SPATO: A Student Project Allocation Based Task Offloading in IoT-Fog Systems

Chittaranjan Swain, Manmath Narayan Sahoo and Anurag Satpathy
Department of Computer Science and Engineering
National Institute of Technology, Rourkela, India.
{chittaranjanswain518, anurag.satpathy}@gmail.com, sahoom@nitrkl.ac.in

***Abstract*—The Internet of Things (IoT) devices are highly reliant on cloud systems to meet their storage and computational demands. However, due to the remote location of cloud servers, IoT devices often suffer from intermittent Wide Area Network (WAN) latency which makes execution of delay-critical IoT applications inconceivable. To overcome this, service providers (SPs) often deploy multiple fog nodes (FNs) at the network edge that helps in executing offloaded computations from IoT devices with improved user experience. As the FNs have limited resources, matching IoT services to FNs while ensuring minimum latency and energy from an end-user's perspective and maximizing revenue and tasks meeting deadlines from a SP's standpoint is challenging. Therefore in this paper, we propose a student project allocation (SPA) based efficient task offloading strategy called *SPATO* that takes into account key parameters from different stakeholders. Thorough simulation analysis shows that *SPATO* is able to reduce the offloading energy and latency respectively by 29% and 40% and improves the revenue by 25% with 99.3% tasks executing within their deadline.**

***Index Terms*—IoT, Fog Computing, Task Offloading, Student-Project Allocation, Matching Game.**

I. Introduction

Internet of Things (IoT) has become an indispensable aspect of everyday life owing to its wide range of applications ranging from wearable devices, smart meters, connected vehicles, smart grid, smart health, intelligent transportation systems, and many more [1]. In fact, the number of IoT devices supporting different applications is estimated to exceed the 30 billion mark by 2030 [2]. Typically the IoT devices are resource-constrained and rely upon the remote cloud for computation, storage, and analytics of data. However, cloud data centers (DCs) are usually deployed in locations that are distant from the IoT devices thereby incurring higher response time due to intermittent WAN delays and multi-hopping. The next-generation IoT applications not only require the processing of a huge volume, velocity, and variety of data but also demands the quality of service (QoS), location awareness, real-time mobility support, and latency-sensitive requirements. These specifications render the cloud-based IoT platforms impractical for modern applications.

Fog computing (FC), on the other hand, brings the cloud services closer to the users thereby improving the responsiveness of applications. IoT devices garner the benefits of FC by offloading computations to the nearby FNs. These FNs deployed by the service providers (SPs) have limited resources compared to the cloud, hence offloading IoT services with heterogeneous requirements to disparate FNs is challenging. Considering full offloading scenarios, Adhikari *et al.* [3] proposed an application offloading strategy based on accelerated particle swarm optimization (APSO) for a hierarchical fog-cloud environment that takes into account multiple quality-of-service (QoS) parameters such as cost and resource utilization. Alternatively, Hussein and Mousa [4] discussed two offloading strategies based on ant-colony-optimization (ACO) and particle swarm optimization (PSO) to load balance the assignment of tasks to FNs under communication cost and response time considerations. Some other works that have modeled task offloading as optimization problems are discussed in [5], [6]. Although optimization approaches may guarantee sub-optimal solutions, they suffer from the following pitfalls. Firstly, optimization techniques are incapable of considering contrasting objectives of stakeholders that do not align well with the system-wide objectives. Secondly, the optimization solvers are computationally intensive and are not scalable. Matching theory-based solutions overcome these drawbacks and focus on reducing the response time and energy consumption [7]–[9]. Although these approaches resolve the concerns of optimization solutions, they face the following issues, however. The solution approaches discussed in [7]–[9] are restricted to a single SP environment. Moreover, Gu *et al.* [10] pointed out that the direct interaction between the FNs and IoT devices may raise security concerns such as eavesdropping and data hijacking. As a remedy, all communications including authentication and authorization should be carried out under the supervision of SPs, and FNs are restricted to computations. Hence, in addition to FNs and IoT devices, we introduce SP as another entity into our model [11]. This addition increases the complexity as the agenda of SPs have also to be taken into consideration while generating a solution to the offloading problem. The overall offloading problem is analogous to the student-project-allocation (SPA) model [12]. Additionally, the SPA-based solution converges faster to a stable allocation compared to the conventional matching-based solutions [11].

