PAPER • OPEN ACCESS

Offloading Decision Making for Workflow Applications Using an Enhanced Genetic Algorithm in the Edge Environment

To cite this article: S. Balasubramanian and T. Meyyappan 2022 *J. Phys.: Conf. Ser.* **2224** 012011

View the article online for updates and enhancements.

Offloading Decision Making for Workflow Applications Using an Enhanced Genetic Algorithm in the Edge Environment

S. Balasubramanian[1,*], T. Meyyappan[2]

[1]DDE, Alagappa University, Karaikudi, India

[2]Department of Computer Science, Alagappa University, Karaikudi, India

Email: ssbala44@gmail.com, meyyappant@alagappauniversity.ac.in

Abstract. Edge computing is an environment suitable for processing the workflow applications produced by the IoT devices to reduce time and energy consumption. The execution of the workflow task in the Cloud computing environment increases the consumption of both time and energy. In order to solve this issue, this paper proposes a new approach, namely, Decision making Regarding the offloading of A subset of the Workflow application (DRAW). In the DRAW approach, selecting the destination environment for executing the subset of the workflow application occurs in the Edge environment. The DRAW uses the genetic algorithm for offloading the subset to minimize the objective factors, including total execution time and energy consumption. It equally prioritizes both the objective factors for improving the execution of the subset in the Cloud environment using the improved genetic algorithm. The DRAW approach improves the genetic algorithm by removing its traditional limitations and produces an effective possible solution in terms of better offspring. Finally, the algorithm stops by attaining the best solution from the possible solutions. Thus, the implementation results show that the DRAW approach significantly outperforms the existing approach by minimizing execution time and energy consumption.

1. Introduction

The Cloud computing [1, 2] has been the ever-growing technology, that shares its resources such as storage, application, and computational capabilities on the pay-per-use basis, and its computation depends on the Internet connectivity. The Cloud computing technique processes several workflow tasks from the IoT devices that a resource-constrained device is incapable of providing several resources for processing these tasks. The execution of the IoT workflow application in the Cloud environment is possible through the computational offload technique. It reallocates the task from the resource-constrained environment to the resourceful environment, including Cloud, grid, and Edge. The Cloud infrastructure can process all the tasks of the IoT devices. However, the distance between these devices and the Cloud environment is large, that increases delay and minimizes the Quality of Service (QoS). For improving this constraint, continual research is held in the Cloud computing environment, that leads to the emergence of a new computation environment known as Edge computing. The edge computing [3, 4] addresses the problems in Cloud computation, including latency, traffic, and lack of mobility. It supports services and resources to the end user; that is, the extension of the cloud computing environment to the end device. Hence, most of the existing system transmits all its data from the IoT device to the Edge computing environment and decides the offloading of the task.

The IoT devices generate the workflow tasks [5, 6] as the DAG, in that the several vertices are connected using the directed edges. In this, the vertices are represented as the tasks, and the edges are represented as the interdependencies among the tasks. However, making the offloading decision for these workflow tasks is quite challenging, since this decision should consider the dependency between the tasks while offloading them. For processing these workflow tasks, it is necessary to have an effective offloading technique with the minimum delay and energy consumption. Hence, several existing systems propose several techniques for offloading decision making on the edge environment [7] to minimize the delay and cost, along with the improvement in energy efficiency. Although, most of the existing system struggle in an efficient offloading decision of the workflow task either to the Edge environment or to the Cloud environment. To cope up with this constraint, the DRAW approach uses the genetic algorithm for deciding the destination environment for executing the workflow tasks. It executes the dependent tasks of the workflow in the same environment for minimizing the time and energy consumption. The DRAW approach efficiently chooses the best solution from the possible solutions for offloading decision of all the workflow tasks.

The main contributions of the DRAW approach for offloading the workflow task in the Edge environment are as follows:

- ❖ The DRAW approach aims to propose an effective offloading decision regarding the selection of a suitable environment for executing the workflow application.
- ❖ The DRAW approach uses the Edge environment for making an offloading decision regarding the execution of a workflow task either in Edge or Cloud environment.
- ❖ It aims to minimize the objective factors, including the execution time and energy consumption while deciding on the task offloading for executing the workflow tasks.
- ❖ The DRAW approach proposes an optimized genetic algorithm for choosing the best solution from the possible solutions that satisfy the objective factors.
- ❖ The experimental results prove that the DRAW approach gives higher performance than the existing approach by an effective computational offloading decision for saving both time and energy consumption.

1.1. Paper Organization
The paper is organized as follows. In this paper, section 2 reviews the existing approaches related to the DRAW approach. Section 3 outlines the DRAW methodology and models the system of the DRAW approach. In section 4, the overall working procedure of the DRAW approach, and in section 5, various experiments are conducted for examining the performance of DRAW. At last, section 6 concludes the paper.

2. Related Works
This section provides the relative concept of the computational task offloading method. It also reviews the existing system for improving the offloading strategy for the workflow application.

2.1. Computational Task Offloading Based Mechanisms
Most of the existing systems try to improve the efficiency of the computational offloading process. Some of the existing systems fail to perform the computational offloading from the IoT device to the edge device or the cloud environment effectively. The heuristic offloading decision algorithm (HODA) in [8] improves the offloading decision, communication, and computation resources for minimizing the time and energy consumption. The HODA approach reduces the offloading decision problem with the sub-modular maximization problem and uses a greedy heuristic algorithm for solving this problem. Though, the HODA approach tends to offload the task only for the delay-tolerant applications. Reformulation-linearization-technique-based Branch-and-Bound (RLTBB) and Gini coefficient-based greedy heuristic (GCGH) in [9] solves the mixed integer nonlinear programming (MINLP) problem that is the minimization of energy consumption problem. This model reduces the energy consumption of the smart mobile device by optimizing offloading decision, and resource allocation of both radio resources and computational resources. However, considering the coordination between cloud and Mobile edge computing (MEC) is a challenging task. This paper [10] proposes the task offloading policy algorithm

