DAG DATA OFFLOADING FOR IOT-ASSISTED CLOUD EDGE COMPUTING

Baskar M[1], Dr.J.Chenni Kumaran[2]

[1]*Associate Consultant, Slogix (OPC) Pvt. Ltd., Chennai, Tamilnadu, India*
[2]*Professor, Department of IT, Panimalar Institute of Technology, Chennai, Tamilnadu, India*

---***---

Abstract -*The most challenging problem for a mobile cloud service provider is providing the quality of service through minimizing the execution time and reducing fast discharge of mobile device batteries. Therefore, the mobile cloud service provider needs an efficient mobile application offloading algorithm for execution to edge devices and centralized datacenter. We propose a task offloading framework that reduces execution time of mobile application and drastically reduces energy consumption of mobile devices. The proposed system employs a combination of NSGA-III (non-dominated sorting genetic algorithm) with DE (Differential Evolution) algorithms to optimize the multi-objective problem of directed acyclic graph workload offloading in cloud or edge computing(cloudlet). The proposed model is designed and simulated in WorkflowSim toolkit, analyzed the experimented metrics with existing model.*

Key Words: Genetic Algorithm, Differential Evolution, Directed Acyclic Graph, Cloudlet, Internet of Things.

1.INTRODUCTION

Nowadays, cloud computing is considered as one of the most powerful trends and widespread computing services which are distributed all over the world. The appearing of Cloud Computing environment enables a new framework that changes the physical location of computation and storage into the network to reduce operational and maintenance costs. In high performance computing, Cloud computing is considered a new tool and asset. Cloud computing is internet-based and central service provision in order to maintain data and applications. Consumers use different applications through the internet by cloud computing without installing software. Besides, cloud computing is used in high performance computing to centralized storages, memory, processing and bandwidth. In dispersed registering, a cloud is a lot of coursed laptops which gives on-demand computational resources or organizations to the far-off customers over a framework [10].

Also, in this situation, Internet of Things trying to make availability of all services on line and can be accessed globally. Sometimes, mobile devices act as an internet of things to sense the user's environment and generate huge amount of data. These data should be processed in an application running on the mobile device and provide the services to end users. When the big data are processed on a handheld instrument of mobile device which limits the availability of computational and data resources, it surpasses the constrained execution and response time and it leads to consume more energy. In this scenario, internet of mobile things searches supports from cloud resources for computational, transferring and storing data. But the mobile devices are connected through wide area network to cloud resources. Due to separation of mobile devices geographically far from cloud resource, it increases response time, delay of data transferring between computational resources and drastically reduces the performance of mobile services. To overcome this situation, edge devices help to cloud datacenter through enclosing a small datacenter between edge devices and cloud datacenter with average computational resources, base stations and access point at edge of radio access network. This small datacenter (Cloudlet) is interconnected to mobile devices via local area network with high bandwidth and low latency. The mobile devices can offload their computation tasks either to cloud datacenter or cloudlet supported edge devices to increase its performance by minimizing latency and response time[1]. The edge device offloading processes construct a green cloud computing environment.

This proposal is organized as follows. In Section II, theories and previous works on hybrid offloading strategy, GA, and energy consumptions are surveyed. Section III illustrates the proposed system architecture of DCOM. The flows of experiment for each module in testing the benchmark functions are described in section IV. Analysis and discussion on the experimental results are briefed in Section V. Conclusion and future enhancements are demonstrated in section VI.

2. LITERATURE REVIEW

Compared with the cloud, Edge Computing is an emerging technology that aims at pushing applications and content close to the users to reduce latency, improve the quality of experience and ensure highly efficient network operation and service delivery, which has the potential to address the concerns of response time requirement, battery life constraint and bandwidth cost saving [3].

In a system of mobile cloud computing based on unmanned aerial vehicles (UAVs) to reduce the mobile energy consumption through computation offloading. In the system, joint optimization of bit allocation and trajectory of cloudlet was proposed. Jin et al. proposed an incentive compatible mechanism (ICAM) to distribute cloudlets based on the demand of mobile services.

The proposal of a Markov Decision Process (MDP) to seek a hybrid offloading strategy in mobile devices, the edge and the cloud, while optimizing the execution time and the energy consumption. A research offloading from a mobile device to several edges. In such scenario, task allocation and central process unit frequency of the mobile device are optimized to reduce the execution latency and energy consumption of the mobile device[11].

To demonstrate the feasibility of the proposed approach, we have used the framework and scenario into an edge computing based environment using CloudSim (Calheiros et al., 2011) and WorkflowSim-1.0. In this paper, event simulation functionalities of CloudSim have been used to implementing functionalities of WorkflowSim architecture. CloudSim entities such as datacenter and communication amongst datacenter through message sending operations are included. Therefore, the core CloudSim layer is responsible for handling events between fog computing components in WorkflowSim.