In this work, we propose SPA based solution approach called *SPATO* for full offloading scenarios considering an additional entity, i.e., SPs in the allocation process. The preferences assigned to FNs by tasks generated by IoT devices are computed considering latency and energy consumption in offloading whereas SPs rank the tasks taking into account the hosting cost and deadline. The overall contributions of the work are as follows:

- We propose SPA based efficient task offloading strategy called *SPATO* that aims at optimizing multiple QoS parameters such as latency, energy, and cost.
- As multiple parameters are involved, the preferences of the tasks are generated using analytical hierarchy process *(AHP)*. We define a new concept called *provider efficiency* (PE) that is used by SPs to rank the tasks. PE takes into consideration the hosting cost and deadline of the task.
- To evaluate *SPATO*, we compare its performance with two different baselines: *SMETO* [13], and a *RANDOM* allocation strategy. Simulation results state that the proposed scheme is able to reduce the offloading energy and latency by 29% and 40% respectively. Moreover, *SPATO* is also able to maximize the overall revenue of SPs by 25% with 99.3% tasks executing with their specified deadlines.

The rest of the paper is organized as follows. Section II discusses the literature that we have reviewed. In Section III we discuss the system model in detail. Section IV talks about the SPA based solution approach. Performance analysis of *SPATO* is discussed in Section V and conclusion are drawn in Section VI.

II. Related Work

Task offloading in a densely connected network is proven to be $\mathcal{NP}$-Hard [9]. In view of full offloading scenarios, Adhikari *et al.* [3] proposed a particle swarm optimization (PSO) based offloading strategy to improve the QoS parameters such as cost and resource utilization. On the other hand, Hussein and Mousa [4] focused on load-balanced assignment of tasks to FNs using ant-colony optimization (ACO) and particle swarm optimization (PSO) based meta-heuristics. Some other optimization based solutions are discussed in [5] [6]. The inherent downside of optimization solutions such as its inability to consider agendas of multiple stakeholders, extensive computational requirements, and non-scalable nature does not make it a plausible solution candidate. In this regard, matching theory-based solution approaches have recently gained popularity owing to their ability to capture objectives of different stakeholders via preferences, ensuring the fairness of allocation using the concept of stability and its highly scalable nature. In the context of matching based solution approaches Abouaomar *et al.* [7] discussed a response time-oriented fog user assignment considering a densely connected IoT-Fog interconnection network. To reduce the overall energy consumption and satisfy the heterogeneous delay requirements in multi-access edge computing environments, Gu *et al.* [8] proposed a context-aware task offloading technique based on a matching game with externalities. To jointly optimize the system energy and the overall latency in offloading a hybrid CRITIC and TOPSIS based ranking followed by matching is presented in [9].

All the above approaches do not consider the presence of a third party called the SPs that often deploy and manage these FNs. However, the absence of SPs raises security concerns as the FNs directly interact with the IoT devices [10]. Moreover, in this work, we consider the presence of SPs in addition to that of IoT devices and FNs. As the traditional matching involves two sets of agents, we use SPA to model the task offloading problem considering multiple QoS parameters such as energy, latency, and cost. Next, we discuss the system model followed by SPA based allocation procedure.

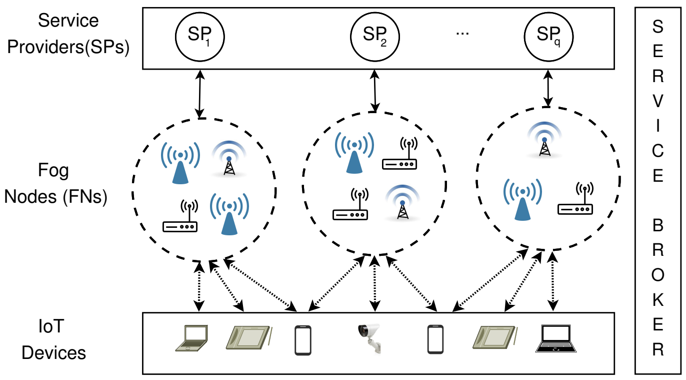

Fig. 1: Typical fog computing architecture with multiple SPs.

III. System Model

The overall architecture of an interconnected fog network is depicted in Fig. 1. It consists of a set IoT devices denoted by $\mathbb{D} = \{d_1, d_2, d_3, \cdots, d_m\}$, where a device d_i generates an atomic task $t_i \in \mathbb{T}$. For execution, these tasks are offloaded to FNs that are deployed and owned by a set of geo-separated SPs represented by $\mathbb{S} = \{s_1, s_2, s_3, \cdots, s_q\}$. Let F_k be the set of FNs of SP s_k. The set of FNs in the system, $\mathbb{F} = \bigcup_{k=1}^{q} \mathbb{F}_k$. Each FN $f_j^k \in \mathbb{F}$ corresponds to the j^{th} FN of the k^{th} SP. Additionally, we also consider the presence of a centralized service broker (SB) that performs the assignment of tasks to FNs based on a SPA matching strategy.

