IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 5 March 2024, accepted 26 March 2024, date of publication 2 April 2024, date of current version 16 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3384334

RESEARCH ARTICLE

Game-Theoretic Resource Allocation and Dynamic Pricing Mechanism in Fog Computing

ANJAN BANDOPADHYAY[1], SUJATA SWAIN[1], RAJ SINGH[1], PRITAM SARKAR[1], SIDDHARTHA BHATTACHARYYA[2,3], (Senior Member, IEEE), AND LEO MRSIC[3,4]

[1]School of Computer Engineering, Kalinga Institute of Industrial Technology (Deemed to be University), Bhubaneswar, Odisha 751024, India
[2]Department of Computer Science, VSB Technical University of Ostrava, 708 00 Ostrava, Czech Republic
[3]Algebra University, 10000 Zagreb, Croatia
[4]Rudolfovo Scientific and Technological Center, 8000 Novo Mesto, Slovenia

Corresponding author: Siddhartha Bhattacharyya (dr.siddhartha.bhattacharyya@gmail.com)

This work was supported by the Project: 101104579, AI-powered Next Generation of VET (AI4VET4AI) ERASMUS-EDU-2022-PEX-COVE.

ABSTRACT Fog computing is a promising and challenging paradigm that enhances cloud computing by enabling efficient data processing and storage closer to data sources and users. This paper introduces a game-theoretic approach called GTRADPMFC (Game-Theoretic Resource Allocation and Dynamic Pricing Mechanism in Fog Computing) to address resource allocation and dynamic pricing challenges in fog computing environments with limited resources. The proposed model features non-cooperative competition among fog nodes for resources and dynamic pricing mechanisms to encourage efficient resource utilization. Theoretical analysis and simulations demonstrate that GTRADPMFC improves resource efficiency and overall fog computing system performance. Additionally, the paper discusses how to handle situations with insufficient samples and provide flexibility for users unable to meet completion time requirements. GTRADPMFC effectively manages resource allocation by establishing pricing in fog computing, considering potential delays in completion time. This is achieved through research, simulations, convergence analysis, complexity evaluation, and optimization guarantees.

INDEX TERMS Fog computing, dynamic pricing, resource allocation, game-theoretic approach.

I. INTRODUCTION

Fog computing has emerged as a transformative paradigm that brings cloud computing capabilities closer to the edge of the network, enabling efficient data processing and storage in proximity to data sources and end-users. This decentralized approach to computing offers numerous advantages, such as reduced latency, improved bandwidth utilization, enhanced privacy, and the ability to handle real-time data processing requirements. However, the resource constraints inherent in fog computing environments pose significant challenges in achieving efficient resource allocation and pricing strategies.

Resource allocation in fog computing involves the allocation of computational, storage, and communication resources among multiple entities, including fog nodes, users, and service providers. These entities often have diverse objectives and requirements, making it essential to design mechanisms that balance their competing interests while maximizing overall system performance. Moreover, the dynamic nature of fog computing environments, characterized by varying resource availability and user demand, necessitates adaptive resource allocation approaches [1], [2].

Additionally, pricing mechanisms play a crucial role in incentivizing efficient resource utilization and optimizing the allocation of resources. By appropriately pricing resources, fog nodes can encourage users to make optimal decisions regarding resource consumption, considering the quality of service requirements, available resources, and demand fluctuations. Dynamic pricing mechanisms that adjust prices in real-time based on changing conditions enable the system to adapt to varying resource availability and demand patterns. Game theory provides a robust framework for modeling and analyzing the interactions among entities in fog computing environments to address these challenges. Game theory

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Wang.

allows us to capture the strategic behavior of fog nodes and users, considering their rational decision-making processes and self-interest. By formulating resource allocation and pricing problems as games, we can design mechanisms that lead to desirable outcomes, such as maximizing system utility, achieving fairness, and enhancing resource utilization efficiency [3].

This research paper explores the application of game theory to address resource allocation and dynamic pricing in fog computing. We propose a non-cooperative game model that captures the competitive interactions among fog nodes in resource allocation. Additionally, we design a dynamic pricing mechanism that adjusts prices based on resource availability, demand, and quality of service requirements. The performance of the proposed approach is evaluated through a combination of theoretical analysis and simulations. Theoretical analysis involves rigorous study of the algorithm's properties, complexity, convergence, and performance guarantees. This analysis helps establish the theoretical foundations of the approach and provides insights into its expected behavior and performance.