One of the scheduling issues is to allocate the correct resource to the arriving tasks. The dynamic scheduling process is considered complex if several tasks arrive at the same time, Therefore, they have introduced a system to avoid this problem by allowing the arrived tasks to wait in a queue and the scheduling will recompute and sort these tasks. Therefore, the scheduling is done by taking the first task from the queue and allocated to the resource that will be the best fit using GA. The objective of this system is to maximize utilization of resources as also reduce execution time.

Also, cloud computing provides computational resources to users on the Internet as a public service based on the pay-as-you-go model which this services demand by the cloud consumers. From the other point of view, scheduling is considered as a decision-making process that is commonly used in the majority of production and service-providing industries and is employed to optimize efficiency. Constraint Programming (CP) and Linear Programming (LP) can be used to create exact formulations of the VM placement and consolidation problems and find optimal solutions for those problems. However, due to the inherent complexity and NP-hard nature of those problems, the time complexity of such exact solutions turns out to be exponential and thus impractical for large problem sizes. Furthermore, such programming languages do not allow the optimization of multiple objective functions simultaneously.

VMs' consolidation aims at improving resources' usage efficiency by placing under-loaded resources in a suspension or idle state. This is achieved by grouping dispersed VMs on a minimal number of active PMs, then switching off or suspending unused servers – thus saving energy. There are several challenges associated with the VMs' placement and consolidation problems. The first challenge is associated with the NP-hard nature of such optimization problems, which are time and resource consuming to solve. Another challenge is the potential impact of energy saving on the system's performance.

3. SYSTEM ARCHITECTURE

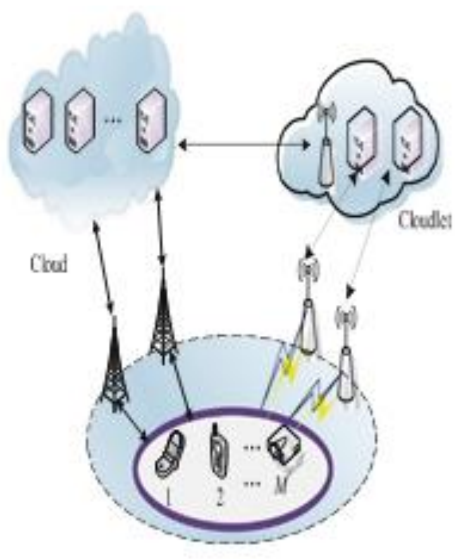

Fig -1: System Architecture of IOT enabled Cloud Edge

Fig.1. shows the System Architecture of IOT enabled Cloud Edge. So, consider a scenario where a cloudlet covers M mobile devices which are connected to a cloud deployed in the remote area. Each mobile application is formalized as a workflow, denoted as a directed acyclic graph (DAG). A workflow contains several vertices and edges-constrained computing tasks. The vertices and edges denote the computational resources(nodes) and the communication between the nodes respectively. The mobile devices are connected to cloudlet through local area network with high bandwidth, and are connected to cloud with low bandwidth [2]. The bandwidth determines the latency of data transferring between the resources and it causes the response time.

This system needs four phases to build and experiment it, first the cloud, cloudlet and edge devices are designed with large, moderate and average computational and data resources respectively. Second, the proposed method decides where to offload the computation tasks whether on cloud datacenter or edge supported devices. Third, the offloaded tasks are scheduled on the resources and finally the edge supported cloud datacenter is monitored and calculated to improve and evaluate the performance of the internet of mobile things.

The proposed design, assures to provide the efficient and effective services to an end user of a mobile service provider by avoiding the exceeds of constrained time service agreements.

This model aims to achieve the multi-objectives in the view of improving performance of an internet of mobile things. This achievement is fulfilled through minimizing execution time and reducing energy consumption of mobile devices.

4. FLOW OF EXPERIMENT

4.1 Methodology

This proposed system is designed and experimented in WorkflowSim tool which was extended from CloudSim, these simulation tools are developed in Java programming language. In this scenario, the proposed method is an enhanced version of existing method of COM and the implementation of proposed system classified as five modules based on the system design and user feasibilities. Each module is having different infrastructural environment and important user and resource parameters to evaluate the system in an efficient manner and the modules are designed as shown in the below section.