For a device d_i, let $\mathbb{A}_i \subseteq \mathbb{F}$ be the set of feasible FNs to which t_i can be offloaded. The offloading request corresponding to a task t_i is represented using a triplet $\langle I_i, \Gamma_i, \Upsilon_i \rangle$, where I_i is the size ($bits$), Γ_i corresponds to computational requirement in CPU $cycles$ and Υ_i expresses the maximum tolerable delay, i.e., deadline in sec. The computational capacity of a FN f_j^k is logically partitioned into a number of executable components called virtual resources units (VRUs) [7]. The number of VRUs of f_j^k is recorded as C_j^k, which denotes its capacity. Each VRU of f_j^k gets an equal share of the host's computational cycles, represented by η_j^k. In order to add heterogeneity to our model, we consider FNs with different capacities. The capacity of a provider s_k, denoted by C_k, indicates the maximum number of offloaded tasks that it can serve in a given time frame using its deployed FNs $\mathbb{F}_k$ [12]. C_k can be computed as $C_k = \sum C_j^k,\ j \in [1, |\mathbb{F}_k|]$.

A. Communication Model

Offloading encompasses three phases: (*i.*) Transmission of the task to a FN, (*ii.*) Execution at the FN, *(iii.)* Retrieval of computed results from the FN. Assuming a provider operates at a bandwidth B with OFDMA (orthogonal frequency division multiple access), an IoT device communicates with a FN of s_k with a channel capacity $W_k \leftarrow \frac{B}{C_k}$ [14]. The effective uplink rate $R^k_{i,j}$ from d_i to f^k_j is calculated as per Eq. (1). Here, p_i represents the transmission power of d_i, $h^k_{i,j}$ is the channel gain between d_i and f^k_j, and n_0 denotes the noise power of the channel.

$$R^k_{i,j} = W_k\ log_2 \left(1 + \frac{p_i h^k_{i,j}}{n_0}\right) \tag{1}$$

B. Parameters concerning IoTs

The IoT devices aim at minimizing the total latency and energy consumption in offloading a task.

1) Latency Computation: The latency incurred in offloading a task t_i to a FN f^k_j is computed based on *(i.)* Transmission delay to f^k_j, as per Eq. (2), *(ii.)* Processing delay at f^k_j, as per Eq. (3), and *(iii.)* Receiving delay from f^k_j, which is negligible as a small amount of processed result is to be transmitted back over a channel with comparatively high downlink rate [15] [16].

$$TT^k_{i,j} = \frac{I_i}{R^k_{i,j}} \tag{2}$$

$$ET^k_{i,j} = \frac{\Gamma_i}{\eta^k_j} \tag{3}$$

Thus, total latency $T^k_{i,j}$ in offloading is derived as per Eq. (4).

$$T^k_{i,j} = TT^k_{i,j} + ET^k_{i,j} \tag{4}$$

2) Energy Computation: The total energy incurred in offloading a task t_i form an IoT device d_i to a FN f^k_j consists of three components: *(i.)* Transmission energy of d_i, as per Eq. (5) *(ii.)* Execution energy at f^k_j, as per Eq. (6), and *(iii.)* Receiving energy at d_i, which is ignored since it is negligible [15]. In Eq. (5) and (6), p_i and p^k_j refer to the transmission power of d_i and computational power of the f^k_j respectively.

$$TE^k_{i,j} = p_i * TT^k_{i,j} \tag{5}$$

$$EE^k_{i,j} = ET^k_{i,j} * p^k_j \tag{6}$$

Thus, total energy $E^k_{i,j}$ in offloading is computed as per Eq. (7).

$$E^k_{i,j} = TE^k_{i,j} + EE^k_{i,j} \tag{7}$$

The bi-objective minimization cost function for a device d_i is captured by a *utility score* $C^k_{i,j}$, which is weighted average of $T^k_{i,j}$ and $E^k_{i,j}$ and is computed using Eq. (8). The weights w_1 and w_2 are obtained using analytical hierarchy process *(AHP)* which is discussed in Section IV-1.

