Efficient Welfare Maximization in Fog-Egde Computing Environment

***Abstract*— A Fog network comprises a group of peer nodes where all the peer nodes take part in a collaborative network for sharing computing and storage resources. In a collaborative FOG network, the major challenge for a task node is to efficiently select and offload the tasks to helper nodes. The collaborative scheme in general works with a reasonable incentive policy for sharing load of other or helping others where revenue and profit involves in task execution.**

We formulate the problem of welfare maximization which is profit maximization and minimization of disparity in profit between sharer nodes, and formulate the said problem in integer linear program for understanding the problem clearly and solved the using standard LP solver. We propose three welfare scheme heuristics for load sharing considering the local and centralized solution approaches, and compared the performance of these for data sets. The proposed welfare schemes uses task selection for load sharing which consider by properly balancing the effects of executing time, revenue and data size of the task for selection in sharing the tasks with others for welfare maximization. We heuristically calculate weight of the task parameters in selecting task for sharing with other sharer nodes. Based on the experimental result, the propose weight based task selection in load sharing achieves on an average of 96.23% profit as compared to optimal amount of total profit using ILP based profit maximization. Also, weight based local reservation scheme for load sharing achieve significantly better result in both maximizing total profit and minimizing the disparity in profits of the sharer nodes.

***Index Terms*— FOG Network, Egde, Profit, Welfare, Incentive Model, Load Sharing**

I. Introduction

It is a well-known fact that nowadays mobile device become power full in terms of compute capacity, data storage, battery storage and network connection as compared to older generation of mobile devices. Also, we are inseparable from mobile phones due to increased large scale network and effective networking speed. Improved network infrastructure combined with user demand of compute and date intensive applications like augmented reality applications, virtual reality applications, highly advanced gaming applications, human-computer interaction services, and online streaming are on the roll and we are getting fascinated by all the latest and trending services.

In reality the demand for executing these modern compute intensive application require very high amount of compute and storage requirement, which is most of the time is not available in mobile device. The advanced mobile application seeks intensive computation resources for the task to get completed. The demand for intensive computation resources in resource limited devices leads to high energy consumption cost [1], [2].

So it becomes a vital responsibility to perform not only efficiently but also energy-efficiently execution of task when it comes to mobile devices. Task offloading offers an interesting and efficient solution that has been studied and worked upon. Task offloading has been one of the solutions in the industry and academy level. In mobile cloud computing [3], which offers a high resource enriched remote cloud environment where mobile devices with limited resources can offload tasks requiring high computation. Many commercial and cloud services such as AWS, Windows Azure are providing mobile cloud computing paradigm but sometimes the delay between mobile devices and cloud services becomes noticeable.

Researchers have emerged with a new paradigm named Mobile Edge Computing which offers leverages over standard mobile cloud computing providing collaboration among end-users such as smart phones and/or network edge facilities ranging from base stations to network providers. The concept of mobile edge computing also known as FOG computing is implemented at the end-user edge so that mobile users can have both low-latency and intensive computing resources. Also the nearby mobile devices may be share the load of the compute demand of any mobile device, this concept is called device to device (D2D) fogging.

Let us take an example to understand the D2D fogging (E-E sharing) scenario as shown in Figure 1. The user task load is shared among all the sharer or helper nodes. At a particular time slot or interval, some of the nodes may be act as helper node based on their own compute load and other node act as non-helper node at that time slot. Suppose a user mobile device A (the task node) has been assigned to execute a computation-intensive task and the computing capacity of user device A is already being utilized by other application, in this scenario, a helper node H from the set of available helper node is chosen for the task to be offloaded on helper node H and collaboratively share the load in FOG network, this kind of network generally termed as federated fog computing [4].

If a task node is unable to find any sharer than it upload the task to higher level of remote compute infrastructure. The sharing of load can be possible in device to device level (D2D fogging), the same set of D2D device may be connected to edge devices and many edge devices (servers) may use load sharing policy among them. Similar to D2D fogging, in the edge server level if the task or work is not shareable by any

other egde server helper then the same task get uploaded to remote cloud servers. Clearly there are three level where the task can be executed and these are (a) device level on any one of the task node or helper node, (b) edge server level on any task originated edge server or helper edge servers, and (c) remote cloud level.

This FOG network strongly depends on user collaboration, the collaborative scheme in general works with a reasonable incentive policy for sharing load of other or helping others. This incentive mechanism motivate users to collaborate in sharing computation resources and can extinguish free-riding behaviors and over-exploiting that can hamper the user's desire for collaboration. In such case, we safely assume, every incoming task user task to the mobile device (or node) comes with revenue in executing the task along with task compute size and data size.

So in this work, we propose load sharing policy in FoG network whose objective is to maximize the profit of each individual node and maximize the sum of profit of all the nodes such the overall welfare is maximized, which means overall profit is maximized and overall difference between profit between nodes is minimized with the computationally efficient task sharing or offloading policy.