4.2 Designing Edge and Cloud Infrastructures.

In this module, the required infrastructural entities are defined to construct the proposed system architecture. The system architecture contains mobile devices, cloudlet and cloud environment and all are interconnected with network edge devices. Hence, the resources of network devices, cloudlet (small datacenter) and cloud datacenter are configured and deployed. The mobile devices and cloudlet are connected via radio wave local area network with high bandwidth, also the mobiles devices and the public cloud are interconnected with low bandwidth of wide area network. The network bandwidths cause the latency of offloading of computation tasks from mobile devices to cloud environment. The entities of cloud and cloudlet's processing capacity, number of processors, memory, bandwidth and storage capacity of servers are defined to design the physical infrastructure. The system architecture of proposed and existing systems is designed in Workflow Sim toolkit to experiment the methods and algorithms for evaluation and analysis.

4.3 Mobile devices and applications

This framework proposes an internet of mobile things, it tries to bring all the services on a handheld instrument of mobile devices. The mobile device is interacted through an application which runs based on time-intensive and energy saving tendency. The configuration of mobile devices and number of devices are defined and designed in the toolkit as a user interface or actuator. Moreover, on each mobile device runs an application and it can be offloaded towards edge supported cloud environment. Table 1 represents the key notations and descriptions which are used in this model.

Table -1: Key notations and descriptions

Notation	Descriptions
M	The number of mobile devices in IoT
Vm	The computing task set of the mth workflow
EDm	The dependency set of the mth workflow
dm,i	The input data of the computing task vm,i receives
wm,i	The computation workload of the computing task vm,i
Xm	The hybrid offloading strategies of the mth workflow
xm,i	The offloading strategy of the computing task vm,i
T L(Xm)	The offloading latency of the network
T e(Xm)	The computing time in executing the mth workflow
T t (Xm)	The transmission time in executing the mth workflow
Ai	The transmission strategies of two computing tasks
Tm(Xm)	The execution time of the mth workflow
EL(Xm)	The offloading energy consumption in executing the mth workflow
Ee(Xm)	The computing energy consumption in executing the mth workflow
Et (Xm)	The transmission energy consumption in executing the mth workflow
Em(Xm)	The energy consumption for the mth mobile device

The application design in WorkflowSim follows the IoT applications are modelled as Directed Acyclic Graph (DAG) file, which can be fragmented as multiple tasks and each is having inter communications between them. A graph is formed by vertices and by edges connecting pairs of vertices, where the vertices can be any kind of object that is connected in pairs by edges. In Fig.2., the circles represented by alphabets are denoted as vertices and the arrows denoted as edges of a DAG file. In the case of a directed graph, each edge has an orientation, from one vertex to another vertex. A path in a directed graph is a sequence of edges having the property that the ending vertex of each edge in the sequence is the same as the starting vertex of the next edge in the sequence; a path forms a cycle if the starting vertex of its first edge equals the ending vertex of its last edge. A directed acyclic graph is a directed graph that has no cycles.

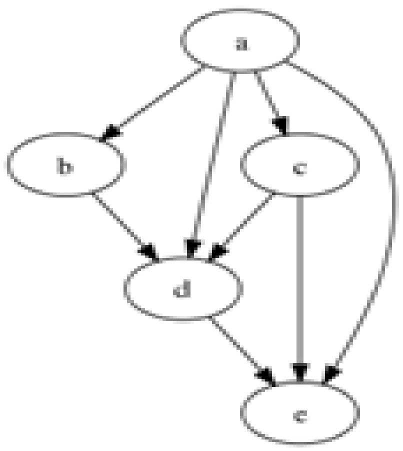

Fig -2: Directed Acyclic Graph

4.4 Tasks offloading and scheduling

This module confirms the dynamic schedules of concurrent workflows in cloud-edge computing. Then NSGA-III (Non-dominated Sorting Genetic Algorithm-III) based DE (Differential Evolutionary) is utilized to find the global optimal solution. Finally, schedule evaluation is conducted based on SAW and MCDM to select the optimal solutions for the computing tasks in the same schedule. In the execution of concurrent workflows, we separate the computing tasks in the workflows into three categories: the scheduled, the ready and the unready. Each time, we implement the computing tasks ready and suppose this process as a schedule. Consider after S schedules, the concurrent workflows finish executions. Let SKD={skds|1≤ s≤ S} represent the computing task sets for S schedules, where skds (1≤ s ≤ S) represents the set of computing tasks executed in the sth schedule. We consider each computing task in the workflow have similar computation load in this paper. It is depicted in Fig.3. that there are two workflows, i.e., WF1 and WF2, for execution. In the first schedule, v1,1 along with v2,1, the root tasks of WF1 and WF2, are ready for execution. Thus, skd1={v1,1, v1,2}, skd2={v1,2, v2,2, v2,3}, skd3={v1,3, v1,4, v2,4}. After three schedules, the two workflows finish executions.

Algorithm 1 presents the confirmation of schedules for M concurrent workflows in cloud-edge computing. We input the workflow set, denoted as wf . U and V represent the set of scheduled computing tasks and the set of unscheduled computing tasks (Lines 2 and 3). If a computing task is the root of the remaining tasks in the same workflow, then the computing task is executed and we consider the M workflows simultaneously (Lines 5–10). Finally, the schedule times, and SKD are output.