$$C^k_{i,j} = w_1 * T^k_{i,j} + w_2 * E^k_{i,j} \tag{8}$$

C. Parameter Concerning SPs

The overall aim of a SP is to optimize its revenue by executing the maximum number of offloaded tasks within their respective deadlines. We introduce a term called *provider efficiency* (PE) that captures the efficiency of a provider taking into consideration two parameters, viz. (*i.*) hosting cost at a SP, and (*ii.*) deadline of the tasks. The PE $\mathcal{P}^k_i$ of a SP s_k to execute a task t_i can be derived as per Eq. (9). The numerator expresses the overall revenue obtained by executing t_i and is directly dependent on the task size I_i. However, a SP can achieve higher efficiency by executing more tasks satisfying their deadlines. This can only be achieved if a higher preference is assigned to tasks with closer deadlines. Thus $\mathcal{P}^k_i$ varies inversely with Υ_i. Here, $\mathcal{C}_k$ is a constant with unit *dollar/Mbps*.

$$\mathcal{P}^k_i = \mathcal{C}_k * \frac{I_i}{\Upsilon_i} \tag{9}$$

Next, we introduce $x^k_{i,j}$ to be a binary indicator variable defined as per Eq. (10).

$$x^k_{i,j} = \begin{cases} 1 : \text{if } t_i \text{ is assigned to } f^k_j \text{ owned by SP } s_k \\ 0 : \text{otherwise} \end{cases} \tag{10}$$

The revenue obtained by a SP s_k, denoted as Rev_k, for executing a set of tasks offloaded to it is calculated as per Eq. (11).

$$Rev_k = \sum_{i=1}^{m} x^k_{i,j} * \mathcal{P}^k_i \tag{11}$$

D. Problem Formulation

The IoT devices aim at minimizing the overall latency and energy in the offloading process whereas SPs aim at maximizing the hosting cost and minimizing the number of outages due to tasks exceeding their deadlines. As discussed previously, PE encapsulates the dual goals of the SPs. The overall objective of *SPATO* is presented in Eq.(12a).

$$\min_{\forall i \in [1,m]} \left(C^k_{i,j}\right) \ \textit{and} \ \min_{\forall k \in [1,q]} \left(\frac{1}{Rev_k}\right) \tag{12a}$$

$$\textit{s.t.} \quad \sum_{k=1}^{q} \sum_{j=1}^{|\mathbb{F}_k|} x^k_{i,j} = 1; \ \ i \in [1, m] \tag{12b}$$

$$\sum_{i=1}^{m} x^k_{i,j} \le C^k_j; \ \ k \in [1, |\mathbb{S}|], j \in [1, |\mathbb{F}_k|] \tag{12c}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{|\mathbb{F}_k|} x^k_{i,j} \le C_k; \ \ k \in [1, |\mathbb{S}|] \tag{12d}$$

Constraint (12b) ensures that a task is allocated to only one FN. A FN can service tasks up to its capacity which is put as Constraint (12c) The total number of tasks assigned to a SP is limited to its capacity which the sum of the capacities of the FNs owned by it. This is presented as Constraint (12d). The overall problem expressed in Eq. (12a) is a combinatorial problem and is proven to be $\mathcal{NP}$-Hard [9]. In fact, for a larger sample space, it is almost infeasible to solve the optimization problem in the polynomial-time frame. Therefore, we propose a matching theory-based heuristic to solve the task offloading problem in polynomial time. The solution approach is detailed in the next section.

IV. Task Offloading via Student Project Allocation Game

The SPA strategy is primarily used to assign students to projects offered by lecturers in universities [12]. In a typical SPA setting, each lecturer has a quota indicating the maximum number of students that he can supervise. On similar grounds, each project of a supervisor is also characterized by its quota indicating the maximum intake of the project. Moreover, each student expresses his preference by ranking all of its acceptable projects, likewise, a lecturer constructs his preferences over students who opted for at least one of his projects. Motivated from [12], we model the task offloading problem as a SPA game. Here IoT devices, FNs, and SPs are analogous to students, projects, and lecturers respectively. As preferences of IoT devices are computed based on multiple criteria, we use the analytical hierarchy process *(AHP)* to obtain a unified ranking of the FNs.

