparison of humidity application

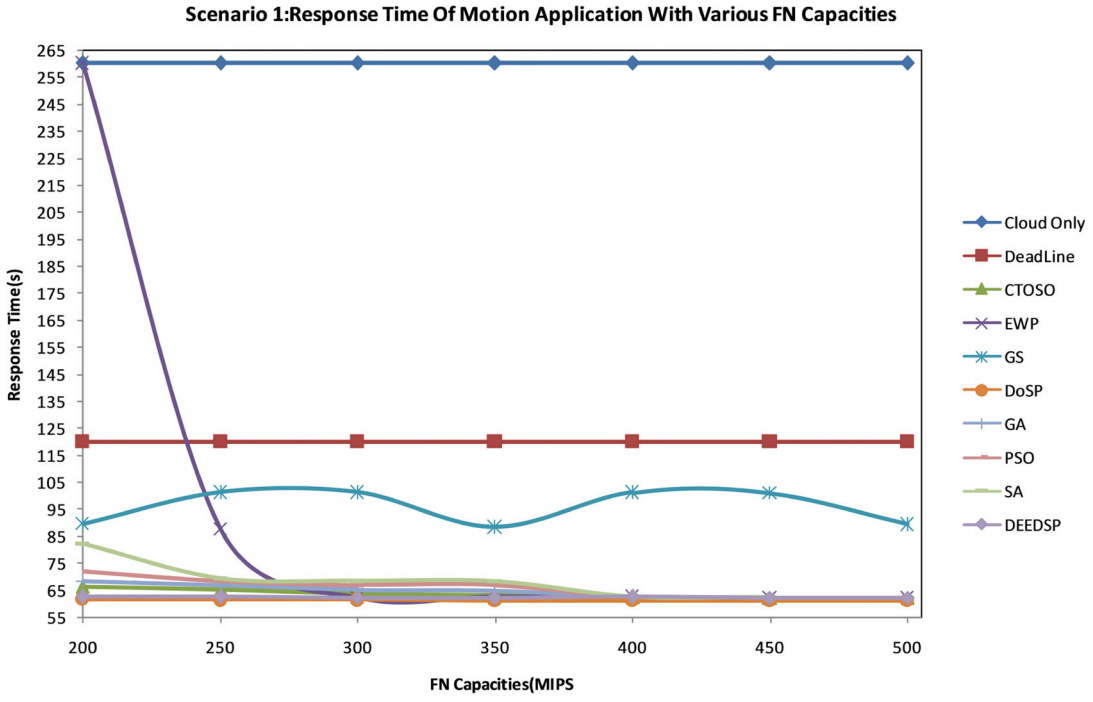


FIGURE 9 Response time of motion application with various FN capacities

5.7 | Resource utilization

The cloud utilization for Cloud Only, CTOSO, EWP, GS, DoSP, GA, PSO, SA, and DEEDSP policies is 100%, 24%, 0%, 36%, 32%, 4%, 8%, 8%, and 0%, respectively is shown in Figure 14. The FN utilization of Cloud Only, CTOSO, EWP, GS, DoSP, GA, PSO, SA, and DEEDSP policies is 0%, 76%, 76%, 40%, 40%, 68%, 68%, 68%, and 68%, respectively. The FCN utilization of Cloud Only, CTOSO, EWP, GS, DoSP, GA, PSO, SA, and DEEDSP policies is 0%, 0%, 12%, 20%, 16%, 16%, 16%, 16%,

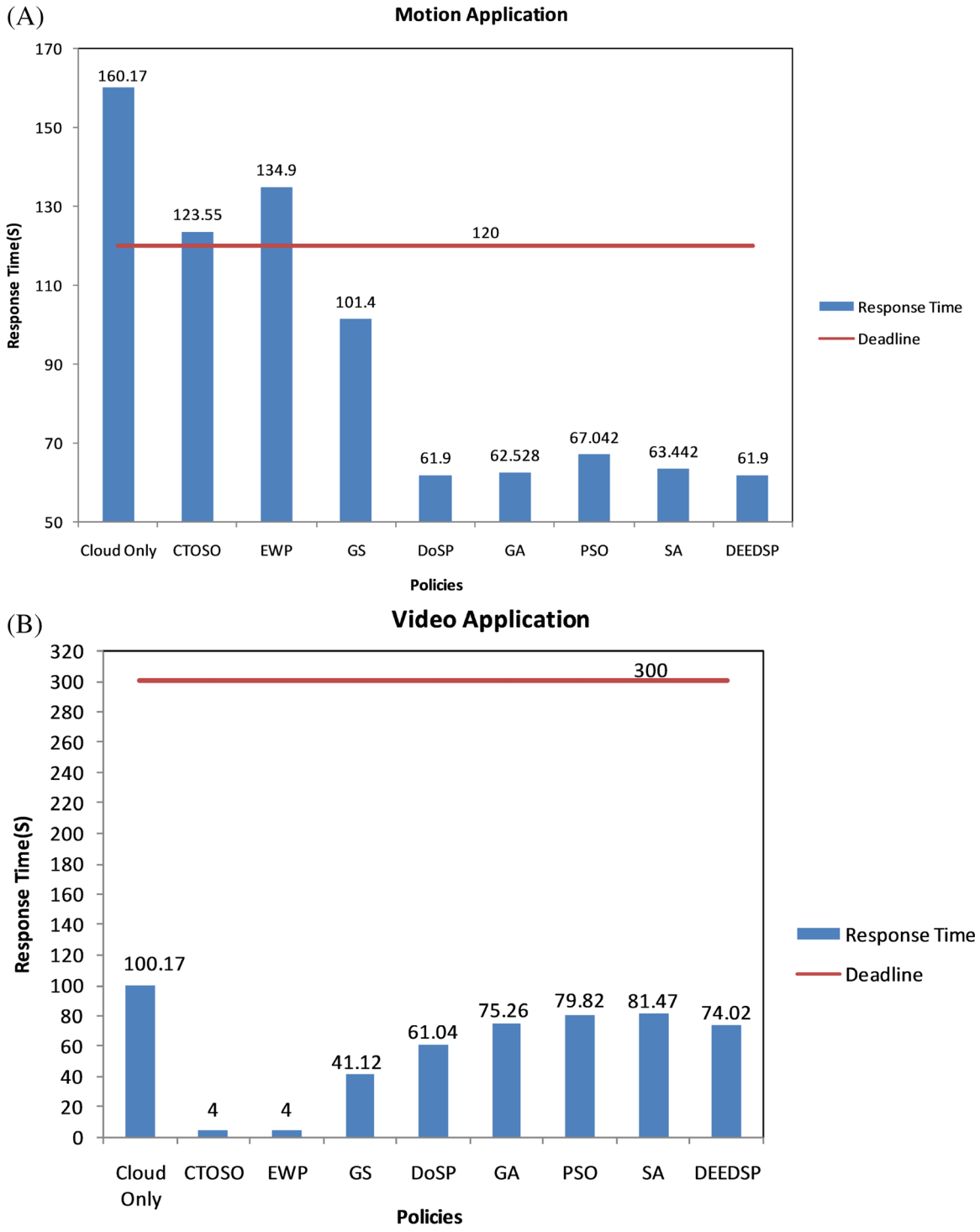


FIGURE 10 Performance comparison of motion and video applications

and 16%, respectively. Finally, the NFCN utilization for cloud only, CTOSO, EWP, GS, DoSP, GA, PSO, SA, and DEEDSP policies is 0%, 0%, 12%, 4%, 12%, 12%, 8%, 8%, and 16%, respectively. In both GS and DoSP policies, the fog nodes are allowed to run only the sensing and actuation modules of an application. If we observe all the policies, the EWP and DEEDSP policy has placed no application modules in the cloud node. The GS policy has placed more modules in cloud node except in Cloud Only policy. CTOSO, EWP, GA, PSO, SA, and DEEDSP policies placed more or almost same number of modules in FN nodes than GS, DoSP and Cloud Only.