Algorithm 1: Schedule confirmation of concurrent workflows

```
Require: wf
Ensure: SKD, S
1: s= 1
2: U = ∅
3: V= {vm,i| 1≤ i≤ |Vm|, 1≤ i ≤ M}
4: while ∅= V do
5: for vi∈ V do
6: if pre(vi) = ∅ then
7: U = U∪{vi}
8: V = V-{vi}
9: skds = skds∪ {vi}
10: end if
11: s = s+1
12: end for
13: end while
14: S = s
15: return S, SKD
```

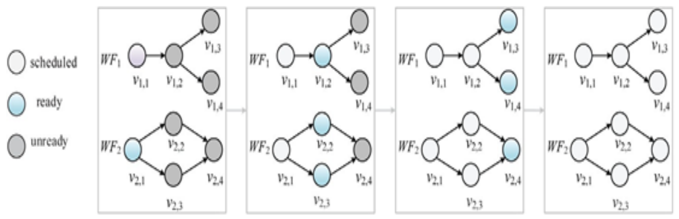

Fig -3: Dynamic schedules of two workflows with four computing tasks respectively

The computation tasks offloading method using NSGA-III based DE algorithm is described in section E.

4.5 Tasks offloading method

The computation tasks are offload from mobile devices to cloud environment to address the multi-objective optimization of the proposed system. The offloading method supports to the mobile device in reducing the congestion of execution queue of big data and it minimizes the energy consumption of computational resources. In other hand, the offloading process leads to increase the latency of transferring of data between cloud and mobile devices, to avoid this condition a small datacenter (cloudlet) is deployed between the network devices. Now we can offload the tasks either towards cloud or edge supported devices, it drastically reduces the execution time and energy consumption of mobile devices. The tasks are offloaded using the NSGA-III with DE algorithms.

4.6 Genetic Algorithm

Each computing task has a computation offloading strategy. In the genetic algorithm (GA), a gene represents the computation offloading strategy of a computing task and the genes compromise a chromosome, reflecting a hybrid computation offloading of the computing tasks in the same schedule. Fig.4 illustrates an example of computation offloading strategy encoding for the computing tasks in the first schedule. In this example, the chromosome is encoded in an array of integers (0, 1, 2).

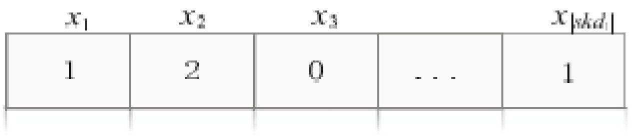

Fig -4: An encoding instance for the computing tasks in the first schedule

The Genetic algorithm contains mainly four operations of initialization, crossover, mutation and selection to create a new generation of children and the fitness values of newly generated offspring is compared with the parent fitness values, if it is better than the parent, then the parent is replaced with the new child.

4.7 Formulation

In this paper, we intend to shorten the execution time, given in (1) and save the energy consumption of each mobile device, presented in (2). The formalized problem is defined as

Let Tm(Xm) be the execution time of the mth workflow, which is calculated by

$$Tm(Xm) = TL(Xm) + Te(Xm) + Tt(Xm) \quad \text{--- (1)}$$

Let Em(Xm) be the energy consumption of themth mobile device, and we can get:

$$Em(Xm) = EL(Xm) + Ee(Xm) + Et(Xm) \quad \text{--- (2)}$$

$$\min Tm(Xm), Em(Xm), (\forall m \in \{1, 2, \dots, M\}) \text{--- (3)}$$

$$\text{s. t. } \sum_{m=1}^{M} \mu m \le C \quad \text{------- (4)}$$

$$\sum_{i=1}^{|Vm|} xm,i = \mu m(xm,i = 1, \forall m \in \{1, 2, \dots, M\}) \text{ --- (5)}$$

$$Tm(pre(xm,i)) \le T(pre(xm,i) + xm,i)(i \le |Vm|, m \le M) \text{ -- (6)}$$

In this problem, C represents the maximum number of virtual machines (VMs) the cloudlet can instantiate and μm represents the number of instantiated VMs for executing the mth workflow. The constraint presented in (4) describes that the aggregated computing resources of the instantiated VMs in a cloudlet are not over the computing capacity of the cloudlet. The constraint given in (5) indicates that each computing task offloaded to the cloudlet occupies one VM. The constraint elaborated in (6) ensures the precursor tasks of a computing task are implemented before the execution of it.