Certain assumptions are made while designing task sharing framework. We assume that user mobility is random so it can not be predicted hence the framework works on the current system information only. Sufficient bandwidth is available to offload the task from one node to another nodes but depends on the delay bandwidth available between the nodes. A good task D2D (E2E) fogging load sharing framework should have the capability to achieve optimal objective when it comes to the execution of all the tasks. As the users demand (or number of tasks) grow at a rapid pace, our sharing decision complexity should not be compute intensive and should be proportional to the polynomial function of number of tasks in the system.

The major contributions of work is listed as follows

- We formulate the problem of profit maximization which is welfare maximization and minimization of disparity of profit between sharer nodes.
- We formulate the said problem in integer linear program for understanding the concept clearly and solved the using standard LP solver to find out the optimal solution for comparision purpose.
- We propose three welfare scheme heuristics for load sharing considering the local and centralized solution approaches, and compared the performance of these for data sets.
- Load sharing scheme considering properly balancing the executing time, revenue and data size of the task have significant impact in task selection in sharing with others. We propose to heuristically calculate weight of the task parameters in selecting task for sharing with other sharer nodes.

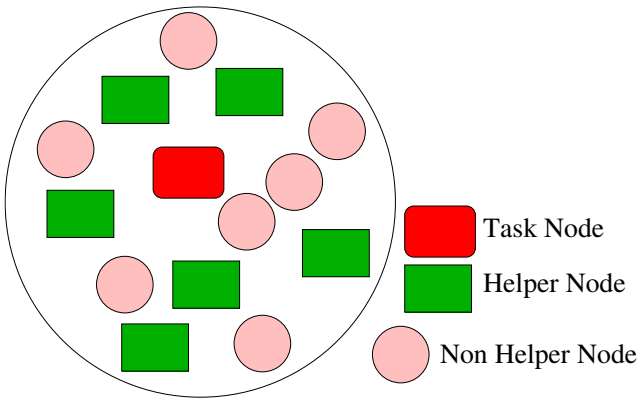

Fig. 1. **Considered D2D Fogging Architecture**

Rest of the paper is organized as follows. In section II we describe related work of the general fog network and profit maximization schemes. Section III we describe the system model, task model, energy consumption and profit model, and define the problem definition. We represent the problem in integer linear program and proposed four heuristics to solve the problem in section IV. In Section V, we describe the experimental evaluation and performance of the proposed approaches using real life simulation environment. We finally conclude our work and discuss possible future extension of the work in section VI.

II. Related Work

We can roughly categorize already existing research on Fog network into two categories [5], [6] and these are Fog network architecture and revenue-profit optimization. In this section, we focus mostly on profit maximation and welfare maximization aspect of the edge or fog computing platforms.

However, there are two prerequisites for the success of the user collaboration based architecture. First, the user mobility such as duration time at a place and contact behavior should be predictable. Second, the collaborated users should trust each other or an incentive mechanism should be provided. Most of existing researches leverage user social relationship to fulfill these requirements, while they may restrict the services in social communities and lose a mass of opportunities for collaboration with nearby strangers. In addition, they normally focus on single-user level optimization rather than the network-wide optimization, and ignore the device energy consumption for predicting user mobility and making offloading decision.

In [9], Jonathan et al. described profit maximization model for task assignment problem in 2-tier fog/cloud system but they consider only profit maximzation issue not the disparity in the profit of the nodes. Similarly in [10], Anglano et al. proposed methods for profit-aware coalition formation in fog computing providers in federated fog nodes using a game-theoretic approach, but here also, they consider only profit maximization of the profit of individual node but not as a

TABLE I
NOTATIONS AND THEIR MEANINGS

Notation	Definition
M	Total number of nodes
N	Total number of tasks
$\mathbf{T}$	Set of tasks
$\mathbf{V}$	Set of nodes
V_j	Represent the j^{th} node
t_i	Represent the i^{th} task
d_i	Deadline of the i^{th} task
l_i^c	Compute size of i^{th} task in MI
l_i^s	Data size of i^{th} task
r_i	Revenue associated with the i^{th} task
O_i	Originating node for the i^{th} task
$D_{i,j}$	Distance between nodes V_i and V_j
c_i	Number of cores in node V_i
cp_i	Computing capacity per core in MIPS of node V_i
ecr_i^{proc}	Processing energy consumption rate of core at node V_i
ecr_i^{comm}	Communication energy consumption rate of core at node V_i
ρ_i	Energy cost (in $ per Joule) at node V_i
$Profit_j$	Profit of the node V_j

whole fog system. In [7], Yin et al. proposed approach for distributed resource sharing in fog-assisted big data streaming application, where there focus to maximize the pay of individual nodes by distributed coordination manner.

In [11], Cosimo et al. proposed a profit-aware resource management for approach edge computing systems where they consider dynamically resource management for online profit maximization using ILP in a given area for egde system but not sharing properly, and their method is not scalable to bigger task system as they use compute intensive ILP. In [8], Kumar et al. propose different pricing mechanism and workload scheduling to optimize social welfare and cost for fog computing systems, but in their approach do not consider overall profit maximization but considers only the profit disparity among node.