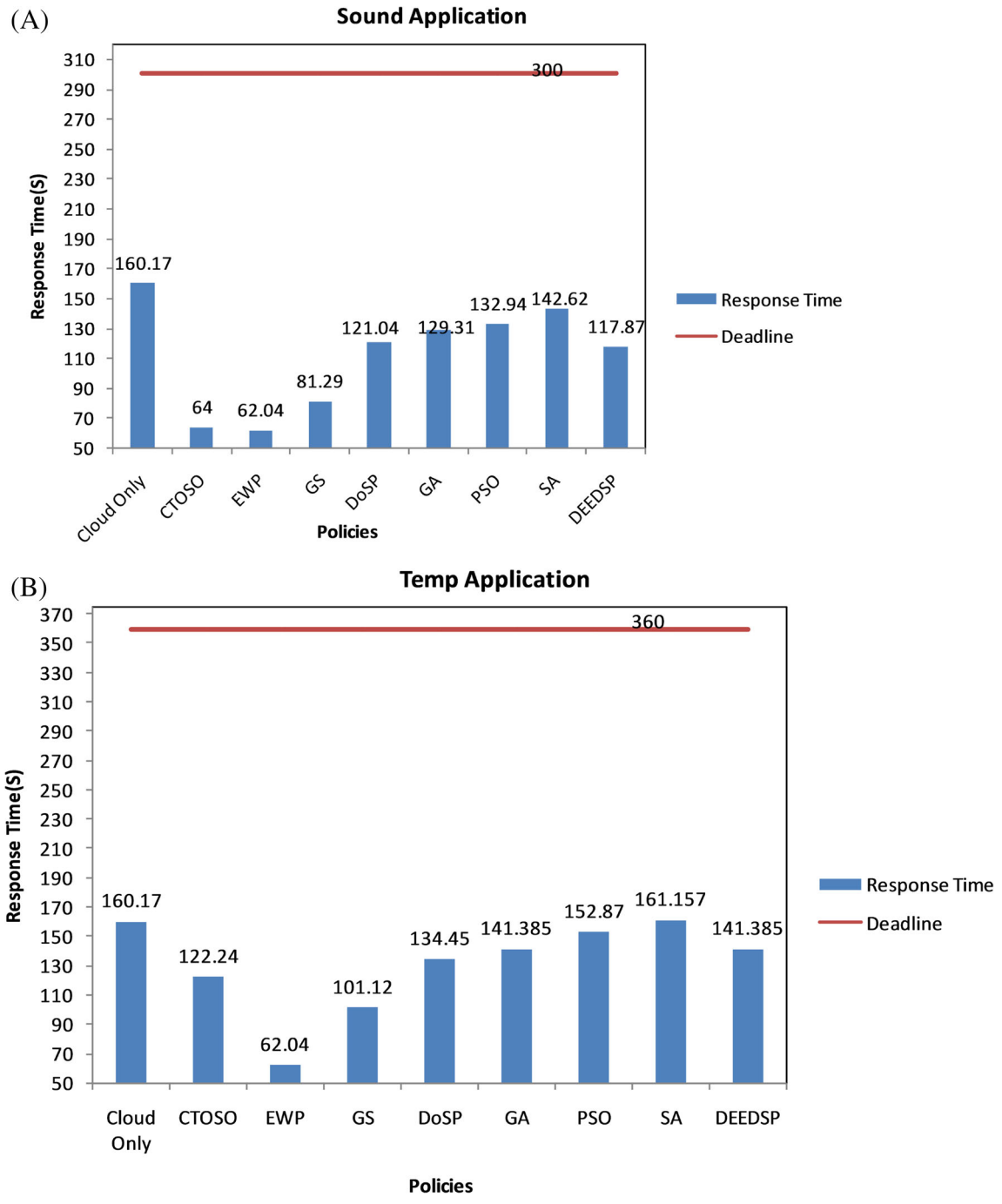


FIGURE 11 Performance comparison of sound and temp applications

Figure 15 shows the share of each resource type, while varying the cloud distance. As the cloud distance increase the module placement in the cloud node gets gradually decreased. To avoid applications to exceed its deadline the applications are placed in fog node, this will increase the fog node utilization. In a deadline-aware environment, with the increase in the TAU value and respectively the cloud distances, the modules are preferred to get placed mostly in FCN rather than in NFCN.

With the increase in TAU value the module placement in the NFCN is gradually decreased because of the application deadline, this can be observed in Figure 16. The module placement increases either in cloud node, FCN, or FN node. If we increase the cloud distance along with the TAU values the modules is preferred to run in FN node or FCN node. The applications which are far from the deadline are preferred to place either in NFCN or CN.