4.8 Fitness functions and constraints

The fitness functions are utilized to judge whether a possible solution is optimal in GA. A chromosome is the offloading strategies of all the computing tasks of the same schedule and each chromosome is an individual, representing a solution of the multi- objective optimization problem. The fitness functions include two categories, the execution time and energy consumption for each mobile device, presented in (1) and (2) respectively. The goal of the method is to find an optimal offloading strategy to minimize the two fitness functions for each mobile device. The fitness of a solution is to achieve the trade-offs between the 2M objectives.

In this method, we seek a hybrid offloading strategy of optimizing the execution time and the energy consumption for each mobile device. The constraints are given in (4), (5) and (6). NSGA-III performs well in addressing the optimization problem of multiple objectives with potential constraints. The execution time is one fitness function. Algorithm 2 elaborates how we evaluate the execution time. In this algorithm, we input SKD and the offloading strategies, denoted as χ. We first calculate the offloading latency, the computing time and the transmission time for executing a computing task in each schedule (Lines 3–12) and then the total time for executing a workflow (Line 13). Finally, the time for executing each workflow is output in each schedule.

Algorithm 2: Execution time evaluation

```
Require: SKD, χ
Ensure: Tm(Xm)
1: for s= 1 to S do
2: for m= 1 to M do
3: for i= 1 to |Vm| do
4: Calculate T L(xm,i) by (2)
5: Calculate T e(xm,i) by (3)
6: end for
7: Calculate T L(Xm) by (4)
8: Calculate T e(Xm) by (5)
9: for (vm,i, vm,j)∈ EDm do
10: Calculate T t (xm,i, xm,j) by (7)
11: end for
12: Calculate T t (Xm) by (8)
13: Tm(Xm) = T L(Xm)+ T e(Xm)+ T t (Xm)
14: end for
15: end for
16: return Tm(Xm)
```

The energy consumption of the mobile device is another fitness function. Algorithm 3 describes the process of evaluating the energy consumption. The inputs of the algorithm are the offloading latency, the computing time, the transmission time and the offloading strategies.

The offloading energy consumption, the computing and the transmission energy consumption in executing each computing task is first calculated (Lines 2–11) and then the total energy consumption of the mth mobile device in executing the workflow is obtained in each schedule (Line 12). Finally, the outputs of the algorithm is the energy consumption for each mobile device in each schedule.

Algorithm 3: Energy consumption evaluation for mobile devices

```
Require: T L(xm,i), T e(xm,i), T t (xm,i, xm,j), χ
Ensure: Em(Xm)
1: for m= 1 to M do
2: for i= 1 to |Vm| do
3: EL(xm,i) = T L(xm,i) · pI
4: Calculate Ee(xm,i) by (11)
5: end for
6: Calculate EL(Xm) by (12)
7: Calculate Ee(Xm) by (13)
8: for (vm,i, vm,j)∈ EDm do
9: Et (xm,i, xm,j) = T t (xm,i, xm,j) · pt
10: end for
11: Calculate Et (Xm) by (15)
12: Em(Xm) = EL(Xm)+ Ee(Xm)+ Et (Xm)
13: end for
14: return Em(Xm)
```

Initialization

In the subsection, the parameters of GA are determined, including the population size POP, the maximum of iteration I, the crossover possibility Pc , and the mutation possibility Pm. Each chromosome represents the computation offloading strategies of the computing tasks in the same schedule. In addition, let the gene cs,n be the offloading strategy of the nth computing task in the sth schedule. In the sth schedule, the chromosome is denoted as

Cs,i = (cs,1, cs,2, . . . , cs,N) (i= 1, 2, . . . , POP, N = |skds|).

Crossover and mutation

In this paper, the standard single-point crossover operation is conducted to combine two chromosomes and generate two new individuals. Fig. 5. shows an example of crossover operation for two chromosomes in the first schedule. In this example, a crossover point is first determined, and then swap the genes around this point to create two new chromosomes.

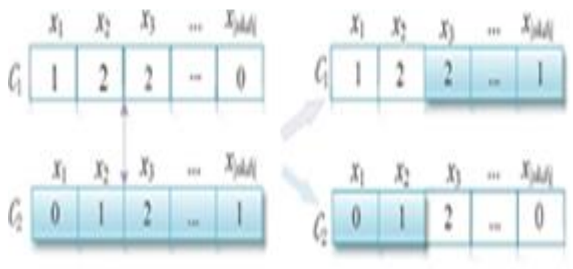

Fig -5: Crossover operations

The mutation is to modify genes of the chromosomes in the hope of generating individuals with higher fitness values. Fig.5. illustrates an example of the mutation operation in the first schedule. Each gene in a chromosome is changed with equal probability.