We argue that these two architectures can complement each other, which motivates our design of the D2D Fogging framework to take advantage of both architectures. To the best of our knowledge, the D2D Fogging framework is the first solution which jointly considers user incentive and collaboration into the network-wide task offloading optimization for profit maximization and disparity minimization, meanwhile providing corresponding task scheduling policies in terms of various system setting.

III. System Model and Problem Statement

A. System Model

We assume a FOG network environment primarily consisting of M nodes $\mathbf{V}$=$\{V_1, V_2, \cdots, V_M\}$ where all the M nodes share computing resources with each other in collaborative way. Bandwidth between the FoG node V_i and V_j is proportional to $\frac{1}{D_{i,j}}$, where $D_{i,j}$ is Euclidean distance between the nodes V_i and V_j. Every FOG node V_j is characterized number of cores c_j (consisting of same computing capacity) , processing energy consumption rate per core ecr_j^{proc}, communication energy consumption rate ecr_j^{comm}, computing capacity per core in MIPS cp_j, energy cost per unit energy consumption ρ_j.

B. Task Model

Let T = $\{t_1, t_2, t_3, \cdots, t_N\}$ be set of N tasks where each task t_i has a structure comprising of attributes associated with it such as task originating node (O_i), deadline (d_i), task compute size (l_i^c) in million of instructions (MI), task data i/o size (l_i^s) in KB and revenue associated with that task (r_i). So the task can be represented in tuple form $t_i(l_i^c, l_i^d, d_i, O_i, r_i)$. The value is O_i is from set $\{1$ to $M\}$, this specify the where the task is originated or created.

For the sake of simplicity, we consider the task arrived to the system in interval wise and in a time interval all the tasks arrived (at the begining of the interval) to the system are considered for execution and deadline of all the tasks of the interval is end time of the interval. Between two consecutive time intervals, all the tasks and machine environment changes and there is no backlog task or node execution after the interval. The profit maximization and welfare maximization for all the individual intervals improve the final result.

C. Task Execution Model, Cost Model and Profit Model

Effective time to the execute task t_i at originated node V_j is given by

$$ET_{i,j} = \left(\frac{l_i^c}{cp_j}\right) \tag{1}$$

Effective time to the execute task t_i on helper node V_j is given by

$$ET_{i,j} = \left(\frac{l_i^c}{cp_j} + \frac{l_i^s}{bw_{i,j}}\right) \tag{2}$$

Cost of execution of task t_i, originated at FoG node V_j can be given as

$$Cost_{i,j} = \left(\frac{l_i^c}{cp_j} * ecr_j^{proc}\right) * \rho_j \tag{3}$$

Cost of execution of task t_i at FoG helper node V_j can be given as

$$Cost_{i,j} = \left(\frac{l_i^c}{cp_j} * ecr_j^{proc} + \frac{l_i^s}{bw_{i,j}} ecr_j^{comm}\right) * \rho_j \tag{4}$$

If a task t_i is executed at the originated node V_j then the revenue $R_{i,j}$ earn at node V_j same as revenue associated with the task and which is $R_{i,j} = r_i$. But if a task t_i originated at V_k is executed at the helper node V_j then the revenue at earn by the helper node V_j can be calculated as $R_{i,j} = r_i.\lambda$ and the revenue at originated node V_k can be calculated as $R_{i,k} = r_i.(1-\lambda)$, where typical value of λ is around 0.9.

As the revenue of a task is shared by originating node and helper node when the task is executed at the helper node and we represent the profit associated with helper node execution mode using the term U^y. The profit of node V_j in executing the task t_i originated at node V_k on helper node V_j is

$$U^y_{ij} = R_{i,j} = r_i.(1-\lambda), \tag{5}$$

and profit the node where the task is originated is given by

$$U^y_{ik} = R_{i,k} - Cost_{i,k} = r_i.\lambda - Cost_{i,k}. \tag{6}$$

When the task is originated at node is executed on the same node, we represent the profit associated that using the term U^x. So the profit of node $V_{j=O_i}$ in executing the task t_i on node V_j in case of task is executed in originated node is given as

$$U^x_{ij} = r_i - Cost_{i,j} \tag{7}$$

The profit of a node V_j in executing the tasks assigned to node is summation of (a) profit in execution of all the originated task at same node on the same node, (b) profit by executing originated task at V_j on other helper node and (c) profit by executing others tasks by acting as helper node and this given by

$$Profit_j = \sum_{i=1}^{N} (U^x_{ij} + U^y_{ij}) \tag{8}$$

where the U^x_{ij} is associated with profit mentioned in part(a) and U^y_{ij} is associated profit mentioned in part (b) and (c).

Total profit of the system can be calculated as sum of profit of all the nodes and which is given by

$$TP = \sum_{j=1}^{M} Profit_j. \tag{9}$$

The disparity in the profit can be calculated as difference between the maximum profit among all the nodes and the minimum profit among all the nodes and this can be represented as

$$DP = \underset{j}{\text{argmax}}\{Profit_j\} - \underset{j}{\text{argmin}}\{Profit_j\} \tag{10}$$

D. Problem Statement

We want to schedule the tasks on to the nodes of the system such that these two objectives are fulfilled.