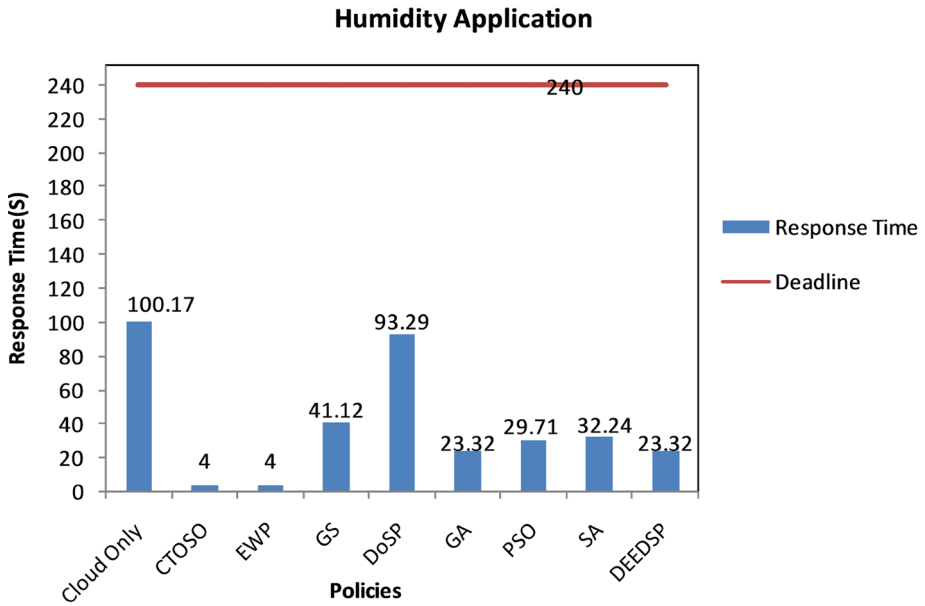


FIGURE 12 Performance comparison of humidity application

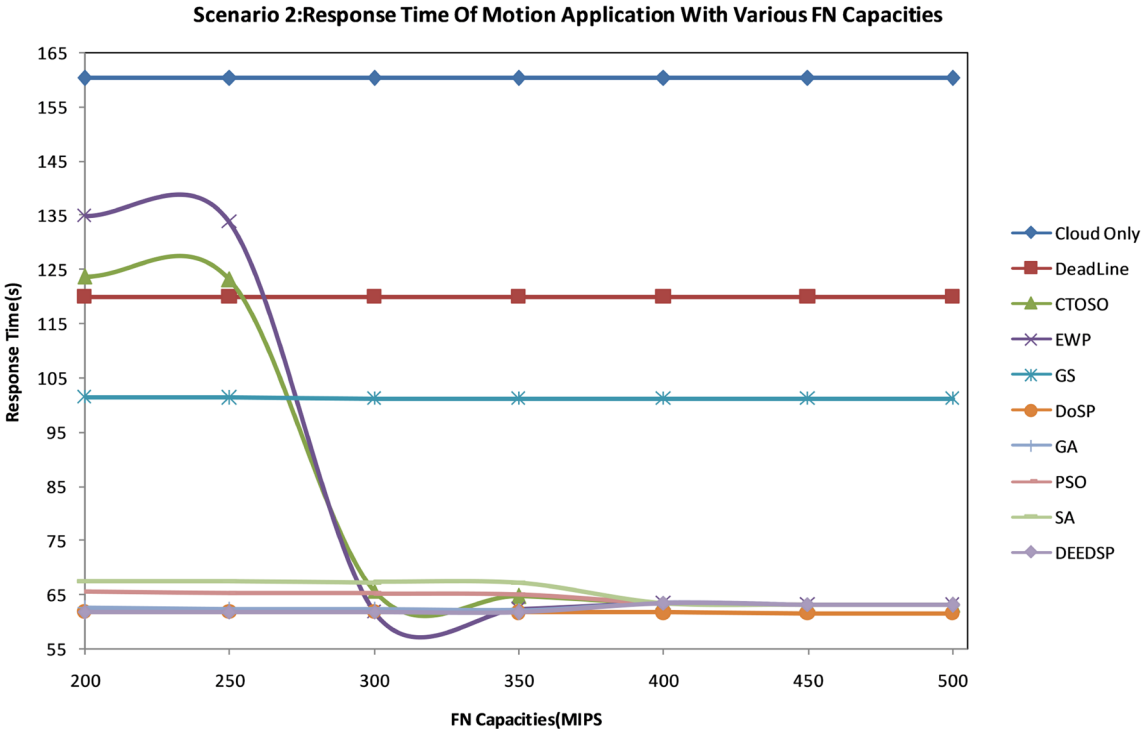


FIGURE 13 Response time of motion application with various FN capacities

5.7.1 | Scenario 1

Figure 17 shows the FN utilization for varying FN's MIPS values. The applications are assumed to have low computational cost and short deadline. Observations can be made that the Cloud Only policy does not place the application modules in the FN node. GA, PSO, SA, CTOSO, and EWP policies places the equal number of modules in the FN node. GS and DoSP policies utilize 40% of the fog node resources for the all FN node MIPS. These two policies do not allow placing the application's process modules. In the GA, PSO, SA, and DEEDSP schemes, until 350 MIPS, FN node utilization is lesser

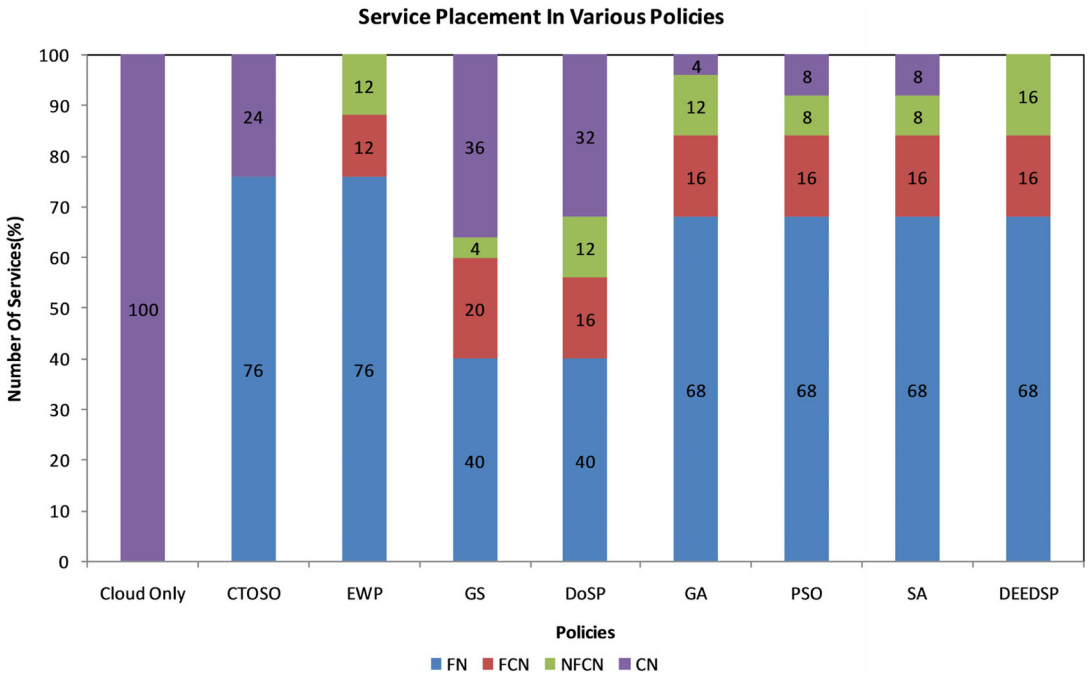


FIGURE 14 Service placement in various policies

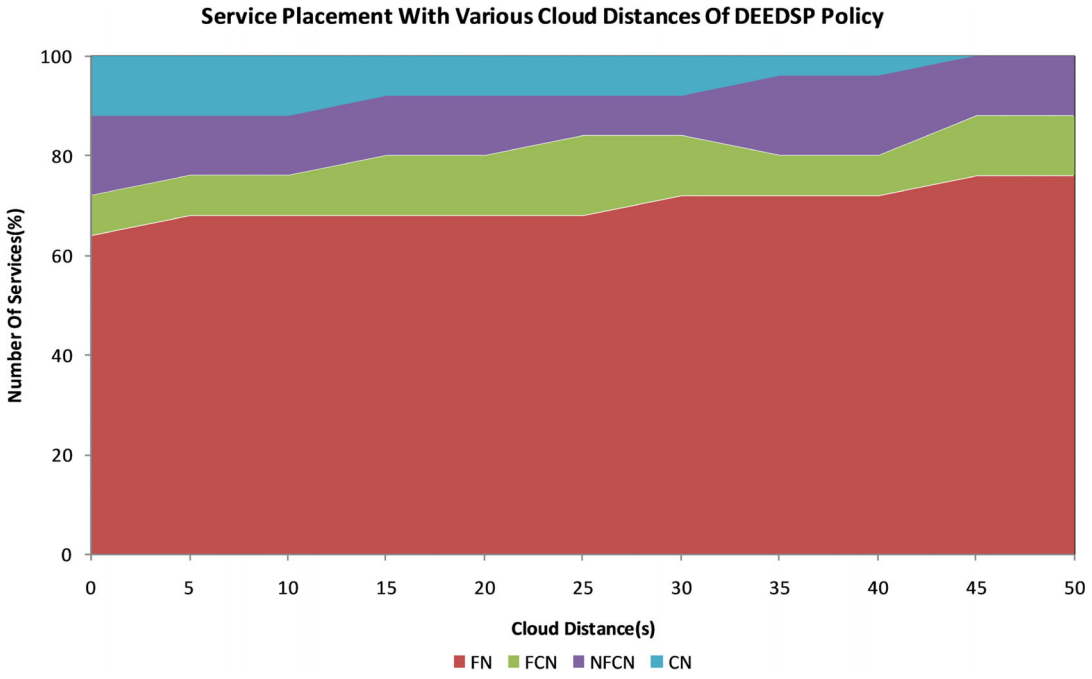


FIGURE 15 Service placement with various cloud distances of DEEDSP policy

than CTOSO, EWP, GA, PSO, and SA policies. After 400 MIPS CTOSO, EWP, and DEEDSP policies place all the modules in FN node.