that brings the fog computation closer to the local devices since this fog computation model connects with both the cloud nodes and local cloud nodes via wireless access infrastructure. Although this model reduces the cost of offloading the task to the fog computing environment, it lacks to offload the task dynamically and to perform virtual machine migration. The model in [11] proposes the framework for supporting the design model and the estimation model. It automatically selects the resources in the Cloud for offloading the task effectively. Although it reduces the execution time and the network delay, it is a static network model, that is, random and small-world networks. Computation Offloading and Resource Allocation algorithm (CORA) [12] jointly optimize the decision of offloading process, resource allocation, the bandwidth of the radio, and transmit power with the high tolerable delay. CORA improves the performance of this model via minimizing the system cost that is the energy consumption and the latency. Although it reduces the energy consumption, it lacks in managing the workload of the fog node. The Online Code Offloading and Scheduling (OCOS) algorithm in [13] offloads the independent task from the mobile device to the Cloud environment and also allocates the offloaded task to the resources. This model aims to reduce the overall completion time that is makespan of the global task. Though it minimizes the latency, it has some deficiency, including fault tolerance and detection; and lacks in efficient optimization of energy consumption, and latency. Similar to the previous research work this paper [14] also proposes the Multi-User Multi-Task Offloading (MUMTO) and MUMTO with CAP (MUMTO-C) algorithm for effective offloading decision making of the individual task along with the resource allocation of these offloaded tasks. Although this algorithm minimizes both the latency and energy consumption, it lacks in managing the task that arrives dynamically. A fast hybrid multi-site computation offloading solution in [15] provides the offloading decision to minimize the time consumption. This approach brings the solution for optimal or near-optimal offloading decision using the two algorithms, namely branch and bound (B&B) algorithm, and optimized Particle Swarm Optimization (PSO) algorithm. Although this approach fails to consider the execution of the parallel application that reduces the advantage of using multiple sites. An energy-efficient dynamic offloading and resource scheduling (eDors) policy in [16] reduces the completion time and energy consumption while executing an application with an effective offloading decision. Similarly, the model in [17] uses the Lyapunov optimization-based scheme for making the offloading decision by considering the current length of the queue. This model performs tasks scheduling efficiently along with the offloading decision making for minimizing the delay and energy consumption. However, this model considers that the data is transmitted easily through the available bandwidth for reducing time and energy consumption that is not possible for real-time applications. A Markov decision process approach in [18] decides the offloading environment for executing the task and allocates the resources to these tasks in the decided environment. This approach achieves minimized time consumption while executing the task using the optimal stochastic task scheduling policy. However, in this approach, feedback from the remote server is not provided, which is required for making the offloading decision and fails to discuss that generated signaling overhead. This paper [19] proposes the locally optimal algorithm derived from the univariate search technique to obtain the optimal computational speed and the transmit power of the SMD(Smart Mobile Device). Though it reduces both the energy consumption and latency excellently, the implementation of the process becomes complex because of the partial offloading and task breakdown. In this paper [20] the framework utilizes the contract theory for achieving the Nash equilibrium solution in the negotiation between the fog environment and the user, that completes the task with the minimum latency. Although this approach improves the utility gain of the user by improved 26%, it lacks to scale the resources for executing the task. Software Defined Task Offloading (SDTO) scheme [21] uses the task placement algorithm for optimally offloading the task from the mobile devices to reduce the delay and the energy consumption of the mobile devices. Although, the existing edge computing faces difficulties in issuing the requirements of the real-time applications since the edge server has limited storage and the computation capacity. This paper [22], performs a computational offloading of the task through the fog environment using the robotic sensor simulated in ROS/Gazebo. This model enhances the QoS and reduces communication latency and energy consumption. Although it manages the heterogeneous devices, it fails to extend the IoT networks to the large magnitude and this process is a costlier one. The greedy algorithm based on the partial critical path in [23] reduces the latency while executing the workflow task along with the minimization of energy consumption. However, this model assigns the entry and exit task in the mobile device; this increases energy consumption and makespan.

Currently, several existing techniques use the offloading technique to improve the processing of the application of IoT devices. These system offloads the entire application to the Edge environment from the IoT devices and decides the offloading of the task either to the Edge environment or to the Cloud environment. However, offloading of the dynamically arriving task from the IoT device, and providing the processing needs of the real-time intelligent application in many existing systems is still a challenging task. In consequence, this tends to increase the complexity in processing the overall technique. The existing system lacks to consider the start time of the task and the available bandwidth in the particular time. It results in the maximization of energy consumption and the delay by choosing the wrong destination environment.

3. An Overview of DRAW Methodology

Computational offloading is the technique that helps to execute the task in the Cloud environment, which is unfeasible in the Edge environment. Hence, the DRAW approach decides for the offloading of the workflow task using the optimized genetic algorithm to improve the profit for the cloud users. Furthermore, the DRAW model schedules the dynamically arriving tasks to the resources to maintain the load of the hosts in the Cloud or Edge environment.

3.1. System Model

For minimizing the energy consumption and makespan of the user, it is necessary to make an effective decision regarding the computational offloading. Hence, the DRAW approach performs both the computational offloading technique and task scheduling for executing the task to enhance the profit for both the user and the service provider. The DRAW approach performs the partitioning of the workflow applications into the several subsets of workflow application SU_i. Let us assume, the incoming workflow application W = (T, E) from the users, where W comprises n number of tasks $T = \{T_1, T_2,..., T_n\}$ and set of data and control dependencies $E = \{(T_1, T_2) |T_1, T_2 \in T \wedge T_1 \neq T_2\}$. In the workflow, the edge (T_1, T_2) represents the dependencies between the workflow task T_1 and T_2. The task T_{ent} is the entry task of the workflow application; that is, T_{ent} has no parent task and children task such as T_1, T_2, T_3. In the workflow application, the tasks T_1, T_2, T_3 are the parents and T_4, T_5, T_6 are their respective children. In this case, the workflow tasks are grouped to form as the subset of the workflow application for reducing communication latency and cost. The subset of workflow application is formed as $SU_i = \{T_1, T_4\}$ $SU_i = \{T_2, T_5\}$ $SU_i = \{T_3, T_6\}$. In the DRAW approach, the Cloud environment contains 'm' number of hosts $H_c = \{H_1, H_2,..., H_m\}$, and the Edge environment contains 'v' number of devices $D_E = \{D_1, D_2,..., D_v\}$, for executing the workflow application. Then, estimate the energy consumption EC and the execution time ET for the workflow tasks in edge and Cloud environment. The total time consumption estimated for executing the subset of the workflow application i in the edge environment is denoted as TET^{Cl}_i and cloud environment is denoted as TET^{E}_i. Similarly, the energy consumption estimated for executing the subset of the workflow application i in the Edge environment is denoted as $EC^{E}_{i,}$ and in the Cloud, it is denoted as EC^{Cl}_i. Estimating the execution time and energy consumption in the DRAW approach helps to make an offloading decision for completing the whole workflow application within its assigned deadline DL using the genetic algorithm. In the genetic algorithm, gene g represents an individual in the set of offloading decision of the subset i, and chromosome C_k denote an individual solution for the offloading decision on the number of subsets in the workflow application, and population P is the set of all possible solutions regarding execution of subsets in the Edge or Cloud environment.