1) Ranking based on AHP: The overall working of *AHP* to generate the preference profile of IoT devices is discussed subsequently. We highlight important steps used in the process. A detailed discussion of *AHP* can be found in [17]. Considering $\mathbb{C} = \{c_1, c_2, \cdots, c_{|\mathbb{C}|}\}$ to be a set of distinct criteria, we construct a pairwise comparison matrix $\mathbb{P} \in R^{|\mathbb{C}|*|\mathbb{C}|}$. In our model $\mathbb{P} \in R^{2*2}$. An entry $c_{r,v} \in \mathbb{P}$ denotes the relative importance of criterion c_r against c_v. We use a linear judgment scale to set the relative importance of criteria, due to its proven superiority [17] over other judgment scales. This relative importance of criteria is imposed by the stakeholder; IoT devices in our model. After constructing the pairwise comparison matrix $\mathbb{P}$, we normalize each entry to obtain a normalized pairwise comparison matrix $\mathbb{P}'$. The normalized matrix is then averaged row-wise to obtain the column vector $\mathcal{W} \in R^{|\mathbb{C}|\times 1}$ containing the weights of each criterion. Next, the decision matrix $D_i \in R^{|A_i|\times|\mathbb{C}|}$ corresponding to each device $d_i \in \mathbb{D}$ is also normalized to $D_i^{'}$. Finally, the global rank vector for an IoT device d_i, given by $\mathbb{G}_i \in R^{|A_i|\times 1}$, can be obtained as $\mathbb{G}_i = D_i^{'} \times \mathcal{W}$.

2) SPA based efficient task offloading: The analogous SPA-based matching game for task offloading can be mathematically expressed as per Definition 1 and 2.

Definition 1. *Considering two sets of agents $\mathbb{T}$ and $\mathbb{S}$, let $P(a)$ be the preference profile of agent $a \in \mathbb{T} \cup \mathbb{S}$. For instance, a task $t_i \in \mathbb{T}$ ranks one or more FNs from $\mathbb{F}$. Similarly, a SP $s_k \in \mathbb{S}$ ranks some or all tasks from $\mathbb{T}$.*

Definition 2. *The matching game is based on a mapping function $\lambda : \mathbb{T} \cup \mathbb{F} \rightarrow 2^{\mathbb{T}\cup\mathbb{F}}$ such that:*

$$\lambda(t_i) \subset \mathbb{F} \text{ and } |\lambda(t_i)| \leq 1 \tag{13a}$$

$$\lambda(f_j^k) \subseteq \mathbb{T} \text{ and } |\lambda(f_j^k)| \leq C_j^k \tag{13b}$$

$$\sum_{j=1}^{|\mathbb{F}_k|} \lambda(f_j^k) \leq C_k, \; k \in [1, |\mathbb{S}|] \tag{13c}$$

$$f_j^k \in \lambda(t_i) \Leftrightarrow t_i \in \lambda(f_j^k) \tag{13d}$$

Condition (13a) states that a task is matched to at most one FN. Condition (13b) ensures a maximum number of tasks assigned to a FN should be less than or equal to its quota. The maximum number of tasks that can be served by a SP can be no more than C_k and this is reflected in Condition (13c). Condition (13d) states that a task t_i is matched to a FN f_j^k *iff* f_j^k is matched to t_i.

Definition 3. *A pair (t_i, f_j^k) is a blocking pair if $f_j^k \notin \lambda(t_i)$ and the following conditions are satisfied:*

1) $f_j^k \in A_i$, i.e., t_i finds f_j^k acceptable,
2) $\lambda(t_i) = \phi$, or $f_j^k \succ_{t_i} \lambda(t_i)$, and
3) either
  a) f_j^k is undersubscribed, i.e., $|\lambda(f_j^k)| < C_j^k$ or
  b) f_j^k is full, i.e., $|\lambda(f_j^k)| = C_j^k$ and $\exists\, t_{i'} \in \lambda(f_j^k)$ s.t. $t_i \succ_{s_k} t_{i'}$ or
  c) s_k is full, i.e., $|\lambda_k| = C_k$, where $\lambda_k = \cup_{j=1}^{|\mathbb{F}_k|}\lambda(f_j^k)$ and $t_i \succ_{s_k} t_{i'}$, where $t_{i'}$ is the worst assigned task to s_k.

Definition 4. *A matching λ is said to be stable iff it is not blocked by any pair of agents.*

The preference profile of all agents are *strict* and *transitive*. *Strictness* ensures that an agent is not indifferent between any two agents of the other set implying the absence of ties. Considering tasks t_x, t_y, t_z of $\mathbb{T}$, *transitivity* implies that if an agent $f_j^k \in \mathbb{F}$ of another set has preferences of the type $t_x \succ_{f_j^k} t_y$ and $t_y \succ_{f_j^k} t_z$ then $t_x \succ_{f_j^k} t_z$ also holds.