- Maximize the value of TP as defined in the equation 9
- Minimize the value of DP as defined in the equation 10

IV. Solution Approaches

In this section, we will define the solution approaches to solve the problem of profit maximization in FOG Network

A. Integer Linear Programming Based Approach

The problem defined above can be solved using ILP (Integer Linear Programming) formulation. We define two types of binary variables X_{ij}, Y_{ijk}. The value of $X_{ij} = 1$ if the task t_i is originated at node V_j and executed at the same node V_j otherwise $X_{ij} = 0$. Similarly the value of $Y_{ijk} = 1$ if the task t_i is originated at node V_j and executed at node V_k otherwise $Y_{ijk} = 0$. Now we focus on ILP formulation to maximize overall profit in FOG network with deadline constraints satisfied.

The objective of the ILP is,

$$max\left(\sum_{i=1}^{N}\sum_{j=1}^{M} X_{ij}U^x_{ij} + \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{\substack{k=1\\k!=j}}^{M} Y_{ijk}\left(U^y_{ij} + U^y_{ik}\right)\right) \tag{11}$$

Subject To:

$$X_{ij} + \sum_{\substack{k=1\\k!=j}}^{k=M} Y_{ijk} \leq 1 \tag{12}$$

The above constraint is valid for each task t_i originated at node V_j. This constraint makes sure that a task t_i gets executed on one node either V_j or V_k.

The execution time of each task t_i should be less than the deadline

$$\sum_{j=1}^{M} ET_{ij}.X_{ij} + \sum_{\substack{k=1\\k!=j}}^{M} Y_{ikj} * ET_{ik} \leq ID \quad \forall i \tag{13}$$

For all the node (or machine) V_j the capacity constraints should be meet

$$\sum_{i=1}^{N} X_{ij} * \left(\frac{l^c_i}{ET_{ij}}\right) + \sum_{\substack{k=1\\k!=j}}^{M} Y_{ijk} * \left(\frac{l^c_i}{ET_{ij}}\right) \leq ID * cp_j \; \forall j \tag{14}$$

B. Global pool based centralized scheduling approach

Global Pool based scheduling approach is a greedy based where all the tasks from all the nodes are submitted to a centralized global queue and then scheduled by a centralized scheduler. The global scheduler uses information like distance matrix between nodes, data size of tasks, compute size and revenue of tasks to schedule tasks to approximate nodes to optimize both the total profit T_P maximization and minimization of the difference in profits among the nodes which is D_P. The main target of the global scheduler is maximize the profit by executing a most profitable task on the available node where it can contribute maximum to the total profit. Pseudo code of

Algorithm 1 Global Pool based scheduling

Require: List of tasks T, List of nodes V
Ensure: Total profit T_P and difference in profit D_P

Initialize the global pool of task: $GPList \leftarrow \phi$
for $j \leftarrow 1$ to M **do**
    Add all the tasks originated at node V_j to $GPList$.
Sort all the task of $GPList$ based non increasing order of $\frac{r_i}{l_i^c}$ of the task
for each task $i \leftarrow 1$ to N from $GPList$ **do**
    Choose the node V_j for the task t_i where profit is maximized for the task and task is feasible to schedule on that node. The profit can be calculated using equations 5, 6 and 7
    Update the load of that node V_j.
    if (**then**no node is available for the task to execute)
        Upload the task to next level remote cloud/edge server for execution and cost to be borne by the originated node in uploading the task.
for $j \leftarrow 1$ to M **do**
    $T_p \leftarrow T_p+$ profit_of(V_j)
Calculate D_P using Equation 10.
return T_P and D_P .

Algorithm 2 Local reservation based scheduling

Require: List of tasks T, List of nodes V, Selection criteria
Ensure: Total profit T_P and difference in profit D_P

Initialize the global pool of task: $GPList \leftarrow \phi$
for $j \leftarrow 1$ to M **do**
    Sort all the tasks of the node V_j in the input selection criteria.
    Reserve κ percentage of tasks for local execution
    Check if all the reserve task can be executed in local node in selection order V_j, if some tasks can not be executed due to capacity constraint of the node then send the excess task to the global pool $GPList$
    Add the remaining unreserve tasks of the node to the global pool $GPList$
Schedule the tasks of $GPList$ using global pool based scheduling given Algorithm 1
for $j \leftarrow 1$ to M **do**
    $T_p \leftarrow T_p+$ profit_of(V_j)
Calculate D_P using Equation 10.
return T_P and D_P .

global pool based scheduling in Algorithm 1. The algorithm initialize by adding all the tasks from all the node to the list of global pool of task. After that it takes the task t_i one by one in the order of higher expected profit (higher value of $\frac{r_i}{l_i^c}$) and choose the node V_j for the task t_i where profit is maximized for the task and task is feasible to schedule on that node.