3.2. Problem Formulation

This paper aims to perform decision making regarding the offloading of the subset of the workflow application with the minimization of both the execution time and energy consumption. Many of the existing approaches fail to consider the completion time of the previously running tasks or subsets in the cloud or edge environment. The estimation of the completion time of the previously running tasks or subsets changes the offloading decision of the subset. Hence it is necessary to check the starting time of the subset of the workflow application along with its execution time TET^{E}_{i} in the edge environment. Similarly, for the cloud, it is essential to estimate the starting time of the subset along with its total time consumption TET^{Cl}_{i} while making an offloading decision. From the estimated time, the minimum time-consuming environment is chosen for executing the subset.

$$TET_i = \min\left\{\sum_{E_i} TET_i^{E},\ \sum_{C_i} TET_i^{Cl}\right\} \tag{1}$$

Correspondingly, from estimated energy consumption of the subset in the various environments, the minimum energy consuming environment is chosen for execution

$$EC_i = \min\left\{\sum_{E_i} EC_i^{E},\ \sum_{C_i} EC_i^{Cl}\right\} \tag{2}$$

However, the objective of this paper is to minimize both of them efficiently. It is challenging to minimize both the time and energy consumption unitedly. Hence, it is an essential thing to couple these two factors for minimizing them. Thus, the coupled objective function is constructed as follows:

$$C(TET_i, EC_i) = a \times TET_i + b \times EC_i \tag{3}$$

In equation (3), a and b are the two parameters helps to alter the weightage based upon the requirement of the workflow application, that is, for making the offloading decision for a delay sensitive application, the weightage of the time is higher; otherwise, the weightage of energy is higher or equal to time. Finally, the optimization of the objective function is as follows:

$$\min C(TET_i, EC_i) \tag{4}$$

$$\text{Subject to } TET_i \leq DL$$

$$ST(SU_i) = \max\left\{TT_i^{Cl},\ \max_{T_j \in pred(T_i)}\left\{ET_j + RT_j\right\}\right\} \tag{5}$$

In equation (5), finds the starting time of the subset i, that is, $ST(SU_i)$ by finding the execution time of the previous subset. If the predecessor of subset i, that is, j executes in the Cloud environment, then the appropriate $ST(SU_i)$ is obtained by finding the transmission time of the subset i to the Cloud or by detecting the execution time ET_j in the cloud and the receiving time RT_j of j to the edge environment. The maximum value from these two sets is used as the start time of the subset i.

4. DRAW Methodology

The DRAW approach makes an effective decision making in the cloud computing regarding the offloading to the resource-rich environments, including Cloud or Edge, in the Edge environment. The DRAW model schedules the task effectively by reducing the time, cost, and energy consumption. This model efficiently uses the resources within the minimization of needless migration.

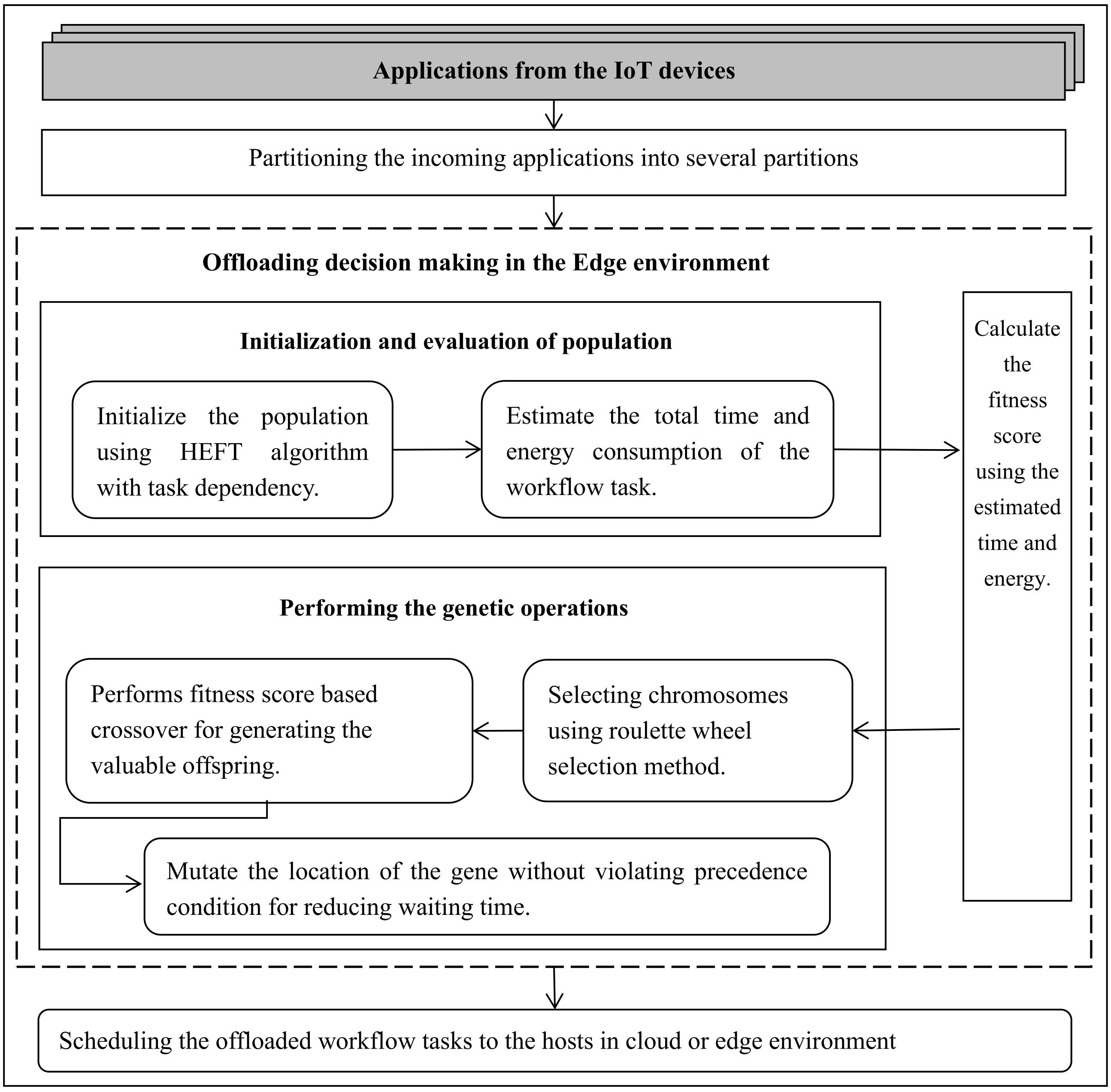
Applications from the IoT devices
Partitioning the incoming applications into several partitions
Offloading decision making in the Edge environment
Initialization and evaluation of population
Initialize the population using HEFT algorithm with task dependency.
Estimate the total time and energy consumption of the workflow task.
Calculate the fitness score using the estimated time and energy.
Performing the genetic operations
Selecting chromosomes using roulette wheel selection method.
Performs fitness score based crossover for generating the valuable offspring.
Mutate the location of the gene without violating precedence condition for reducing waiting time.
Scheduling the offloaded workflow tasks to the hosts in cloud or edge environment

Figure 1. Offloading decision model in the Cloud/Edge environment.