Task t_i assigns preferences to $f_j^k \in \mathbb{F}$ depending on utility score $C_{i,j}^k$ computed as per Eq. (8). Therefore,

$$f_j^k \succ_{t_i} f_{j'}^{k'} \iff C_{i,j'}^{k'} > C_{i,j}^k; \;\; j \neq j' \text{ or } k \neq k'$$

Likewise, s_k assigns preferences to $t_i \in \mathbb{T}$ based on the PE $\mathcal{P}_i^k$ calculated as per Eq. (9). Hence,

$$t_i \succ_{s_k} t_{i'} \iff \mathcal{P}_i^k > \mathcal{P}_{i'}^k; \;\; i \neq i'$$

The working of *SPATO* is shown in Algorithm 1. The input to the algorithm is the set of agents $\mathbb{T}$, $\mathbb{S}$ and $\mathbb{F}$; and the preferences of each agent $a \in \mathbb{T} \cup \mathbb{S}$. The algorithm outputs a stable assignment through λ. Initially, all IoT devices are set to be free and each FN and SP are unsubscribed. Steps 2-4 involve each unassigned task t_i sending a proposal to its most preferred FN f_j^k that it has not yet proposed. Then we perform a provisional assignment of t_i to f_j^k. After performing the provisional assignment the following cases may arise. If f_j^k is oversubscribed, then worst task $t_{i'}$ assigned to f_j^k is found and the provisional assignment between $t_{i'}$ and f_j^k is broken

Algorithm 1: SPA based efficient task offloading algorithm *(SPATO)*

```
Input: $P_i, \forall t_i \in \mathbb{T};\ C_k, P_k, \forall s_k \in \mathbb{S};\ C_j^k, \forall f_j^k \in \mathbb{F}$.
Result: $\lambda : \mathbb{T} \cup \mathbb{F} \rightarrow 2^{\mathbb{T} \cup \mathbb{F}}$
1  Initialize: All $i \in \mathbb{T}$ as free and each $f_j^k \in \mathbb{F}$ and $s_k \in \mathbb{S}$ as unsubscribed.
2  while $\exists\, i \mid t_i$ is free and $P_i \neq \phi$ do
3      $f_j^k$ = most preferred FN in $P_i$ not yet proposed
4      Send proposal to $f_j^k$ and performs provisional assignment with $t_i$
5      if $f_j^k$ is over-subscribed then
6          $t_{i'}$ = worst task assigned to $f_j^k$
7          Break the assignment between ($t_{i'}$, $f_j^k$)
8      if $f_j^k$ is full then
9          $t_{i'}$ = worst task assigned to $f_j^k$
10         for each $t_{i^*} \mid t_{i'} \succ_{f_j^k} t_{i^*}$ in $P_j^k$ do
11             Delete ($t_{i^*}$, $f_j^k$)
12     if $s_k$ is full then
13         $t_{i'}$ = worst task assigned to $s_k$
14         for each $t_{i^*} \mid t_{i'} \succ_{s_k} t_{i^*}$ in $P_k$ do
15             Remove $t_{i^*}$ from $P_k$
16             for each $f_j^k \in \mathbb{F}_k \cap A_{i^*}$ do
17                 Delete ($t_{i^*}$, $f_j^k$)
```

The Delete(x, y) operation removes x from preference list of y and vice versa.

(Steps 5-7). If f_j^k is full, then the worst task $t_{i'}$ assigned to it is identified and all tasks having preference lower than $t_{i'}$ are removed from the preference list P_j^k of f_j^k derived form preference list of s_k, i.e., P_k (Steps 8- 11). Accordingly, P_k is also updated. Alternatively if s_k is full, all less preferred tasks t_{i^*} than $t_{i'}$ are removed from P_k. As a consequence all $f_j^k \in \mathbb{F}_k \cap A_{i^*}$ are also eliminated from P_{i^*} (Steps 12-17). The algorithm outputs a stable allocation with no agent having an incentive to deviate from their current allocation.

V. Performance Evaluation

We have performed a simulation using the iFogSim simulator [18]. The environmental setup and analysis of the simulation results are discussed elaborately in this section.