5.7.2 | Scenario 2

Figure 18 shows the FN utilization with respect to various FN MIPS values. The applications are assumed to have high computational cost and short deadline. Here in the case of cloud only policy, it does not place the application modules

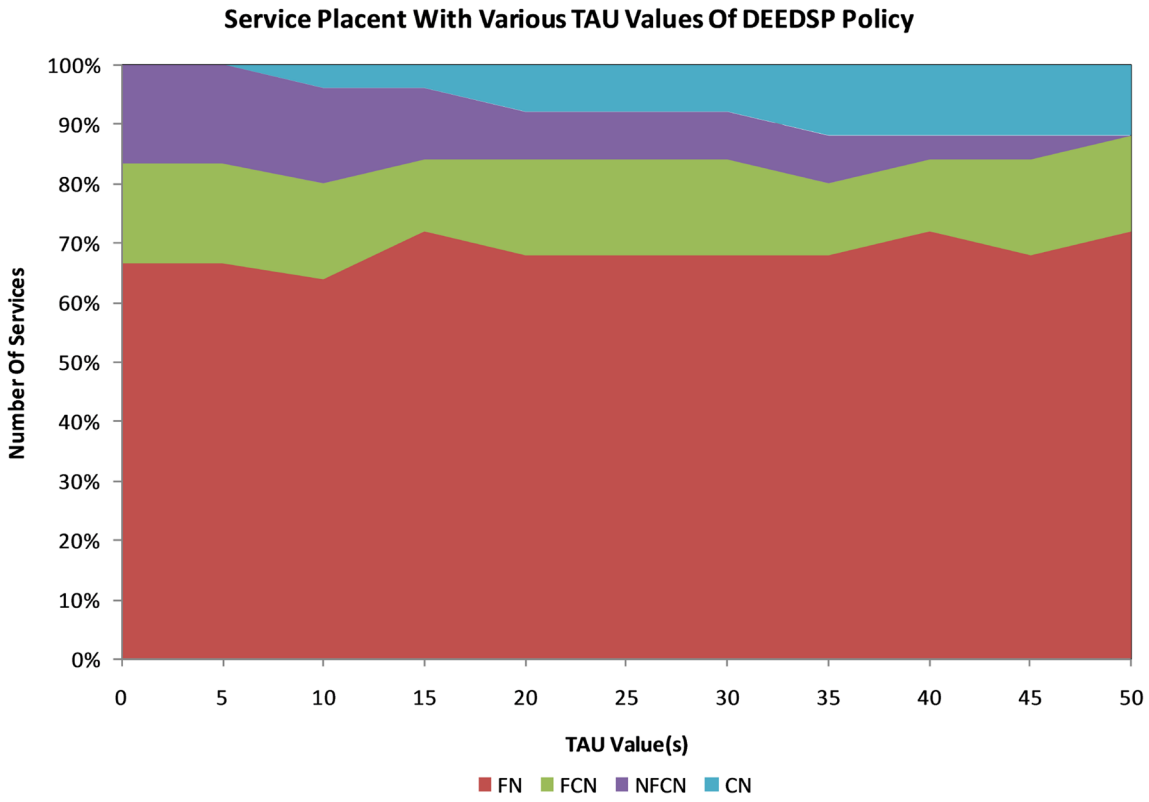


FIGURE 16 Service placement with various TAU values of DEEDSP policy

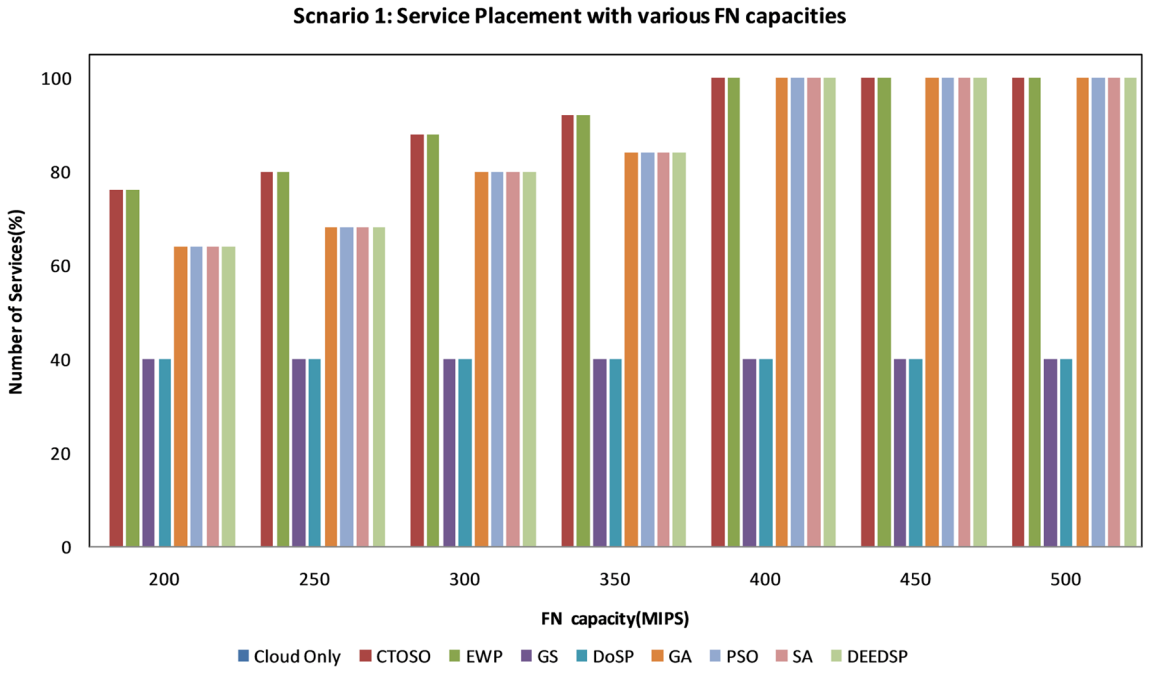


FIGURE 17 FN utilization with various capacities

in the FN node. Same as in the previous case, CTOSO and EWP policies places the equal number of modules in the FN node. Coming to GS and DoSP policies, they utilize 40% of the fog node resources, here only the sensing and actuation modules of the applications are allowed to run in FN. These two policies do not place the application process modules in FN. In the GA, PSO, SA, and DEEDSP policies, FN node utilization is lesser than CTOSO, EWP, GA, PSO, SA, and