This centralized approach is bound to achieve higher profit but it perform in general badly due to local node have no control in holding their own tasks execution on their own place. So this approach is not good for balancing profit. But the global scheduler can be modified to balance the profit simultaneously.

In the next three approach each node can reserve some quota (κ) percentage (say upto 70%) of local tasks to be executed on the local node if they can execute the reserved task. If capacity exceed for node in executing the reserved task then the node put the extra excess tasks to global pool for sharing with other nodes. Clearly their can be many policies for selection of local tasks to be reserved for local execution and these policies can be based on first come first serve (FCFS) , revenue and profit. Once the task send to the global pool the local node do not have any control on that task execution and where that task will be executed. This the policy of global task scheduling is solely on profit maximization based as shown in Algorithm 1.

C. FCFS based approach

In this case, the task selection criteria for selection in to local queue from the locally originated task is based on first come first serve basis. The first κ percentage of task get reserved for the local execution and rest of the task are added to the global pool. Also, if the all the reserved tasks cannot be executed on the local node then some of the reserved tasks get added to the global pool such that all the remaining locally reserved task should be able to execute without any capacity constraints. The tasks in the global list are scheduled based on global scheduling policy as described in Algorithm 1.

D. Revenue based heuristic approach

The main demerit of the FCFS based reservation policy is that FCFS policy do not consider the task revenue and the execution time. In this case, the task selection criteria for selection in to local queue from the locally originated task is based on amount of revenue they contribute in execution of the task. Higher the revenue of the task better chances to be selected for the local execution. The top κ percentage of highest revenue providing tasks get reserved for the local execution and rest of the task are added to the global pool. Similar to FCFS approach, in this case also, if the all the reserved tasks cannot be executed on the local node then some of the reserved tasks get added to the global pool such that all the remaining locally reserved task should be able to execute without any capacity constraints. The tasks in the global list are scheduled based on global scheduling policy as described in Algorithm 1.

E. Weight based heuristic approach

The high revenue based task selection may not gureentee the highly profitable task execution. Tasks with high revenue and lower task compute length contribute higher amount to the profit. So selecting the task with higher revenue to task compute length for local quota definitely increase the profit

of the local node. But this policy may put high data intensive task to global pool which may degrade the profit of the overall system that is total profit. Hence, there should be a balance in task selection. Based on this fact, we propose weight based task selection criteria for selecting tasks for local execution. The proposed weight of task can be computed as

$$w_i = \alpha * (\frac{r_i}{l_i^c}) + (1 - \alpha) * l_i^s \quad (15)$$

whereα is a linear trade-off factor between the profit of the task and data size of the task. The value of α is between $[0to1]$. Higher the value of α gives priority to profit and less to the data size of the task.

The value of α have significant impact on the profit and difference in the profit among all the nodes. Similar to earlier approaches, in this case also, The top κ percentage of highest weight tasks get reserved for the local execution and rest of the task are added to the global pool. In this case also, if the all the reserved tasks cannot be executed on the local node then some of the reserved tasks get added to the global pool such that all the remaining locally reserved task should be able to execute without any capacity constraints. The tasks in the global list are scheduled based on global scheduling policy as described in Algorithm 1.

V. Experiments and Results

For our experiment, we created an simulation environment in C++ and Python similar to iFogSim [12] and CloudSim [13]. The ILP approach was implemented using IBM CPLEX optimizer on Ubuntu 18.04 platform and integrated with our simulation environment.. This section describes the data set used for experiment to evaluate all the four proposed heuristic approaches mentioned in previous section and compared the same with the standard ILP approach.

A. Parameters Used

For our experiment, we use the following configuration parameters: the number of machines/nodes is set to $M = 16$. The weight factor for Algorithm 2 (Local reservation based scheduling Algorithm with weight based selection) with $\alpha = 0.8$ and the interval duration ID $= 5$ and the value of total intervals is set to TI $= 100$. The percentage of task in local pool = $\kappa = 70$ % and the number task set is TS $= 8$. The value of $\kappa = 70\%$ is set based on experiment, it achieves better overall profit. But the percentage is maximum limit but if the node is overloaded then the load upload most of the tasks to the global pool of task for centralized scheduling.

Node/Machine parameters are as follows- the total cores per node $totalcores \in [10, 16]$ (consisting of same computing capacity. The compute capacity per core (are specified in millions instruction per second or MIPS) $cp \in [2000, 2500]$. Processing energy consumption rate per core $ecr^{proc} \in [10, 15]$. Communication energy consumption rate per core $ecr^{comm} \in [6, 8]$. Energy cost per unit energy consumption $\rho \in [5, 10]$. Euclidean distance between any two nodes i and j is $D_{i,j}$ chosen randomly $\in [1, 200]$.