A. Environmental Setup

We consider an interconnected fog network with 4 geo-separated SPs. Each SP owns a 20 *MHz* channel which is further subdivided into C_k sub-channels of equal capacity. The constant $\mathcal{C}_k$ for a SP s_k is randomly assigned in the range $[50, 100]$ *dollar/Mbps*. The IoT devices and FNs are deployed randomly over a 2-D space with coordinates generated uniformly in the range U$[0, 100]$. The maximum coverage of IoT devices is set in the range U$[200, 500]$ *m*. The computational capabilities of FNs are expressed in the form of VRUs which are generated following the uniform distribution U$[50, 300]$. The computational rate (*cycles/s*) and computational power (*W*) are chosen in the range U$[6, 10]$ *GHz* and U$[0.35, 0.55]$ *W* respectively. The number of IoT devices varies in the range 250-1000 at an interval of 250 per observation. The task specific parameters such as input size, computational demand and deadline are generated uniformly in the range U$[300, 600]$ *Kb*, U$[210, 480]$ *million cycles* and, U$[5, 30]$ *s*, respectively. Considering PCS-1900 GSM band, the free space path loss in dB between an IoT device d_i and FN f_j^k is calculated as $PL_{d_i, f_j^k} = 38.02 + 20log(dist(d_i, f_j^k))$, where $dist(d_i, f_j^k)$ is the distance between d_i and f_j^k. The channel gain is then calculated as $h_{i,j}^k = 10^{-\left(PL_{d_i, f_j^k}\right)/10}$. The transmission power of IoT device and noise power of channels is set to 0.5 and 10^{-10} *W* respectively.

B. Baseline Algorithms

To assess the performance of *SPATO*, we compare its behavior with two baseline algorithms, viz., (*i.*) Zu *et al.* [13], referred to as *SMETO* and (*ii.*) a random allocation strategy, referred to as *RANDOM*. *SMETO* is based on a one-to-many matching game aimed at reducing the energy consumption in the offloading process. The *RANDOM* allocation strategy randomly assigns tasks to FNs.

C. Experimental Results

Fig. 2 demonstrates the total energy consumed in offloading [250, 1000] tasks with an interval of 250 tasks across observations. As expected the offloading energy increases for an increasing number of tasks. The proposed algorithm outperforms *SMETO* and *RANDOM* baselines. In contrast to *SMETO*, where FNs are ranked depending on transmission energy, *SPATO* considers both latency and total offloading energy for ranking the FNs. The ranking strategy of *SPATO* ensures that tasks are mostly offloaded to FNs with better computation capabilities which lead to reduced execution time, thereby reducing overall offloading time and energy. Fig. 3 depicts the mean offloading time considering different number of tasks. The inability of *SMETO* to consider offloading delay while generating preferences for IoT devices leads to elevated offloading time. This is because the FNs are ranked based on distance from the IoT devices which reduces the transmission time but does not ensure faster computation. On the contrary, *SPATO* generates a unified ranking considering both energy and latency leading to reduced offloading delay.

The ranking in *SPATO* considers offloading latency while generating preferences (Eq. (8)) for FNs and considers deadline while ranking the IoT devices (Eq. (9)). The combined effect of these rankings boosts the possibility of tasks getting executed within their specified deadlines. As none of the baseline algorithms consider execution time or deadline while generating preferences, they suffer from a higher number of outages which can be easily observed from Fig. 4. The overall revenue obtained considering different approaches is shown in Fig. 5. The proposed strategy ensures higher revenue compared to both *SMETO* and *RANDOM* strategies. The reason for this behavior is twofold. Firstly, offloading latency considered in

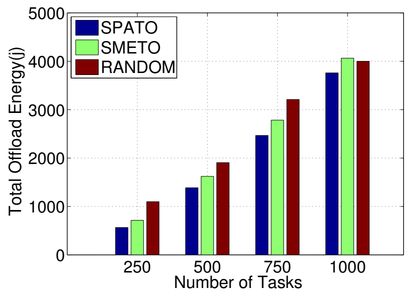

Fig. 2: Total Offload Energy Vs. Number of Tasks.

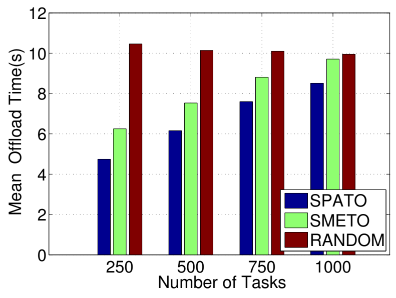

Fig. 3: Mean Offload Time Vs. Number of Tasks.

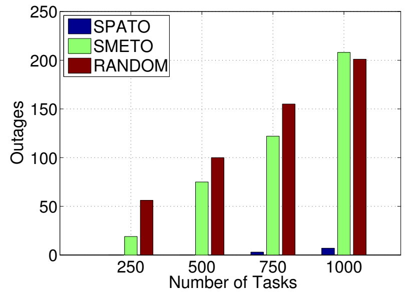

Fig. 4: Outages Vs. Number of Tasks.

ranking the FNs guarantees allocation of more tasks to FNs with better computational capabilities resulting in revenue boost compared to the baseline algorithms. Secondly, SPs prefer large sized tasks (Eq. (9)) leading to higher revenue.