Figure 1 shows the Offloading decision model regarding the execution of subset in the Edge or Cloud environment, and scheduling of the subset in its respective executing environment. On DRAW approach, the dependent tasks are placed on the same host or environment for reducing the communication rate. The DRAW approach takes an effective offloading decision to minimize the execution time, energy consumption without violating its dependencies. Initially, the DRAW approach partitions the incoming workflow applications into several subsets of workflow application and assign a rank to the dependent task based on its priority using Lookahead Heterogeneous Earliest Finish Time (HEFT) algorithm. Then, it initializes the population within the fixed size and several iterations using the HEFT algorithm without violating the dependency between the task of the workflow application. Later, the DRAW approach estimates the time, and energy consumption during the execution of the subset of workflow application in Edge, and the Cloud environment. By using these estimated values, the fitness value is calculated for all the in the enhanced genetic algorithm for performing an effective offloading decision making. Then, the DRAW approach uses the roulette wheel selection method for selecting the chromosomes from the population for performing the genetic operation, including crossover and mutation. After selecting the subset of workflow application, it generates a valuable offspring using the two parent chromosomes using the estimated fitness score. Subsequently, the mutation operation is performed reallocating the position of the gene in the chromosome without violating precedence condition for reducing the waiting time of the offspring generated in the previous operation. After the successful offloading of the subset of workflow application, scheduling of the workflow task occurs effectively in the hosts of the Cloud for minimizing the energy consumption, migration rate, and performance degradation. The DRAW approach efficiently minimizes the objective factors than the existing offloading approaches.

4.1. Effective Decision Making for Offloading the Workflow Tasks to the Cloud/Edge Hosts Using the Genetic Algorithm

Most of the IoT devices offloads its workflow application to the nearby edge devices and executes the workflow application in those devices. However, executing the task in the edge environment is not efficient to attain the reduce execution time and energy consumption. Therefore, most of the existing approaches perform the execution of delay sensitive tasks in the edge environment and offload the remaining tasks to the Cloud environment. However, offloading all these tasks to the cloud environment is not favorable in most of the time; hence, it is necessary to make an effective decision regarding the task offloading. Therefore, many researchers propose several systems for making the computational offloading decision to maximize profit for the users. Although, the existing system lacks to offload the tasks with the minimization of time and energy consumption. Also, they lack to maintain the dependency of the workflow tasks, that increases the communication cost, transmission and receiving time. Hence, the DRAW approach performs an optimal offloading decision of a subset of workflow application for maintaining the task dependency throughout the task execution using an enhanced genetic algorithm for decreasing the objective factors. However, the execution of the subset occurs based on its priority since they are dependent. Hence, before the execution of the subset, they are sorted based on the rank, that is calculated using the Lookahead HEFT algorithm [24]. The DRAW approach uses an enhanced genetic algorithm for making the computational offloading decision, since it accurately solves the mathematical problems, and requires less processing time than the other heuristic algorithms. The DRAW algorithm effectively improves resource utilization through the proper allocation of the workflow task by maintaining its order.

4.1.1. Initialization and Evaluation of Population. The DRAW model uses the genetic algorithm for finding the approximate solutions regarding the execution of subsets in the Edge or Cloud environment. In the genetic algorithm, it is necessary to initialize the population by initializing, set of individual solutions, that consists of genes. It is one of the metaheuristic algorithms produce quality solutions for problems such as optimization and searching problems. In this section, the initialization of population and the calculation of the fitness value occurs as follows:

4.1.1.1. Initialization. In the Initialization phase, the set of possible solutions based on the offloading of the subset of workflow application to the edge or cloud environment is initialized using the HEFT algorithm. The subset of workflow application consists of the collection of the dependent task which is obtained during the partitioning of workflow application used for maintaining the task dependencies. The set of these possible solutions is denoted as population, and it is defined as follows:

$$P=\{C_1, C_2,..., C_k,...,C_S\} \tag{6}$$

In equation (6), S is the size of the initial population, where k = 1, 2, ... , S. In equation (6), the individual solution that is, chromosome in the population is represented as C_k.

$$C_k=X_1^k,\ X_2^k,\ ...,X_i^k,...,\ X_N^k \tag{7}$$

In equation (7), i = {1, 2, ..., N} where N is the total number of subsets from the partitioned application of the IoT device. In equation (7), $x^k{}_i = 0$ denotes that the subset is offloaded to the edge devices and, $x^k{}_i = 1$ denotes that the subset of the workflow application is offloaded to the cloud host.

4.1.1.2. Fitness Function. In the DRAW approach, calculating the fitness function helps to find the optimal solution from the possible solutions. For each chromosome, the DRAW approach evaluates the fitness function to decrease both the execution time and energy consumption. For calculating the fitness function, both the execution time and the energy consumption of the subset of the workflow application at the edge and Cloud environment is estimated as follows:

- Estimation of the execution time

For minimizing the time consumption, it is an essential task to estimate the total time consumed for executing each subset of the workflow application. In the DRAW approach, an effective offloading decision making occurs by estimating an appropriate execution time of the subset of the workflow application. Using the estimated value, a suitable environment is declared for executing these subsets.

- Cloud computing

This section estimates the execution time of the subset of the workflow application i in the Cloud computing environment is as follows:

$$\text{Execution time}\left(ET_i^{Cl}\right)=\frac{WL_i}{DC_{H_C}\times k_{H_C}} \tag{8}$$

In equation (8), WL_i is the workload of the subset of the workflow application i, DC_{H_C} is the capacity of the host H_c in Cloud environment and k_{H_C} *is* the average utilization of the processor executing the subset in the Cloud. Then, the time required for the transmission the subset is as follows:

$$\text{Transmission time}\left(TT_i^{Cl}\right)=\frac{CCR_s+WL_i}{CUL_t} \tag{9}$$

In equation (9), CCR_s is the size of the computational request for executing the subset in the cloud environment, WL_i is the input size of the subset of the workload application required for computing the Cloud and CUL_t is the average uplink bandwidth of the entire trace [25]. Similarly, the time required for receiving the executed subset is as follows:

$$\text{Receiving time}\left(RT_i^{Cl}\right)=\frac{CRR_s+RWL_i}{CDL_t} \tag{10}$$