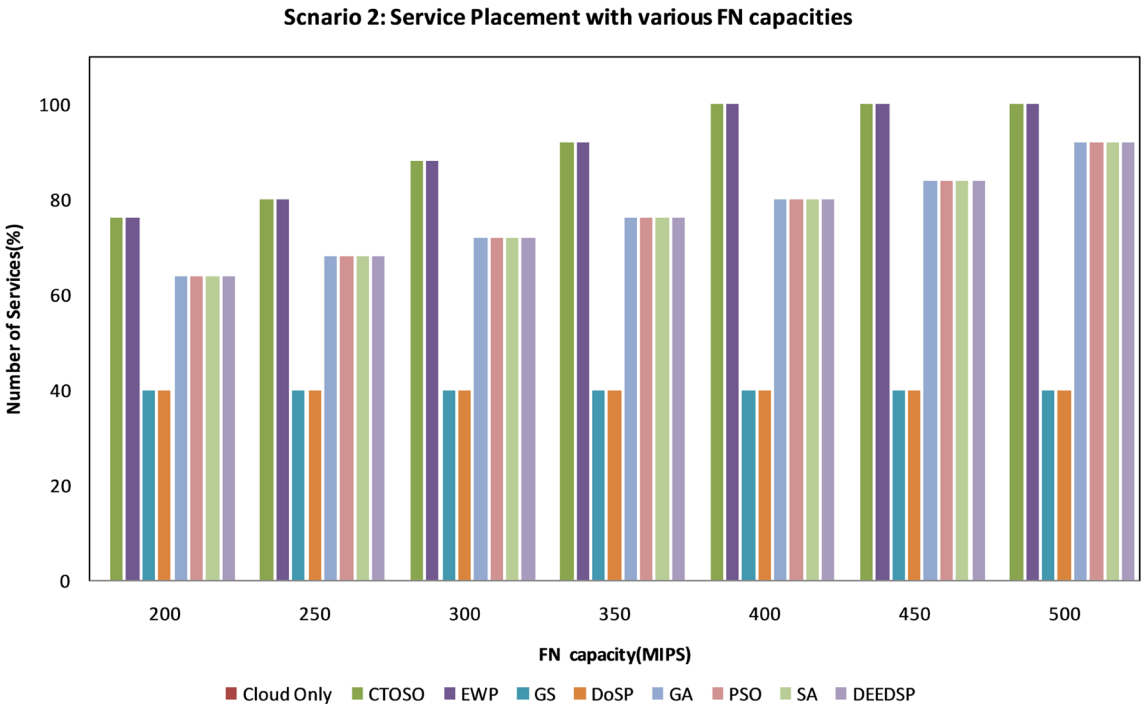


FIGURE 18 FN utilization with various capacities

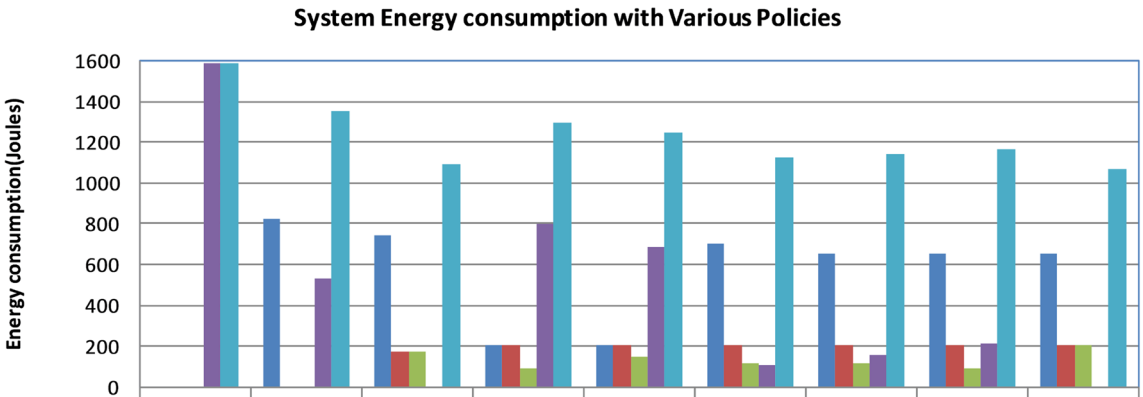
DEEDSP policies. Here in the same policy FN nodes are occupied by sensing and actuation modules which block the FN nodes from running other modules. After 400 MIPS CTOSO, and EWP policies place all the modules in FN node.

5.8 | Energy utilization

In Figure 19, The cloud-only policy consumes more energy than all other policies. It places all modules in cloud, so cloud takes more energy for executing the modules. The policies GA, PSO, SA consumes less energy than all the other policies except EWP, DEEDSP. The EWP and DEEDSP policy consumes lesser or equal cloud (CN) energy than all the other policies. DEEDSP policy consumes less energy than all the other policies. CTOSO policy places more modules in FN. CTOSO consumes more FN energy because it places more modules in FN. In CTOSO policy, it gives priority to place the modules in the fog node and the remaining modules in cloud (CN). EWP and DEEDSP policies mostly does not place the modules in the cloud. EWP consumes more energy than DEEDSP because of high FN energies. All the policies that consider FCN for placement except EWP, consumes equal FCN energies. The NFCN energy is high for DEEDSP than all other policies which consider NFCN. This policy succeeds other policies in utilizing the NFCN node without exceeding the application deadline.

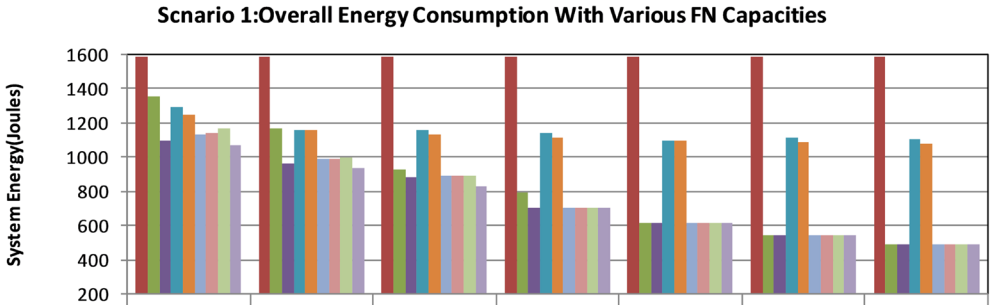
5.8.1 | Scenario 1

Figure 20 shows the energy consumption for varying fog node MIPS. The cloud energy consumption is constant in the case of cloud only policy for all the FN MIPS values. Upto 350 FN MIPS value, DEEDSP consumes less energy than all other policies. Up to 250 FN MIPS, the CTOSO consumes more energy than DEEDSP policy because it places more modules in the cloud node. From 400 FN MIPS, the CTOSO, EWP, GA, PSO, SA, and DEEDSP policies consume same amount of energy, as there is enough space to accommodate all the modules alone in FN. The GS and DoSP policies are not affected by the increase of FN MIPS value. These two policies place the sensing and actuation modules only in FN. It utilizes the FN resources equally even the FN MIPS value increases. The proposed DEEDSP consumes lesser or sometimes equal energy than other policies for to varied FN capacities.



	Cloud Only	CTOSO	EWP	GS	DoSP	GA	PSO	SA	DEEDSP
FN	0	820	738	205	205	697	656	656	656
FCN	0	0	176.571	206	206	206	206	206	206
NFCN	0	0	176.571	88.285714	147.14285	117.714	117.714	88.285	206
CN	1589.1428	529.71428	0	794.57142	688.62857	105.942	158.914	211.885	0
Sys_Energy	1589.1428	1349.7142	1091.142	1293.8571	1246.7714	1126.657	1138.628	1162.171	1068

FIGURE 19 System energy consumption of various policies

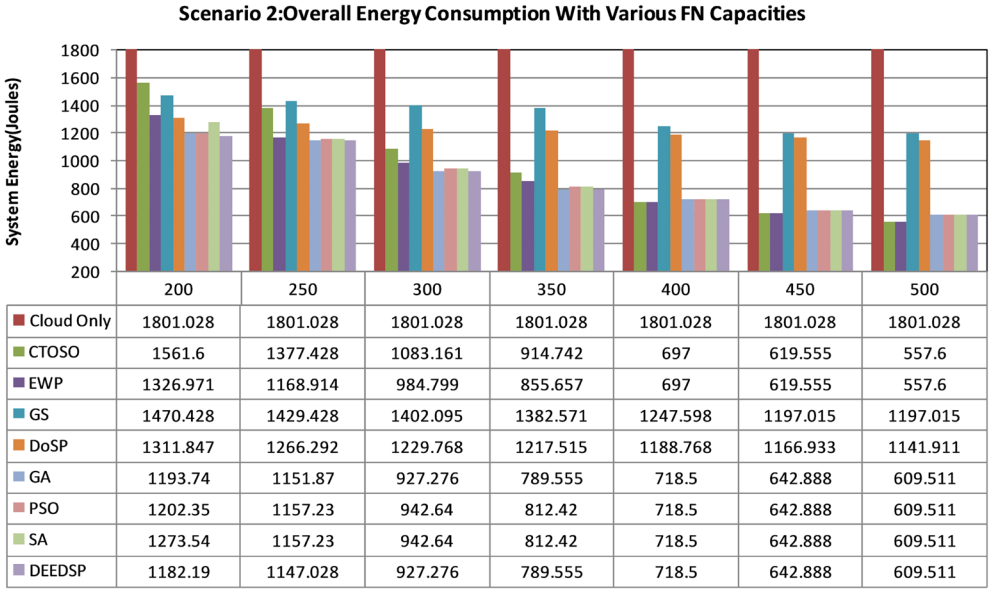


	200	250	300	350	400	450	500
Cloud Only	1589.142	1589.142	1589.142	1589.142	1589.142	1589.142	1589.142
CTOSO	1349.714	1165.542	922.552	789.5556	615	546.66	492
EWP	1091.142	957.028	883.847	702.857	615	546.66	492
GS	1293.857	1158.685	1154.895	1135.371	1097.185	1109.33	1100.22
DoSP	1246.771	1158.685	1131.352	1111.828	1097.185	1085.79	1076.68
GA	1126.657	984.21	892.34	702.857	615	546.66	492
PSO	1138.628	992.173	892.34	702.857	615	546.66	492
SA	1162.171	997.326	892.34	702.857	615	546.66	492
DEEDSP	1068	935.142	824.19	702.857	615	546.66	492

FIGURE 20 Overall energy consumption with various FN capacities

5.8.2 | Scenario 2

The cloud only policy, as shown in Figure 21, consumes constant amount of energy for various FN MIPS as all the modules are placed in CN. Until 350 FN MIPS of the fog node, DEEDSP consumes less or equal energy than all the other policies. DEEDSP policy has organized to place all the high computation modules in FN. From 400 FN MIPS CTOSO and EWP policies are proven to be the best in system energy consumption. GS and DoSP policies are not affected by the FN MIPS value; these two policies place sensing and actuation modules only in FN. It utilizes the FN resources consistently even the FN MIPS value increases. It is also observed that, starting at 400 MIPS GA, PSO, SA, and DEEDSP consumed equal FN energies.