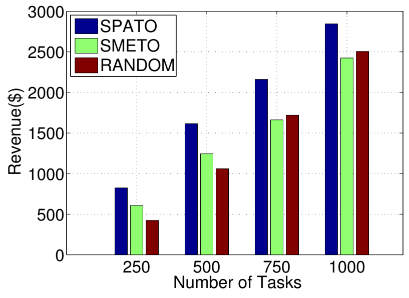

Fig. 5: Revenue Vs. Number of Tasks.

VI. Conclusion

In this paper, we have proposed a model called *SPATO* that aims to optimize multiple quality-of-service (QoS) parameters such as latency, energy, and cost in offloading multiple tasks in a densely connected multi-SP environment. Since, offloading in such a complex network is $\mathcal{NP}$-Hard, a student-project allocation (SPA) based polynomial time solution framework is developed. To assess the performance of the proposed technique, we compare its behavior with two baseline algorithms. Simulation results confirm improved performance in terms of reduced delay and energy in offloading heterogeneous tasks. Moreover, *SPATO* is also able to maximize the SPs' revenue with minimum outages. As an immediate future direction to this work, we would like to consider intra and inter-channel interference arising due to channel re-usability.

References

[1] M. Adhikari, M. Mukherjee, and S. N. Srirama, "Dpto: A deadline and priority-aware task offloading in fog computing framework leveraging multilevel feedback queueing," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5773–5782, 2020.

[2] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO white paper*, vol. 1, no. 2011, pp. 1–11, 2011.

[3] M. Adhikari, S. N. Srirama, and T. Amgoth, "Application offloading strategy for hierarchical fog environment through swarm optimization," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4317–4328, 2020.

[4] M. K. Hussein and M. H. Mousa, "Efficient task offloading for iot-based applications in fog computing using ant colony optimization," *IEEE Access*, vol. 8, pp. 37 191–37 201, 2020.

[5] C. Zhu, J. Tao, G. Pastor, Y. Xiao, Y. Ji, Q. Zhou, Y. Li, and A. Ylä-Jääski, "Folo: Latency and quality optimized task allocation in vehicular fog computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4150–4161, 2018.

[6] X. Li, Y. Liu, H. Ji, H. Zhang, and V. C. M. Leung, "Optimizing resources allocation for fog computing-based internet of things networks," *IEEE Access*, vol. 7, pp. 64 907–64 922, 2019.

[7] A. Abouaomar, A. Kobbane, and S. Cherkaoui, "Matching-game for user-fog assignment," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[8] B. Gu, Z. Zhou, S. Mumtaz, V. Frascolla, and A. Kashif Bashir, "Context-aware task offloading for multi-access edge computing: Matching with externalities," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[9] C. Swain, M. N. Sahoo, A. Satpathy, K. Muhammad, S. Bakshi, J. J. P. C. Rodrigues, and V. H. C. de Albuquerque, "Meto: Matching theory based efficient task offloading in iot-fog interconnection networks," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[10] Y. Gu, Z. Chang, M. Pan, L. Song, and Z. Han, "Joint radio and computational resource allocation in iot fog computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 8, pp. 7475–7484, 2018.

[11] L. Cao, F. Yao, H. Zhao, and J. Zhang, "Distributed resource allocation for d2d-enabled two-tier cellular networks with channel uncertainties," in *2016 IEEE International Conference on Communication Systems (ICCS)*, 2016, pp. 1–5.

[12] D. J. Abraham, R. W. Irving, and D. F. Manlove, "Two algorithms for the student-project allocation problem," *Journal of Discrete Algorithms*, vol. 5, no. 1, pp. 73–90, 2007.

[13] Y. Zu, F. Shen, F. Yan, L. Shen, F. Qin, and R. Yang, "Smeto: Stable matching for energy-minimized task offloading in cloud-fog networks," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, 2019, pp. 1–5.

[14] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435–3447, 2017.

[15] G. Zhang, F. Shen, Y. Yang, H. Qian, and W. Yao, "Fair task offloading among fog nodes in fog computing networks," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[16] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2019.

[17] A. Satpathy, M. N. Sahoo, L. Behera, C. Swain, and A. Mishra, "Vmatch: A matching theory based vdc reconfiguration strategy," in *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*, 2020, pp. 133–140.

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