Data ↓	Weight	Revenue	FCFS	Global
TS1	1102439.49	1048969.00	976612.50	1103766.28
TS2	1033312.15	1013827.00	947928.37	1026483.24
TS3	1020055.18	975548.00	873379.21	1014649.15
TS4	992032.32	961450.00	869320.03	987607.79
TS5	916523.26	933171.00	871022.25	921809.46
TS6	1014464.84	972824.00	861531.27	1006695.48
TS7	945354.31	935853.00	845895.96	944742.98
TS8	930768.93	915263.00	890791.89	924452.17

TABLE II
Comparison of T_P for different solution approaches

Taskset parameters are as follows - the total number of tasks in task set is chosen $\in [100, 250]$. The originating node of each task $\in [1, M]$. Deadline is same for every task which belongs to same time interval (ID value) and the deadline is set to the end time of Interval, which means all the task of the interval have common deadline. The total number of tasks belonging to a particular data set is divided into four equal parts. These parts are

- First part contains tasks having low revenue r_i and high task computing size (in MI) l_i^c where $r_i \in [200, 400]$ and $l_i^c \in [500, 1000]$.
- Second part contains tasks having high revenue r_i and high task computing size (in MI) l_i^c where $r_i \in [1000, 2000]$ and $l_i^c \in [500, 1000]$.
- Third part contains tasks having low revenue r_i and low task computing size (in MI) l_i^c where $r_i \in [200, 400]$ and $l_i^c \in [100, 200]$.
- Fourth part contains tasks having high revenue r_i and low task computing size (in MI) l_i^c where $r_i \in [1000, 2000]$ and $l_i^c \in [100, 200]$.

B. Results and Comparison

For our experiment we considered tasks with good mixture variation of both revenue r_i and task computing size l_i^c.

Now we run the ILP and all the four heuristic algorithms over the above data set and compare the results. In this section, we have compared total profit T_P and difference in profit D_P retrieved from all algorithms over eight datasets.

Figure 2 and Table II shows the comparison of total profit T_P for all the four heuristic approaches on eight data sets. The value of α was set to 0.8 for the weight based heuristics. Higher value of TP is better and preferable. The weight based and global pool based approaches performs better than

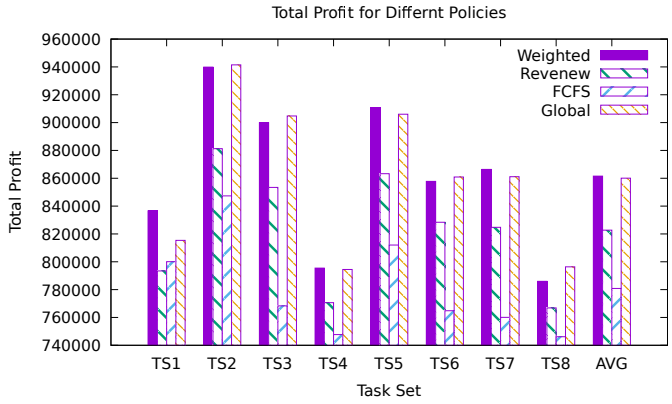

Fig. 2. **Comparison of T_P for different solution approaches.**

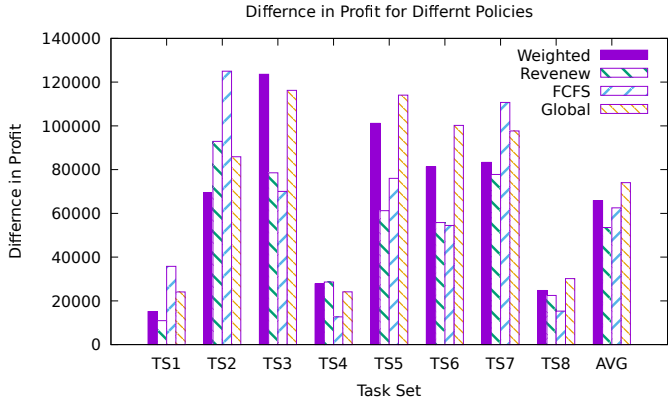

Fig. 3. **Comparison of D_P for different solution approaches.**

revenue based and FCFS based approaches. For six out of eight data sets, weight based scheduling approach produces higher total profit T_P as compared to the revenue based scheduling approach.

Figure 3 shows the comparison of difference in profit (DP=Max Profit - Min Profit) for all the four heuristic approaches for the considered eight data sets. The lower value of DP is preferable and lower the value of DP signifies the profit distribution among all the nodes is fair. For most of the tasks set, the weight based heuristic performs better than global pool based scheduling in distributing the profit among all the node as both the global pool based and weight based heuristics produces significantly higher total profit as compared to the other two approaches.