	200	250	300	350	400	450	500
Cloud Only	1801.028	1801.028	1801.028	1801.028	1801.028	1801.028	1801.028
CTOSO	1561.6	1377.428	1083.161	914.742	697	619.555	557.6
EWP	1326.971	1168.914	984.799	855.657	697	619.555	557.6
GS	1470.428	1429.428	1402.095	1382.571	1247.598	1197.015	1197.015
DoSP	1311.847	1266.292	1229.768	1217.515	1188.768	1166.933	1141.911
GA	1193.74	1151.87	927.276	789.555	718.5	642.888	609.511
PSO	1202.35	1157.23	942.64	812.42	718.5	642.888	609.511
SA	1273.54	1157.23	942.64	812.42	718.5	642.888	609.511
DEEDSP	1182.19	1147.028	927.276	789.555	718.5	642.888	609.511

FIGURE 21 Overall energy consumption with various FN capacities

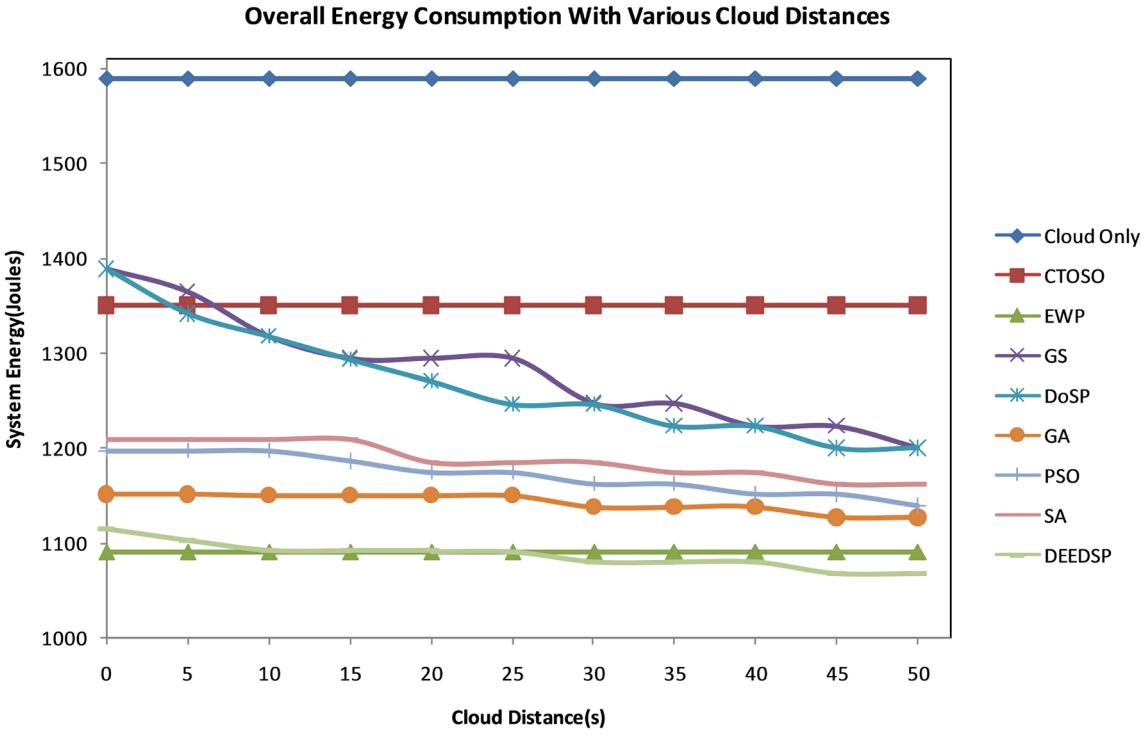


FIGURE 22 Overall energy consumption with various cloud distances

Figure 22 presents the energy consumption of all the policies with varied cloud distances. The energy consumption of cloud only, CTOSO and EWP policies does not get affect by the change in cloud distance. GS, DoSP, GA, PSO, SA, and DEEDSP polices energy consumption is decreased with increase in the cloud distance. When the cloud distance is increased these three policies avoid placing the modules in the cloud. Reducing the module placement in cloud avoids the application to exceed its deadline. The proposed method consumes less energy than all other policies at high cloud distances. GA, PSO, and SA policies consumes less energy than other policies except EWP and DEEDSP policies.

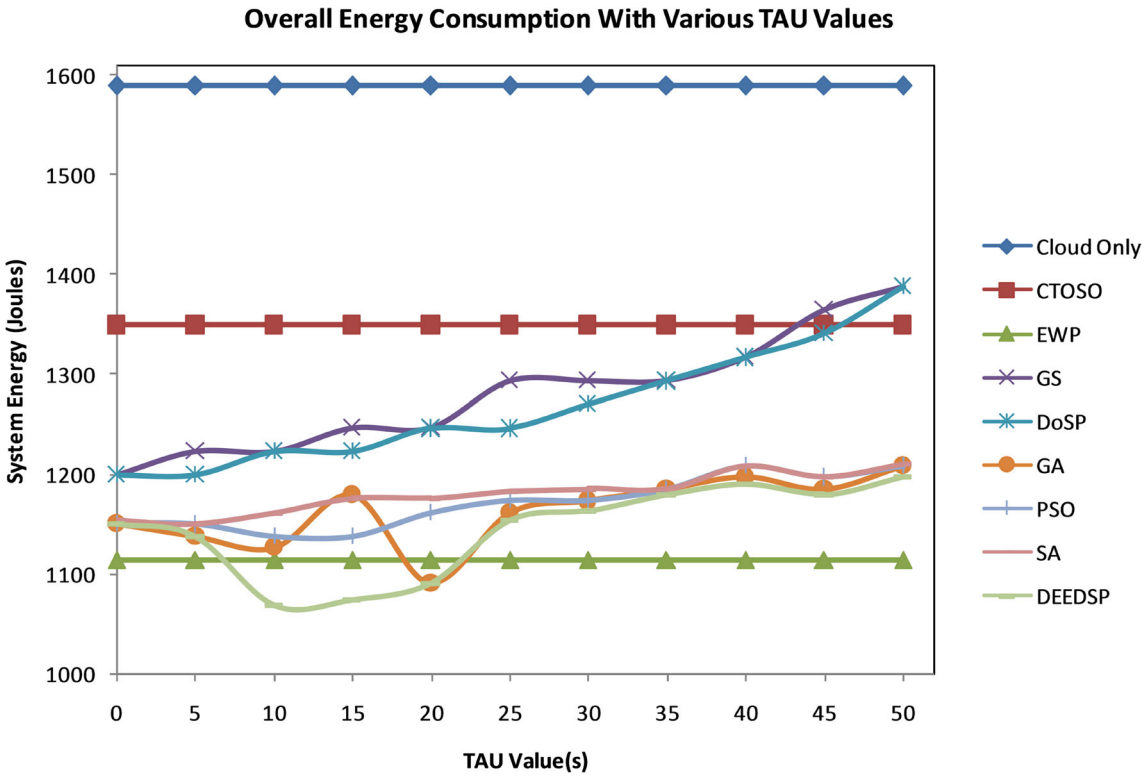


FIGURE 23 Overall energy consumption with various TAU values

Figure 23 shows the energy consumption of all policies with various TAU values. Between TAU values 5 and 20, the energy consumption of GA and DEEDSP is less than EWP policy. The energy consumption of cloud only, CTOSO and EWP policies does not change with TAU values. In cloud only policy all modules are placed in cloud node so it does not get impact by the TAU value, hence keeps the energy consumption constant. As in cloud only the scenario in CTOSO policy is same. Even after increasing the TAU value, the EWP policy places the modules in the neighbor controller node in a hierarchical order. So there is no change in the energy consumption with increased TAU values. In GS, DoSP, GA, PSO, SA, and DEEDSP policies, the total energy consumption of the system is increased with respect to the increase in the TAU value. When the TAU values increases, the GS and DoSP policies try to place the modules in FCN and CN. If more modules are placed in controller node and cloud node it automatically consumes more energy. With increase in TAU value, the DEEDSP policy energy consumption is lower than the GS, DoSP, GA, PSO, and SA. When the TAU value is increased the GS, DoSP, GA, PSO, SA, and DEEDSP policies minimize the module placement in neighbor fog controller node to avoid exceeding deadline.