Equation (10) estimates the receiving time of the computed subset of the cloud computing environment to the environment of the immediate successor subset. In the equation (10), CRR_s is the size of the response of the request for executing the subset in the cloud environment, RWL_i is the size of the resultant data on which the computations are performed, and CDL_t is the average downlink bandwidth of the entire trace. The start time of the subset i $ST(SU_i)$ is in the Cloud environment is calculated as follows:

$$ST_1(SU_i)=\max_{T_j\in predT_i}\{ET_j+RT_j\} \tag{11}$$

In equation (11), $ST(SU_i)$ is the start time of the subset i, is calculated by choosing the maximum value of the summation of execution time and the receiving time of the predecessor subset j or the transmission time of the subset. Thus, the total estimated execution time is as follows:

$$\text{Total Estimated Execution time}\left(TET_i^{Cl}\right)=ET_i^{CL}+TT_i^{Cl}+RT_i^{Cl}+ReT_i^{Cl}+ST_1(SU_i) \tag{12}$$

In equation (12), $ReT^{Cl}{}_i$ is denoted as the waiting time of the subset i for starting its execution in the Cloud environment. The value of $ReT^{Cl}{}_i$ is the minimum execution time of the immediate predecessor of the subsets i in the Cloud environment.

- Edge computing

In this section, the DRAW approach estimates the completion time of the subset i in the edge environment is as follows:

$$\text{Estimate Execution time}\left(ET_i^{E}\right)=\frac{WL_i}{DC_{D_E}\times k_{D_E}} \tag{13}$$

In equation (13), the DC_{DE} is the capacity of the device D_E in edge environment, and k_{DE} is the average utilization of the processor executing the subset in the edge environment. Then, the start time of the subset i in the edge environment is calculated as follows:

$$ST_2(SU_i)=\max_{T_j\in predT_i}\{ET_j+\lambda\} \tag{14}$$

In equation (14), the start time of the subset of the workflow application SU_i is calculated by choosing the maximum value of the summation of execution time and the receiving time of the predecessor subsets. In equation (14), ET_j is the execution time of the subset j and $\lambda = 0$, that is the subset i is executed in the Edge environment; else $\lambda = RT_j$, that is the subset i is executed in the Cloud environment. Thus, the total estimation time is as follows:

$$\text{Total Estimated Execution time}\left(TET_i^E\right)=ET_i^E+ST_2(SU_i)+ReT_i^E \tag{15}$$

In equation (15), ReT^E_i is denoted as the waiting time of the subset i for starting its execution in the edge environment. The value of ReT^E_i is the minimum time consumption of the immediate predecessor of the subset i in the edge environment. Using equation (12) and (15), the least time consumption environment is predicted as a suitable environment for executing the subset i of the workflow application.

- Estimation of energy consumption:

The DRAW approach aims at choosing an optimal strategy for improving the optimal offloading decision. It minimizes both the time and energy consumption while executing each subset of the workflow application. The DRAW approach calculates the energy consumed during the execution of a subset for deciding the offloading of the subset of the workflow application.

- Cloud computing:

The energy consumption for the subset i executing in the cloud computing environment is estimated as follows:

$$\text{Transmitting energy}\left(TE_i^{Cl}\right)=\left(TT_i^{Cl}\times TP_i^{trans(Cl)}\right)+\left(RT_i^{Cl}\times TP_i^{rec(Cl)}\right) \tag{16}$$

In equation (16), $TP_i^{trans(cl)}$ is the power consumption for transmitting the workflow subset i from the edge environment to the cloud environment, and $TP_i^{rec(cl)}$ is the power consumption for receiving the subset i from the remote environment to the edge environment. Thus, the total energy required for executing the subset i in the cloud environment is as follows:

$$\text{Estimated Energy consumption}\left(EC_i^{Cl}\right)=\left(ET_i^{Cl}\times E_{Cl}\right)+TE_i^{Cl} \tag{17}$$

In equation (17), E_{Cl} is represented as the energy coefficient value of the processor speed of the Cloud environment. Similarly, the energy consumption required for executing the subset i in the edge environment is estimated as follows:

- Edge computing:

Total energy consumption occurs during the execution of the subset of the workflow application i in the edge environment is estimated as follows:

$$\text{Estimated Energy Consumption}\left(EC_i^E\right)=ET_i^E\times E_E\times Au_m \tag{18}$$

In equation (18), E_E is represented as the energy coefficient value of the processor speed of the edge environment; Au_m is used for the calculating the energy consumption happens during the execution of the subset of the workflow application by avoiding the overall energy consumption of the edge device.

- Fitness score calculation:

The DRAW approach evaluates the possible solutions for finding the optimal solution using the fitness function. The genetic algorithm calculates the fitness value using the objective function. In the DRAW approach, the objective function $FV(SU_i)$ is developed based on two parts, such as total time and energy consumption to minimize them.

$$FV\left(W_{i,y}\right)=\left[a\times\left[\sum_{E_i} TET_i^E+\sum_{Cl_i} TET_i^{Cl}\right]\text{-}DL\right]+\left[b\times\left[\sum_{E_i} EC_i^E+\sum_{Cl_i} EC_i^{Cl}\right]\right] \tag{19}$$

In equation (19), the value represents the hosts in the Cloud and edge devices, that is, $y = H_C + D_E$, $H_C \subseteq y$, $D_E \subseteq y$. In equation (19), DL is the deadline of the workflow application W used for finding better fitness value. In equation (19), the value of the variable a and b changes based on the characteristics of the given application. For the delay sensitive application such traffic management system or healthcare monitoring, the value of a is higher than b that is $a>b$; otherwise, the value of a is lesser than or equal to b that is $a\leq b$. In the genetic algorithm, two chromosomes are selected from the population using the roulette method for performing the genetic operations, including mutation or crossover. The proposed genetic operation generates the new population, and the generated population generates the other population set. This iteration continues until it reaches the best solution or it reaches the maximum number of iterations.

4.1.2. Genetic Operations. The genetic operation is responsible for obtaining the best solution from the possible solutions. In the DRAW approach, the fitness score is calculated efficiently in the selection, crossover, and mutation for improving the decision regarding the offloading. The calculation of the fitness score for acquiring the best solution from the possible solutions using the different process, including selection, crossover, and mutation is as follows:

4.1.2.1 Selection. In the DRAW approach, the selection of chromosome plays a vital role in generating the further population for attaining the best solution. The DRAW approach uses the roulette wheel selection method for selecting the parent chromosome. For recombination, the selection of a chromosome in the genetic algorithm is calculated as follows:

$$\mathrm{SP_i}=\frac{\mathrm{FV(W_{i,y})}}{\sum_{k=1}^{S}\mathrm{FV(W_k)}} \tag{20}$$

The selection phase selects the best two chromosomes for creating the offspring to find the optimal solution for offloading the subset of the workflow application.