Figure 4 is a line graph where the total profit T_P is calculated and plot for all the eight data sets for different value of α ranging between $[0, 1]$ for weight based scheduling approach. We can see that as the value of α increases, the total profit T_P increase for all the eight data sets. The reason behind this is that, the α is multiplied to $\frac{r_i}{l_i^c}$ which is revenue to compute size ration of the task, so as α increases, task with more revenue and less task computing size is given higher weight, hence the total profit T_P increases with α. But for most of the time value of profit get saturated or decrease after

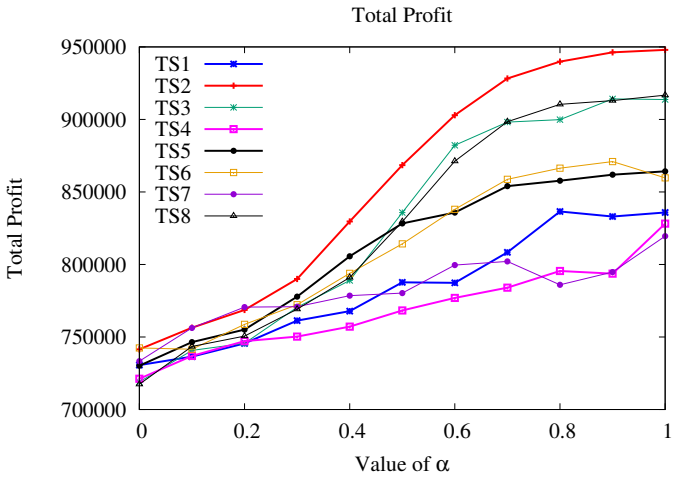

Fig. 4. **T_P for Weight based approach over weight factor α**

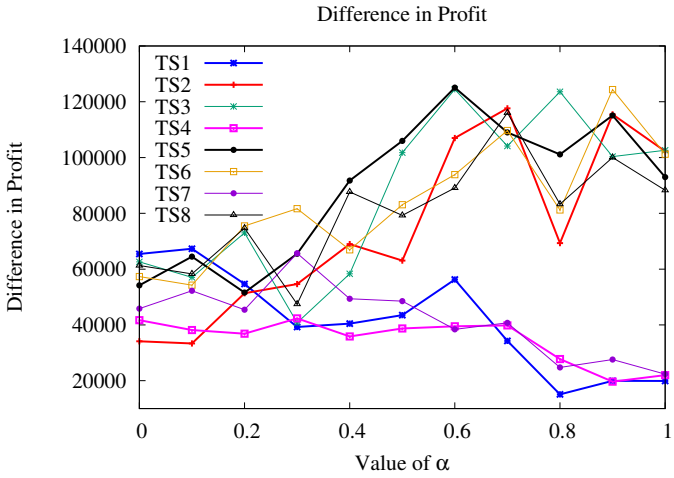

Fig. 5. **D_P for Weight based approach over weight factor α**

value of $\alpha = 0.8$. So for most of the considered data set the value of $\alpha = 0.8$ is better option for weight based heuristic to maximize the profit. Not only the value of $\alpha = 0.8$ maximize the profit but from the Figure 5 it is also clear that for most of the difference in profit in lower at $\alpha = 0.8$ in the range of $\alpha \in [0.6 - 1.0]$. So we choose the value of $\alpha = 0.8$ for the weight based heuristics to compare performance with other approaches.

Figure 6 shows the trends of $\frac{T_P}{D_P}$ value for all the eight data sets for value of α in the range $[0, 1]$ for weight based scheduling approach. The value $\frac{T_P}{D_P}$ is a considered parameter as our objective is to maximize T_P and minimize D_P. It is easily distinguishable that for $\alpha \in [0.6, 0.8]$, $\frac{T_P}{D_P}$ is high for most of the tasks set, and surprisingly for value of $\alpha = 0.8$ the value is maximum for most of the cases.

So far we came up with the conclusion that weight based scheduling approach with $\alpha = 0.8$ is the best among all other heuristic approaches. The performance of weight based scheduling approach to ILP based solution depicted in Figure 7 for all the eight data sets. The ILP solution takes huge time to perform for large data set so for comparison sake, we have opted for small data set and calculated total profit