4.9 Selection for the next generation

In this phase, we aim at selecting the chromosomes for the next population to generate individuals with higher fitness values. Each chromosome represents a hybrid offloading strategy of the computing tasks in the same schedule. After crossover and mutation, the population size becomes 2POP. Algorithms 2 and 3 are used to evaluate the values of two fitness functions for each workflow in a schedule. The solutions are sorted according to the 2M values to generate several non-dominated fronts using the usual domination principle. In primary selection, we select one randomly from the solutions in the highest non-domination front each time to form the next generation until the number of the selected solutions is POP. Suppose the last added solution is in the lth non-domination front. If all the solutions in the lth front are included, then the selection finishes and the chosen solutions go into the next generation. In further selection, consider z solutions in the lth front are selected in primary selection. Then exclude the z solutions and further steps are conducted to make sure the z solutions in the lth front should be included in the next generation. We first normalize the 2M fitness values of each individual in the population. In the 2POP individuals, we search the minimum of the execution time and energy consumption for each mobile device, denoted as $T*m(Xm)(1 \le m \le M)$ and $E*m(Xm)(1 \le m \le M)$ respectively. Then the 2M values for 2POP individuals in the population are updated as

$$T'm(Xm) = Tm(Xm) - Tm*(Xm) \quad \text{-------} \quad (7)$$
$$E'm(Xm) = Em(Xm) - Em*(Xm) \quad \text{-------} \quad (8)$$

Let δmT and δmE represent the extreme values of the execution time and the energy consumption for the mth mobile device respectively, which are calculated by

$$\delta mT = \max \frac{T'm(Xm)}{WTm} \quad \text{--------} \quad (9)$$

$$\delta mE = \max \frac{E'm(Xm)}{WEm} \quad \text{-------} \quad (10)$$

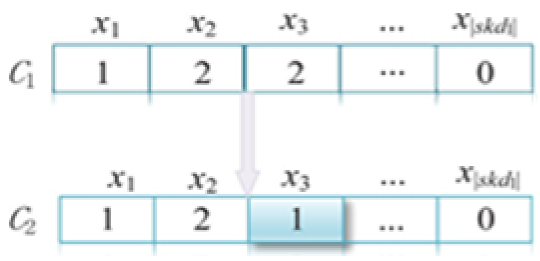

Fig -6: Mutation operation

WTm and WEm in (9) (10) are the weight vectors of the two functions.

We consider each fitness function as an axis. In the hyperplane compromised by the 2M axes, the intercept of each axis is determined, denoted as αmT and αmE respectively for the mth workflow. Then the 2M fitness values of each individual in the population are normalized as:

$$T''m(Xm) = \frac{T'm(Xm)}{\alpha mT} \quad \text{-------} \quad (11)$$

$$E''m(Xm) = \frac{E'}{\alpha mE} \quad \text{-------} \quad (12)$$

After the normalization, the values of the execution time and the energy consumption for each workflow are in the domain [0,1). The solutions in the population has compromised a 2M-dimensional hyperplane. Then the normalized solutions are associated with reference points. A set of reference points are scattered in the 2Mdimensional hyperplane. The intercept of each axis is 1 and each of the axis is divided into g subsections. Then the number of the reference points, represented by θ, is calculated by

$$\theta = Cg2M + g-1 \quad \text{----------} \quad (13)$$

θ is approximately equal to the population size POP to make sure each normalized solution associates with one reference point nearly. Sort the solutions in the lth non-dominated front, according to the number of the reference points they associate with. Each time select one randomly from the solutions with maximum number of associated reference points. This process is repeated until all the z solutions have been selected. In this algorithm, we input the tth generation population (parent population) denoted as PPt and the reference point set, denoted as R. The output is the (t + 1)th generation population (child population) PPt+1.

In this algorithm, we first calculate the execution time and the energy consumption of mobile devices by the Algorithms 2 and 3 respectively (Lines 2 and 3). Non-dominant sorting is conducted for individuals in the population through the usual domination principle (Line 5). Furthermore, we select the solutions primarily and judge whether all the solutions in the lth front are included (Line 6). If not, we conduct further selection to determine the remaining z solutions in the lth front for the next generation (Lines 8–12). Based on the selection algorithm, the POP solutions going into the next generation are selected.

4.10 Combination of NSGA-III and DE algorithm

All the operations of Genetic Algorithm are repeated for Differential Evolutionary algorithm in reciprocal order and the same operations iterated for number of times to reach an optimized fitness value. Fig. 7 flow chart shows that the implementation of NSGA-III and DE algorithms to proposed system.

5. EXPERIMENTAL RESULTS AND ANALYSIS

The proposed system requires a minimum configuration of hardware and software to experiment and evaluate the performance of proposals.

The proposed systems are modelled and experimented in the below mentioned minimum configurations of physical computational resources.