4.1.2.2. Crossover. The DRAW approach uses an adaptive crossover for producing an offspring with high quality. For generating the next generation of the population with a higher quality of the solution, the crossover section in the genetic algorithm combines the two selected individuals Pr_1 and Pr_2 as shown in the figure 2. In this section, the fitness value based on the execution time and the energy consumption for each gene in the individual is calculated as follows:

$$\mathrm{F(W_g)}=\sum_i\left[\left[[\mathrm{a}\times\mathrm{TET_i}]-\mathrm{DL_i}\right]+[\mathrm{b}\times\mathrm{EC_i}]\right] \tag{21}$$

In equation (21), DL_i is the deadline of the subtask i in the workflow application is calculated as follows. $DL_i=DL/N$. In equation (21), TET_i and EC_i are the execution time and energy consumption of the subset of the workflow application i in edge environment or the cloud environment. Based on the equation (21), the DRAW approach generates an offspring Ch_1 by comparing the fitness value of each gene in the two parent chromosomes. Then, the genes with better fitness value from the parents are copied to the offspring.

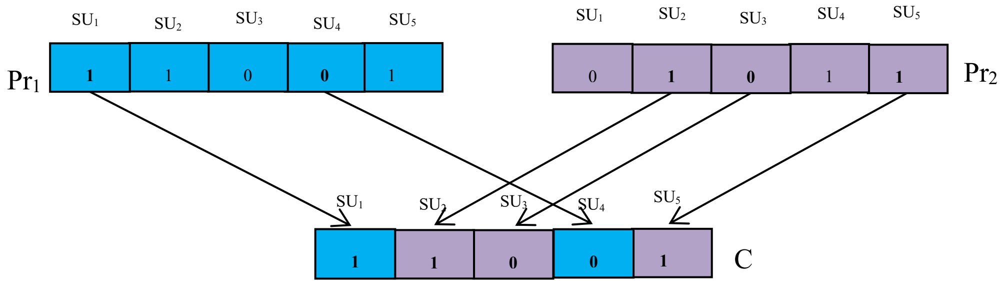

Figure 2. Crossover operation.

4.1.2.3. Mutation. As shown in the figure 3, the position of the genes in the chromosomes are modified to attain the best solution for reducing the waiting time of the subset. The DRAW approach selects the gene g_q in the chromosome and searches for its immediate successor g_r until the end of the queue. Suppose, the other gene g_l lies between the g_q and g_r then, and its predecessor lies before the gene g_q then, this mutation operation changes the location of both the g_q and g_l. This mutation operation continues until the location of the gene $g_q = g_{r-1}$.

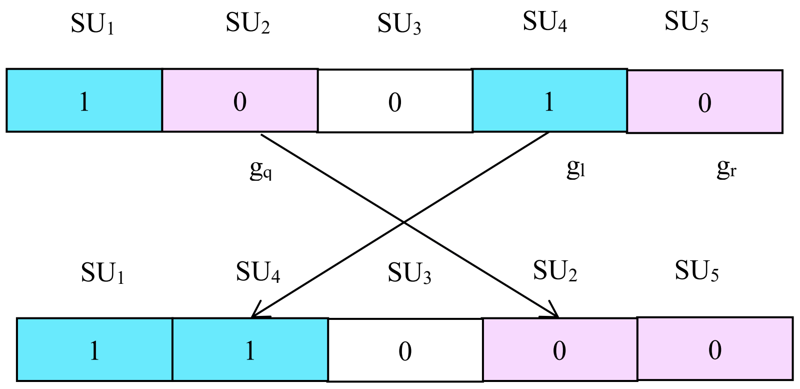

Figure 3. Mutation operation.

4.2. Scheduling of the Offloaded Tasks

After the offloading decision, the DRAW approach performs effective task scheduling in the decided environment within the minimized execution time and energy consumption. The task scheduling algorithm effectively selects the suitable resources in the decided environment and schedules the task intending to maximize resource utilization. A proper task scheduling can balance the load of the system, that results in the minimization of the migration rate and also improves the quality of service (QoS). It is responsible for improving the utilization of the resources and decreasing the time consumption. However, the scheduling of the workflow application in quite tricky, since it is necessary to maintain the dependency of the workflow task while scheduling. Hence, the DRAW approach estimates the load of the host before scheduling the task to the particular host. The DRAW approach uses three threshold approach for categorizing the hosts in the cloud environment. The hosts are divided into four types: Overloaded, Moderate-loaded, Subpar-loaded, Under-loaded hosts. The incoming tasks are scheduled to the moderate-loaded and subpar-loaded hosts for the effective task execution. This scheduling results in the minimization of the performance degradation, energy consumption, execution time, and migration rate. In order to avoid needless migration, it is necessary to monitor the current load of the task in the Cloud or edge environment.

```
Input: Workflow application W of IoT devices
Output: obtaining optimal decision from P for offloading all tasks
    Step 1: Partitioning the incoming application into several subsets and rank them using
    Lookahead HEFT algorithm
//Offloading decision for all the subset of the workflow application
//Initialize population P, chromosome $C_k$, and gene g
Set population size S, iteration IT, mutation probability pm, crossover probability pc, n
for ( $P \leq S$ ) then
    for ( $C_k \leq N$ ) then
        Create gene
    endfor
    Create chromosome using HEFT algorithm
endfor
//Calculating the fitness function
    Step 2: Estimating $FV(W_{i, y})$ based on energy consumption and execution time using equation
    (14).
for ( $C_j$ = optimal solution) then
    Step 3: Calculate $SP_i$ values for all chromosome using equation (15)
    Step 4: perform crossover for first two chromosome
    Step 5: Calculate $F(W_g)$ for each gene of two else chromosome using equation (16) and pass it
    to the offspring
    Step 6: perform mutation based on the successor of the subset $g_q$.
    Step 7: if ( $ch_1$ is better than $Pr_1$ and $Pr_2$ ) then
                Repeat the step 2 until obtaining the best solution
            else
                Mark the offspring with the good fitness score as the best solution at the IT^th
            iteration.
```

Algorithm 1. The Pseudocode of the DRAW approach.

Algorithm 1 clarifies the overall procedure of the DRAW approach. The DRAW approach effectively performs the offloading decision making for the subset of the workflow application to its execution environment in the Edge environment. The DRAW approach initializes the population using the HEFT algorithm by the creation of the possible solutions within the fixed size. Then, set the crossover probability, mutation probability, and several iterations for finding the possible solutions. Now, estimate the execution time, energy consumption of the subset in the various environments for calculating its fitness function using the genetic algorithm. From the possible solution in the population, two parents are selected using the roulette wheel selection method. The combination of these two parents produces higher quality offspring for improving the searching capability. A high-quality offspring is created by copying the genes of the parents that have better fitness value $F(SU_g)$. Then, the improvement of generated offspring occurs using the mutation dependent task operation in the genetic algorithm. Subsequently, the mutation operation reallocates the position of the gene with its decision in the chromosome to minimize the waiting time. After that, repeat these steps until it satisfies the condition.