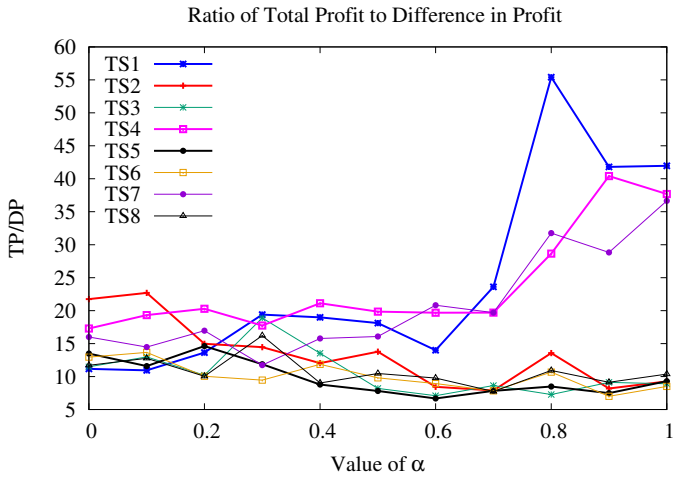

Fig. 6. $\frac{T_P}{D_P}$ **for Weight based approach over weight factor** α

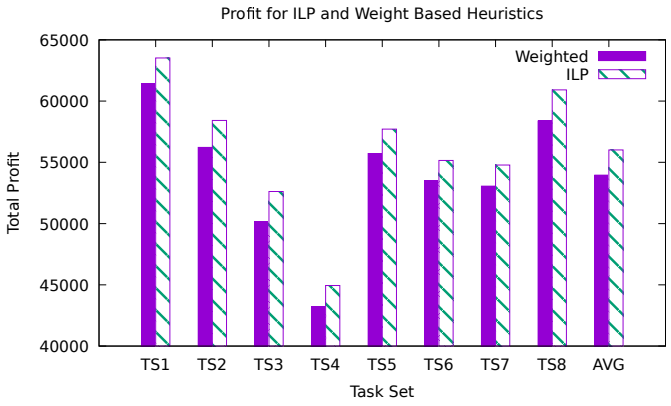

Fig. 7. **Comparing Total Profit** T_P **for ILP and Weight based Heuristic approach**

T_P for both ILP approach and weight based approach. As we can see in the Figure 7, the total profit T_P obtained by ILP is always somewhat higher than total profit obtained by the weight based scheduling approach (with $\alpha = 0.8$) for all the eight data sets. The weight based heuristics achieves on an average of 96.32% of profit of the reported profit by the ILP based optimal solution.

VI. Conclusion and Future Work

A Fog network comprises a group of peer nodes where all the peer nodes take part in a collaborative network for sharing computing and storage resources. We formulate the problem of welfare maximization which is profit maximization and minimization of disparity in profit between sharer nodes, and formulate the said problem in integer linear program for understanding the problem clearly and solved the using standard LP solver. Our propose welfare scheme heuristics for load sharing consider the local and centralized solution approaches, and compared the performance of these for data sets. We heuristically calculate weight of the task parameters in selecting task for sharing with other sharer nodes and the propose weight based task selection in load sharing achieves on an average of 96.23% profit as compared to optimal amount of total profit using ILP based profit maximization. Also, weight based local reservation scheme for load sharing achieve significantly better result in both maximizing total profit and minimizing the disparity in profits of the sharer nodes.

We consider each tasks of the arrived during the time interval has common deadline which is similar to interval duration but it may differ from real world where different deadlines for different tasks and these can be an important extension of the work. We assume the proposed centralized scheduler always works for profit maximization but this can also be modified for both profit maximization and disparity minimization.

References

[1] A. Schulman, V. Navda, R. Ramjee, N. Spring, P. Deshpande, C. Grunewald, K. Jain, and V. N. Padmanabhan, "Bartendr: a practical approach to energy-aware cellular data scheduling," in *Proceedings of Annual International Conference on Mobile Computing and Networking, (MOBICOM)*. ACM, 2010, pp. 85–96.

[2] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications," *IEEE Wirel. Commun.*, vol. 20, no. 3, pp. 1–0, 2013.

[3] D. N. H. T. Dinh, C. Lee and P. Wang, "A survey of mobile cloud-computing: Architecture, applications, and approaches," in *Wireless Commun.Mobile Comput.*, vol. 13, no. 18, 2013, p. 1587–1611.

[4] H. Shamseddine, J. Nizam, A. Hammoud, A. Mourad, H. Otrok, H. Harmanani, and Z. Dziong, "A novel federated fog architecture embedding intelligent formation," *IEEE Network*, vol. 35, no. 3, pp. 198–204, 2021.

[5] J. Z. F. Bonomi, R. Milito and S. Addepalli, "Fog computing and its role in the internet of things," *Workshop Mobile Cloud Comput.(MCC)*, p. 13–16, 2012.

[6] V. Bahl, "Emergence of micro datacenter (cloudlets/edges) for mobile com-puting," *Microsoft Devices Netw. Summit (Keynote Talk)*, 2015.

[7] B. Yin, W. Shen, Y. Cheng, L. X. Cai, and Q. Li, "Distributed resource sharing in fog-assisted big data streaming," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–6.

[8] N. Kumar and A. Mondal, "Work-in-progress: Pricing mechanism and workload scheduling to optimize social welfare and cost for fog computing systems," in *2019 IEEE Real-Time Systems Symposium (RTSS)*, 2019, pp. 544–547.

[9] J. Daigneault and M. St-Hilaire, "Profit maximization model for the task assignment problem in 2-tier fog/cloud network environments," *IEEE Networking Letters*, vol. 3, no. 1, pp. 19–22, 2021.

[10] C. Anglano, M. Canonico, P. Castagno, M. Guazzone, and M. Sereno, "Profit-aware coalition formation in fog computing providers: A game-theoretic approach," *Concurr. Comput. Pract. Exp.*, vol. 32, no. 21, 2020.

[11] C. Anglano, M. Canonico, and M. Guazzone, "Profit-aware resource management for edge computing systems," in *Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking*, ser. EdgeSys'18, 2018, p. 25–30.

[12] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Softw. Pract. Exp.*, vol. 47, no. 9, pp. 1275–1296, 2017.

[13] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. Pract. Exp.*, vol. 41, no. 1, pp. 23–50, 2011.