5.8.3 | Trade-off between energy and execution time

Figure 24 plots the change in time and energy with a function of cloud distance (D) achieved by DEEDSP policy. In the case, D = 0, Energy consumption is maximized, however makespan time is very less at 20.64 seconds because many modules are assigned to CN. At a cloud distance D = 15, energy consumption and makespan time is improved by 23.543 joules and 23.74 seconds respectively. As D gradually changes to 50, the energy consumption decreases and the makespan time increases. Here in the DEEDSP policy Energy consumption is optimized and module placement in FN increases, so the makespan time increases. This experiment showed that, by adjusting the cloud distance D, our proposed work could be flexible. It satisfies user's requirements when they are interested in high performance execution or in energy consumption. At the cloud distance 30, the system balances with respect to energy consumption and makespan time. This point is an optimal point which balances energy consumption and makespan time.

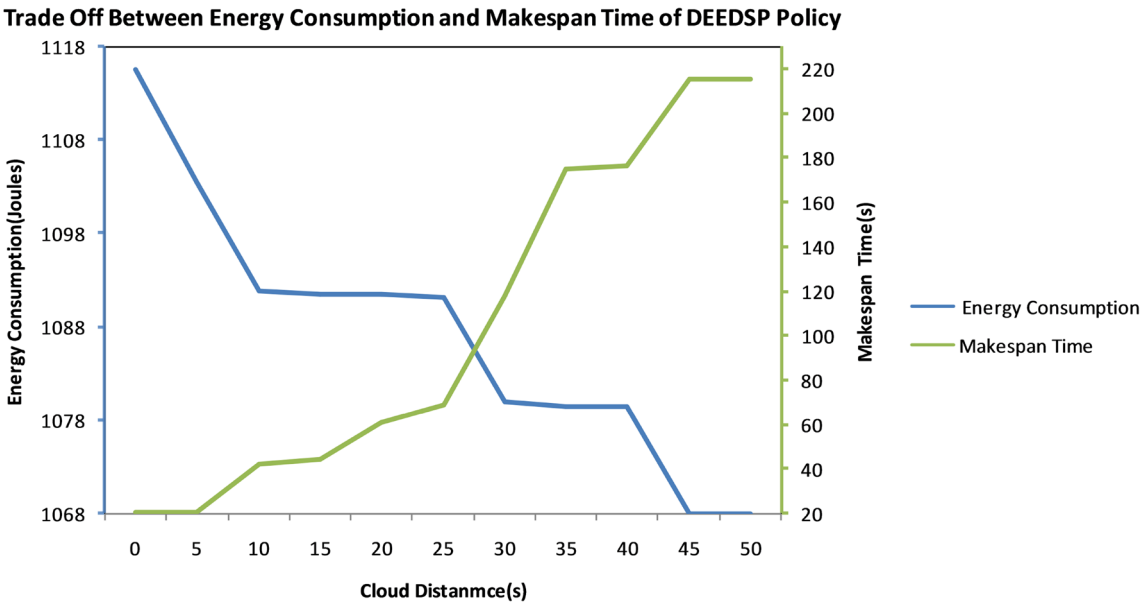


FIGURE 24 Trade-off between energy consumption and makespan time of DEEDSP policy

6 | CONCLUSIONS AND FUTURE WORK

This article suggests a multitiered application service fog computing paradigm and analyses its performance in IoT applications. There are numerous fog nodes with exploitative computational resources in the multitier fog computing model. A deadline-aware and energy-efficient service placement algorithm is proposed to improve the placement of the service over fog environment. To support the deadline-aware and energy-efficient service placement scheme, extensive benchmark experiments over fog environment were conducted to measure the response time of applications and energy consumption of the system over fog environment. By comparing with different state-of-the-art algorithms, the proposed deadline-aware and energy-efficient service placement algorithm is tested. The simulation results showed that the proposed algorithm works effectively over a multilevel fog environment. The possible future directions can be:

- DEEDSP can utilize latest artificial intelligence or machine learning models to optimize the QoS parameters.[12]
- DEEDSP can incorporate security by utilizing the concept of blockchain or quantum computing, which can further increase the computational power.[13]
- DEEDSP can be utilized for other IoT applications such as Industry 4.0, healthcare or agriculture.[12]
- DEEDSP can use the concept of serverless edge computing to scale the applications effectively.[13]

ACKNOWLEDGMENTS

The authors would like to thank Manmeet Singh (Fulbright-Kalam Fellow, The University of Texas at Austin) for his useful suggestions and discussion to improve the quality of the article. The authors would also like to thank the Editor in Chief (Prof. Changqiao Xu), associate editor and anonymous reviewers for their valuable comments and suggestions to help and improve our systematic review.

DATA AVAILABILITY STATEMENT

A significant amount of data is presented in this article. The remaining data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