Processor-Intel(R) Pentium(R)CPU B950 @ 2.4GHz

Memory	-	4GB
Hard disk	-	200 GB
Mouse	-	Logitech
Monitor	-	15" VGA Color

The specified software is required to design the proposed model for implementing the physical frameworks and algorithms. The experimental environment can be modelled either under Windows or ubuntu (open source) OS and WorkflowSim which is extended from CloudSim as a testing toolkit [4]. Both the tools have been developed in Java programming languages and the user interfacing screens are designed using Java Swing. NetBeans is preferred as an integrated development environment and MySQL uses to store experimented values for further references. The below mentioned Directed Acyclic Graph workloads are required to test the proposed system.

1. CyberShake_30.xml
2. CyberShake_50.xml
3. CyberShake_100.xml

In this model, the resultant values are stored in a database's tables for further references. Moreover, the performance of the proposed system is evaluated by plotting the charts and graphs between different metrics and conclude the efficiency of the proposed model.

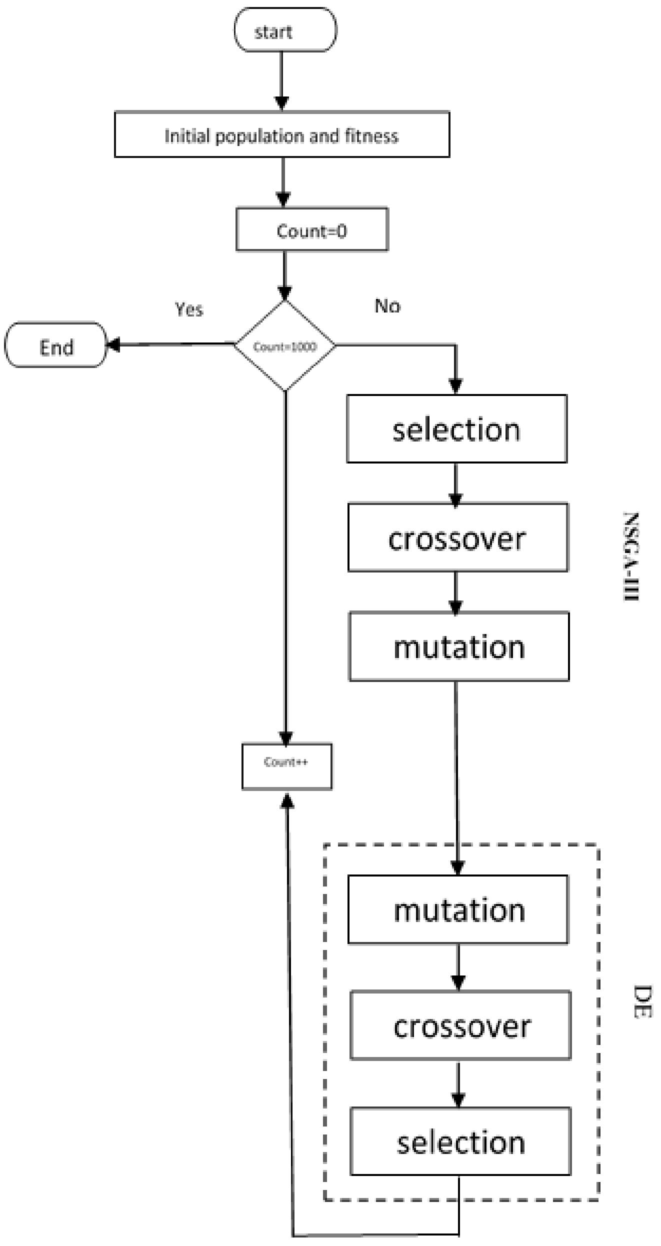

Fig -7: Flow chart of proposed algorithm

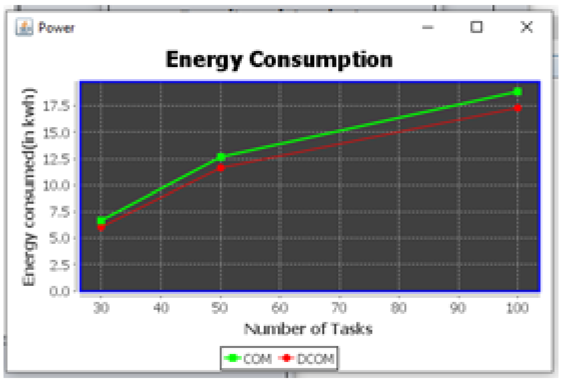

Fig -8: Energy consumption

In Fig. 8, A graph is drawn between the energy consumption on Y-axis and workloads on X-axis, it shows that the proposed model performing better than the existing system in reducing the emission of CO2 and it leads to create a green cloud environment.