5. Experimental Evaluation

This evaluation framework proves that the DRAW approach effectively minimizes the energy consumption and execution time while offloading the task to the destination hosts using genetic algorithm by conducting the experiments compared to the existing task offloading technique.

5.1. Experimental Setup

To examine and demonstrate the performance of the DRAW methodology, it provides a setup for the analysis of the DRAW system and the existing system using the WorkflowSim environment, that is a modeling and simulation framework for infrastructure services. The WorkflowSim environment comprises various factors including several IoT devices, bandwidth, the number of Edge hosts, and the number of VMs to assess the DRAW algorithm through the following evaluation metrics. The input of the workflow task is extracted from the CyberShake_30.xml, Cyber-Shake_50.xml, Cyber-Shake_100.xml.

5.1.2. Evaluation Metrics.

- Energy consumption:

Energy consumption refers to the utilization of energy during the execution of workflow tasks in the Edge or Cloud environment.

- Total execution time:

It denotes the total time consumed by the workflow task while the execution of tasks occurs in the Edge or Cloud environment.

- Average latency:

Latency is defined as the task that fails to complete its execution within its assigned deadline; that is, the deadline of the task is minimum than the execution time of the task [DL $\geq$ TET^{Cl}_{i}].

$$LT = TET_i^{Cl} - DL \tag{22}$$

If the value of LT is negative, then the task executes with no latency; otherwise, the task executes with latency.

5.2. Evaluation Results

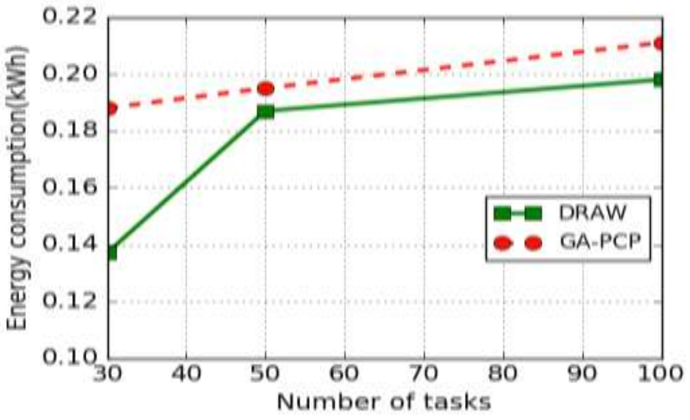

Figure 4. Energy consumption vs. the number of tasks.

Figure 4 illustrates that the DRAW approach compares the energy consumption with the rate of workflow tasks, that is extracted from the IoT applications as the range of tasks such as 30, 50,100. It is determined that the increment in the task rate maximizes the energy consumption in the Cloud and Edge environment. The existing approach, namely a greedy algorithm based on the partial critical path (GA-PCP) maximizes energy consumption as the number of tasks increases. Similarly, the DRAW approach also consumes a number of increases as the number of tasks increases; still, this value is less than the GA-PCP approach. Owing to the fact, GA-PCP approach mainly focuses on minimizing the latency;

conversely, the DRAW approach focuses on both time and energy consumption. From figure 5, it is analyzed that the DRAW approach consumes 4.103% less than the GA-PCP approach. From the above statement, it is clear that the DRAW methodology consumes minimum energy than the existing GA-PCP. The DRAW methodology makes an effective decision execution of the task in the Cloud or edge environment for minimizing the energy consumption during the execution of several tasks.

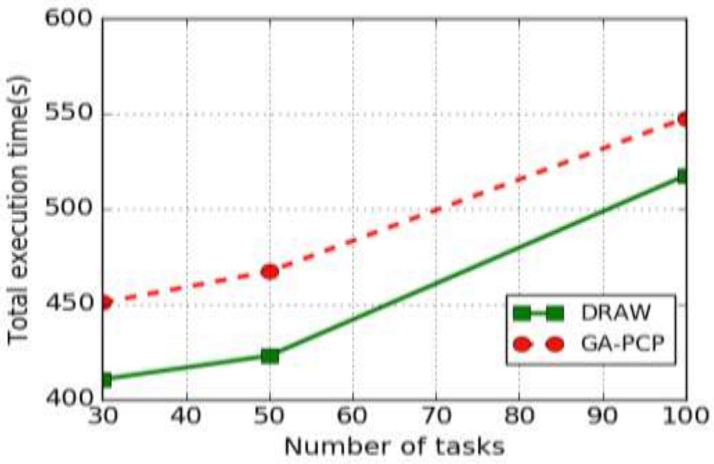

Figure 5. Total execution time vs. the number of tasks.

Figure 5 portrays that in the result of the DRAW approach, the total execution time increases as the number of tasks rise such as 30, 50, 100. It is known that the maximization of the total execution time depends on the rate of the workflow task. Although both GA-PCP approach and DRAW approach increases the total time consumption concerning the rate of workflow tasks, the DRAW method balances the time consumption in a specific limit. As per the fact, GA-PCP model assigns the entry and exit tasks on the mobile device; this increases both the time and energy consumption. For instance, the total execution time for the 30 workflow tasks is about 450.833 seconds, and for DRAW approach is 410.545 seconds. From this statement, it is noted that the DRAW approach reduces 8.94% of total execution time than the GA-PCP approach. It proves that the makespan of the workflow using GA-PCP DRAW method is higher than the DRAW approach. The DRAW approach effectually offloads the task to a particular environment based on its nature and requirement.

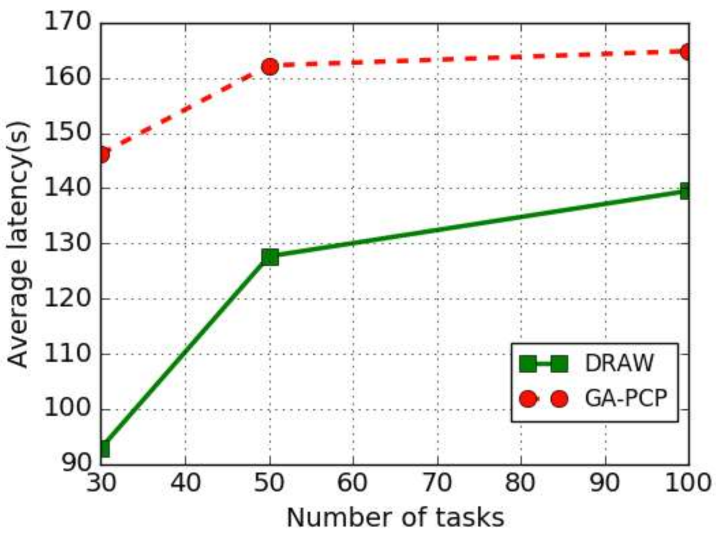

Figure 6. Average latency vs. the number of tasks.