ORCID

Sukhpal Singh Gill https://orcid.org/0000-0002-3913-0369

REFERENCES

1. Dadashi Gavaber M, Rajabzadeh A. BADEP: bandwidth and delay efficient application placement in fog-based IoT systems. *Trans Emerg Telecommun Technol*. 2021;32(8):1–16.e4136.
2. Benrazek AE, Kouahla Z, Farou B, Ferrag MA, Seridi H, Kurulay M. An efficient indexing for Internet of Things massive data based on cloud-fog computing. *Trans Emerging Telecommun Technol*. 2020;31(3):e3868.
3. Bashir H, Lee S, Kim KH. Resource allocation through logistic regression and multicriteria decision making method in IoT fog computing. *Trans Emerg Telecommun Technol*. 2019;1–14.e3824. https://doi.org/10.1002/ett.3824
4. Gill SS, Arya RC, Wander GS, Buyya R. Fog-based smart healthcare as a big data and cloud service for heart patients using IoT. Proceeings of the International Conference on Intelligent Data Communication Technologies and Internet of Things; August 7; 2018:1376-1383; Springer, Cham.
5. Gill SS, Tuli S, Xu M, et al. Transformative effects of IoT, blockchain and artificial intelligence on cloud computing: evolution, vision, trends and open challenges. *Internet of Things*. 2019;8:100118.
6. Singh J, Singh P, Gill SS. Fog computing: a taxonomy, systematic review, current trends and research challenges. *J Parallel Distrib Comput*. 2021;157:56–85. https://doi.org/10.1016/j.jpdc.2021.06.005
7. Rahman G, Wen CC. Fog computing, applications, security, and challenges, review. *Int J Eng Technol*. 2018;7(3):1615-1621.
8. Qi Q, Tao F. A smart manufacturing service system based on edge computing, fog computing, and cloud computing. *IEEE Access*. 2019;7:86769-86777. https://doi.org/10.1109/ACCESS.2019.2923610
9. Gill SS, Garraghan P, Buyya R. ROUTER: fog enabled cloud based intelligent resource management approach for smart home IoT devices. *J Syst Softw*. 2019;154:125-138.
10. Ye X, Xavier E, Loïc L, Thierry C, Frédéric D. Combining hardware nodes and software components ordering-based heuristics applications in the fog. Paper presented at: Proceedings of the 33rd Annual ACM Symposium on Applied Computing; 2018:751-760; ACM, New York, NY.
11. Brogi A, Forti S, Guerrero C, Lera I. How to place your apps in the fog: state of the art and open challenges. *Softw Pract Exper*. 2020;50:719-740. https://doi.org/10.1002/spe.2766
12. Teoh YK, Gill SS, Parlikad AK. IoT and fog computing based predictive maintenance model for effective asset management in industry 4.0 using machine learning. *IEEE Internet Things J*. 2021;1–8. https://doi.org/10.1109/JIOT.2021.3050441
13. Gill SS. Quantum and blockchain based serverless edge computing: a vision, model, new trends and future directions. *Internet Technol Lett*. 2021;1–6. https://doi.org/10.1002/itl2.275
14. Natesha BV, Guddeti RMR. Adopting elitism-based genetic algorithm for minimizing multi-objective problems of IoT service placement in fog computing environment. *J Netw Comput Appl*. 2021;102972.
15. Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. *Science*. 1983;220(4598):671-680.
16. Holland JH. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Cambridge, MA: University of Michigan Press; 1975.
17. Kennedy J, Eberhart RC. Particle swarm optimization. Paper presented at: Proceedings of the IEEE International Conference on Neural Networks, Perth, WA, Australia; 1995:1942-1948; IEEE.
18. Zhang G, Zhang W, Yu C, Li D, Lin W. Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices. *IEEE Trans Ind Inform*. 2018;14(10):4642-4655.
19. Zhou Z, Wang Z, Yu H, et al. Learning-based URLLC-aware task offloading for internet of health things. *IEEE J Select Areas Commun*. 2020;39(2):396–410.
20. Hosseini Bidi A, Movahedi Z, Movahedi Z. A fog-based fault-tolerant and QoE-aware service composition in smart cities. *Trans Emerging Tel Tech*. 2021;1–20.e4326. https://doi.org/10.1002/ett.4326
21. Li X, Qin Y, Zhou H, Zhang Z. An intelligent collaborative inference approach of service partitioning and task offloading for deep learning based service in mobile edge computing networks. *Trans Emerg Tel Tech*. 2021;32(9):1–19.e4263. https://doi.org/10.1002/ett.4263
22. Taneja M, Davy A. Resource aware placement of IoT application modules in fog-cloud computing paradigm. Paper presented at: Proceedings of the 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, Portugal; 2017:1222-1228; IEEE.
23. Pham XQ, Huh EN. Towards task scheduling in a cloud- fog computing system. Paper presented at: Proceedings of the 2016 18th Asia-Pacific network operations and management symposium (APNOMS), Kanazawa, Japan; 2016:1-4; IEEE.
24. Gupta H, Vahid Dastjerdi A, Ghosh SK, Buyya R. iFogSim: a toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments. *Softw Pract Exper*. 2017;47(9):1275-1296.
25. Pham X-Q, Man ND, Tri NDT, Thai NQ, Huh E-N. A cost-and performance-effective approach for task scheduling based on collaboration between cloud and fog computing. *Int J Distrib Sens Netw*. 2017;13(11):1550147717742073.
26. Mahmud R, Ramamohanarao K, Buyya R. Latency-aware application module management for fog computing environments. *ACM Trans Internet Technol*. 2018;19(1):1-21.
27. Huang T, Lin W, Xiong C, Pan R, Huang J. An ant colony optimization-based multiobjective service replicas placement strategy for fog computing. *IEEE Trans Cybern*. 2020;1–14. https://doi.org/10.1109/TCYB.2020.2989309
28. Skarlat O, Nardelli M, Schulte S, Dustdar S. Towards QoS-aware fog service placement. Paper presented at: Proceedings of the 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC), Madrid, Spain; 2017:89-96; IEEE.
29. Skarlat O, Nardelli M, Schulte S, et al. Optimized IoT service placement in the fog. *SOCA*. 2017;11:427-443. https://doi.org/10.1007/s11761-017-0219-8

30. Choudhari T, Moh M, Moh TS. Prioritized task scheduling in fog computing. Paper presented at: Proceedings of the ACMSE; 2018:401-405; Richmond, Kentucky.
31. Sriraghavendra M, Chawla P, Wu H, Gill SS, Buyya R. DoSP: a deadline-aware dynamic service placement algorithm for workflow-oriented IoT applications in fog-cloud computing environments,in book title "energy coservation solutions for fog-edge computing paradigms. Lecture Notes on Data Engineering and Communications Technologies; 2021.
32. Mahmud R, Srirama SN, Ramamohanarao K, Buyya R. Profit-aware application placement for integrated fog–cloud computing environments. *J Parallel Distrib Comput*. 2020;135:177-190.
33. Naranjo PGV, Baccarelli E, Scarpiniti M. Design and energy efficient resource management of virtualized networked fog architectures for the real-time support of iot applications. *J Supercomput*. 2018;74(6):2470-2507.
34. Ramirez W, Masip-Bruin X, Marin-Tordera E, et al. Evaluating the benefts of combined and continuous fog-to-cloud architectures. *Comput Commun*. 2017;113:43-52.
35. Wu HY, Lee CR. Energy efficient scheduling for heterogeneous fog computing architectures. Paper presented at: Paper presented at: Proceedings of the 42nd IEEE International Conference on Computer Software & Applications, Tokyo, Japan; 2018.
36. Kim KH, Buyya R, Kim J. Power aware scheduling of bag-of-tasks applications with deadline constraints on DVS-enabled clusters. Paper presented at: Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid, CCGrid 2007; 2007; Rio de Janeiro, Brazil.
37. Deng R, Lu R, Lai C, Luan T. Towards power consumption delay tradeoff by workload allocation in cloud fog computing. Paper presented at: Proceedings of the 2015 IEEE International Conference on Communications (ICC); June 2015:3909-3914, London; IEEE.
38. Sharma S, Saini H. A novel four-tier architecture for delay aware scheduling and load balancing in fog environment. *Sustain Comput Inform Syst*. 2019;24:100355.
39. Huang M, Liu W, Wang T, Liu A, Zhang S. A cloud–MEC collaborative task offloading scheme with service orchestration. *IEEE Internet Things J*. 2020;7(7):5792-5805. https://doi.org/10.1109/JIOT.2019.2952767
40. Aslanpour MS, Gill SS, Toosi AN. Performance evaluation metrics for cloud, fog and edge computing: a review, taxonomy, benchmarks and standards for future research. *Internet of Things*. 2020;12:100273.
41. Sukhpal Singh and Inderveer Chana. Advance billing and metering architecture for infrastructure as a service. *Int J Cloud Comput Serv Sci*. 2013;2(2):123-133.
42. Qayyum T, Malik AW, Khan Khattak MA, Khalid O, Khan SU. FogNetSim++: a toolkit for modeling and simulation of distributed fog environment. *IEEE Access*. 2018;6:63570-63583. https://doi.org/10.1109/ACCESS.2018.2877696
43. Lopes MM, Higashino WA, Capretz MA, Bittencourt LF. MyiFogSim: a simulator for virtual machine migration in fog computing. Paper presented at: Proceedings of the10th International Conference on Utility and Cloud Computing; December 2017:47-52; Austin, Texas.
44. Lera I, Guerrero C, Juiz C. YAFS: a simulator for iot scenarios in fog computing. *IEEE Access*. 2019;7:91745-91758. https://doi.org/10.1109/ACCESS.2019.2927895

How to cite this article: Sri Raghavendra M, Chawla P, Singh Gill S. DEEDSP: Deadline-aware and energy-efficient dynamic service placement in integrated Internet of Things and fog computing environments. *Trans Emerging Tel Tech*. 2021;e4368. doi: 10.1002/ett.4368