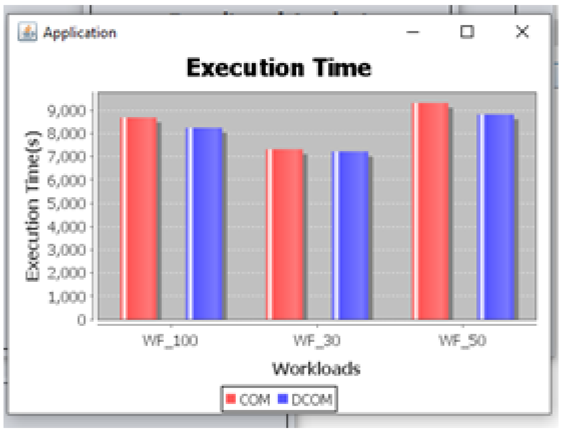

Fig -9: Execution time chart

In Figure 9, another chart is drawn between execution time and workloads and it depicts that the DCOM can act as an efficient and time-intensive response to the mobile user's requests.

6. CONCLUSIONS AND FUTURE ENHANCEMENTS

The proposed method DCOM (DAG Computational task Offloading Method) for IoT mobile application which is running on a mobile device and uses the techniques to offload computation tasks and analysed the proposed dynamic scheduling algorithms NSGA-III with DE to perform the multi-objective optimization problems. Furthermore, extensive experiments and evaluations are conducted using WorkflowSim toolkit to encourage the proposed method DCOM performs well in solving the optimization problem. For future, the proposed technique can be enhanced to work with some other objectives such as fault tolerance and overhead delays. In Internet of Mobile things, trade-off between delay and power consumption is an open research area. Further, the proposed technique will be implemented dynamic migrations of computation tasks over the edge supported cloud environment for efficient resource utilization.

REFERENCES

[1] M.V. Barbera, S. Kosta, A. Mei, J. Stefa, To offload or not to offload? The bandwidth and energy costs of mobile cloud computing, in: INFOCOM, 2013 Proceedings IEEE, 2013, pp.1285–1293.

[2] A. Jin, W. Song, W. Zhuang, Auction-based resource allocation for sharing cloudlets in mobile cloud computing, IEEE Trans. Emerg. Top. Comput. 6 (1) (2018) 45–57.

[3] T.Q. Dinh, J. Tang, Q.D. La, T.Q. Quek, Offloading in mobile edge computing: task allocation and computational frequency scaling, IEEE Trans. Commun. 65 (8) (2017) 3571–3584.

[4] Calheiros, R.N. , Ranjan, R. , Beloglazov, A. , De Rose, C.A.F. , Buyya, R. , 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exper. 41 (1), 23–50.

[5] C. Wu, E. Zapevalova, Y. Chen, F. Li, Time optimization of multiple knowledge transfers in the big data environment, Comput. Mater. Continua 54 (3) (2018) 269–285.

[6] Mashayekhy, L., Nejad, M. M., Grosu, D., & Vasilakos, A. V. (2016). An online mechanism for resource allocation and pricing in clouds. IEEE Transactions on Computers, 65, 1172–1184.

[7] Rossi, Francesca, Peter van Beek and Toby Walsh. 2006. Handbook of Constraint Programming (Foundations of Artificial Intelligence). Elsevier Science Inc.

[8] Schrijver, Alexander. 1986. Theory of linear and integer programming. John Wiley & Sons, Inc., 471 pages.

[9] Beloglazov, Anton. 2013. "Energy-Efficient Management of Virtual Machines in Data Centers for Cloud Computing". THE UNIVERSITY OF MELBOURNE.

[10] J Chenni Kumaran, M Aramudhan, A survey on resource allocation strategies in cloud, International Journal of Reasoning-based Intelligent Systems, Vol. 10, Nos. 3/4, 2018, 328-336.

[11] S. Iniyan, M. Senthilraja, R.Srinivasan, A.Palaniraj, Survey on security threats in mobile cloud computing, International Journal of Engineering & Technology, Vol. 7, No. 1.9 (2018), 238-241.

BIOGRAPHIES

Baskar M working as Associate Consultant in Slogix (OPC) Pvt. Ltd. at Chennai and is having 7 plus years of industry experience and 11+ years of academic experience. My specialized working areas are Cloud Computing, IoT Cloud, Mobile Computing, and

Networking. More than 10 papers has published in International and National Level Journals and Conferences.

Dr.J.Chenni Kumaran working as Professor in IT Department of Panimalar Institute of Technology and having Twenty Two plus years of experience. So far 17 papers has published in International and National Level Journals and Conferences. Awarded UTHAMA ACHARYA PURASKAR(BEST FACULTY AWARD) supported by Indian Servers, Lion Club Internations, Telangana IT Association, Andhra IT Companies Association & Impact foundation. My specialized areas are Cloud Computing, Big Data Computing, Web Technologies, SOA, and OOAD.