Figure 6 compares the latency of the DRAW approach and the existing approach based on the increment in the rate of the task. It is observed that the DRAW approach completes the execution of the task within the assigned deadline. From the graph, it is proved that the DRAW approach minimizes the average latency on comparing with the existing GA-PCP method. Also, the GA-PCP approach executes the workflow task with the latency of 18.154% higher than the DRAW approach. The DRAW methodology uses an effective genetic algorithm for completing the execution of tasks within the assigned deadline by making an effective offloading decision for the workflow task. It is proved that the DRAW approach saves the energy and executes the workflow task with the minimum time consumption than the existing GA-PCP approach.

6. Conclusion

The DRAW approach performs the offloading decision and scheduling of IoT workflow application in the cloud or Edge environment. It makes the best decision regarding the offloading of the subset of the workflow to minimize the objective factors. In the DRAW approach, the communication cost is minimized effectively, since it places the dependent tasks in the same environment. This effective decision making occurs in the edge environment, where the tasks sent to the nearby Edge environment for processing. For effective decision making, the DRAW approach uses a genetic algorithm that obtains the optimal solution within the multiple possible solutions. Notably, the DRAW approach improves the minimization in energy consumption and total execution time than the existing algorithm. The numerical result demonstrates that the DRAW approach reduces the latency of 15.365% than the existing GA-PCP approach.

References

[1] Moghaddam, Faraz Fatemi, Mohammad Ahmadi, Samira Sarvari, Mohammad Eslami, and Ali Golkar, 2015. "Cloud computing challenges and opportunities: A survey," *1st International Conference on Telematics and Future Generation Networks (TAFGEN)*, pp. 34-38.

[2] Varghese, Blesson, and Rajkumar Buyya, 2018. "Next generation cloud computing: New trends and research directions," *Future Generation Computer Systems*, Vol.79, pp.849-861.

[3] Gezer, Volkan, Jumyung Um, and Martin Ruskowski, 2017. "An extensible edge computing architecture: Definition, requirements and enablers," *Eleventh International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, UBICOMM.

[4] Shi, Weisong, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu, 2016. "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, Vol.3, No. 5, pp.637-646.

[5] Singh, Lovejit, and Sarbjeet Singh, 2013. "A survey of workflow scheduling algorithms and research issues," *International Journal of Computer Applications,* Vol.74, No. 15.

[6] Atkinson, Malcolm, Sandra Gesing, Johan Montagnat, and Ian Taylor, 2017. "Scientific workflows: Past, present and future," pp.216-227.

[7] Mach, Pavel, and Zdenek Becvar, 2017. "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, Vol.19, No. 3, pp. 1628-1656.

[8] Lyu, Xinchen, Hui Tian, Cigdem Sengul, and Ping Zhang, 2017. "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Transactions on Vehicular Technology*, Vol.66, No. 4, pp.3435-3447.

[9] Zhao, Pengtao, Hui Tian, Cheng Qin, and Gaofeng Nie, 2017. "Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing," IEEE Access, Vol.5, pp.11255-11268.

[10] Zhu, Qiliang, Baojiang Si, Feifan Yang, and You Ma, 2017. "Task offloading decision in fog computing system," China Communications, Vol.14, No.11, pp.59-68.

[11] Liu, Zhanghui, Xuee Zeng, Wensi Huang, Junxin Lin, Xing Chen, and Wenzhong Guo, 2016. "Framework for context-aware computation offloading in mobile cloud computing," *15th IEEE International Symposium on Parallel and Distributed Computing (ISPDC)*, pp.172-177.
[12] Du, Jianbo, Liqiang Zhao, Jie Feng, and Xiaoli Chu, 2018. "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications,* Vol.66, No.4, pp.1594-1608.
[13] Zhou, Bowen, Amir Vahid Dastjerdi, Rodrigo N. Calheiros, and Rajkumar Buyya, 2018. " An Online Algorithm for Task Offloading in Heterogeneous Mobile Clouds," A*CM Transactions on Internet Technology (TOIT)* Vol.18, No. 2 pp.23.
[14] Chen, Meng-Hsi, Ben Liang, and Min Dong, 2018. "Multi-user Multi-task Offloading and Resource Allocation in Mobile Cloud Systems," *arXiv preprint* arXiv:1803.06577.
[15] Goudarzi, Mohammad, Mehran Zamani, and Abolfazl Toroghi Haghighat, 2017. "A fast hybrid multi-site computation offloading for mobile cloud computing," *Journal of Network and Computer Applications*, Vol.80, pp. 219-231.
[16] Guo, Songtao, Bin Xiao, Yuanyuan Yang, and Yang Yang, 2016. "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," *35th Annual IEEE International Conference on Computer Communications*, pp. 1-9.
[17] Jiang, Zhefeng, and Shiwen Mao, 2015. "Energy delay trade-off in cloud offloading for mutli- core mobile devices," *IEEE Global Communications Conference (GLOBECOM)*, pp. 1-6.
[18] Liu, Juan, Yuyi Mao, Jun Zhang, and Khaled B. Letaief, 2016. "Delay-optimal computation task scheduling for mobile-edge computing systems," *IEEE International Symposium on Information Theory (ISIT)*, pp. 1451-1455.
[19] Wang, Yanting, Min Sheng, Xijun Wang, Liang Wang, and Jiandong Li, 2016. "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Transactions on Communications*, Vol.64, No.10, pp.4268-4282.
[20] Zeng, Ming, Yong Li, Ke Zhang, Muhammad Waqas, and Depeng Jin, 2018. "Incentive mechanism design for computation offloading in heterogeneous fog computing: A contract-based approach," *In IEEE International Conference on Communications (ICC),* pp.1-6.
[21] Chen, Min, and Yixue Hao, 2018. "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, Vol.36, No.3, pp.587-597.
[22] Kattepur, Ajay, Harshit Dohare, Visali Mushunuri, Hemant Kumar Rath, and Anantha Simha, 2016. "Resource Constrained Offloading in Fog Computing," *ACM Proceedings of the 1st Workshop on Middleware for Edge Clouds & Cloudlets,* 1 p.
[23] Zhao, Pengtao, Hui Tian, and Bo Fan, 2016. "Partial critical path based greedy offloading in small cell cloud," *84th IEEE Vehicular Technology Conference (VTC-Fall)*, pp.1-5.