The proposed game theoretic approach can potentially optimize resource utilization, enhance system performance, and enable fair and efficient allocation of resources in fog computing environments. The core difficulty in fog computing is creating an effective dynamic pricing system that overcomes the constraints of current pricing schemes while also integrating resource allocation [3]. In fog computing, where resources are distributed across the network's edge, designing a pricing mechanism that optimally allocates resources to users while considering factors such as resource availability, user demand, and fairness is crucial [4]. This challenge arises due to the unique characteristics of fog computing, such as the distributed nature of resources, varying availability, and the need to meet user demands within specified time frames. A framework has been put forth in this manner, according to [5], where a sound economic mechanism is designed while a value-based efficiency is built to optimize social welfare. Specifically, we introduce a novel mechanism, the Game-Theoretic Resource Allocation, and Dynamic Pricing Mechanism in Fog Computing (GTRADPMFC), to handle completion time failures effectively. This mechanism combines resource allocation and dynamic pricing strategies to optimize resource utilization and enhance the overall performance of fog computing systems [6]. In this situation, we've looked at how to develop a dynamic pricing strategy that works when users are unable to finish the tasks they want to within the allotted time, thereby

- Providing consumers with flexibility if they cannot meet their assignment deadlines within the specified time frame.
- Deciding as to what is the appropriate course of action when insufficient samples are available. (This situation occurs at the beginning of the process).

The structure of the rest of this paper is as follows. Section II offers a comprehensive review of fog computing, resource allocation, dynamic pricing strategies, and completion time failures, providing an in-depth analysis of the existing research in these areas. Section III overviews our proposed system model. We describe the key components, such as the fog computing framework, resource allocation, and pricing structures. This sets the foundation for understanding the subsequent sections. Section IV presents our proposed mechanism in detail here. We explain how resource allocation and dynamic pricing strategies work together to address completion time failures. We discuss the algorithms and techniques for allocating resources, calculating prices, and handling user demands in the fog computing environment. Section V analyzes the proposed mechanism in this section. We evaluate its performance using theoretical analysis, mathematical modeling, and simulations. We discuss the advantages and limitations of our approach and compare it with existing baseline approaches. This analysis provides insights into the effectiveness and efficiency of our proposed mechanism. In Section VII, we summarize the essential findings and contributions of this paper. We highlight the benefits of our proposed mechanism in addressing completion time failures in fog computing. We also discuss the practical implications and potential applications of our approach. Furthermore, we identify future research directions and areas for improvement in fog computing systems [7].

The remaining sections of this paper delve into the details of our proposed mechanism. We provide a comprehensive analysis of its performance and discuss the implications of our findings. We encourage further exploration and advancement in fog computing and dynamic pricing schemes by concluding the paper with future directions. Here are some limitations of current solutions in the domain:

- Dynamic Pricing Challenges: Current dynamic pricing mechanisms may not effectively address completion time failures, where users are unable to meet their assignment deadlines within the specified time frame.
- Insufficient Handling of Incomplete Samples: Current solutions may struggle to handle situations with insufficient samples, particularly at the beginning of the process.
- Complexity and Scalability: Existing solutions may exhibit limitations in terms of complexity and scalability, particularly concerning the integration of resource allocation and dynamic pricing mechanisms.
- Effectiveness in Real-world Deployment: While theoretical analyses and simulations provide insights into the proposed mechanism's performance, the effectiveness of the solution in real-world deployment scenarios may not be adequately addressed.
- Interoperability and Standardization: Current solutions may lack standardized protocols and interfaces for interoperability between fog nodes, cloud platforms, and edge devices.
- Energy Efficiency and Cost-effectiveness: Energy efficiency and cost-effectiveness considerations are crucial in fog computing environments, but current solutions

may not sufficiently optimize resource allocation and pricing.
- Security and Privacy Concerns: Security and privacy challenges are significant in fog computing due to the distributed nature of resources and sensitive data processing at the network edge.

II. LITERATURE REVIEW

The field of fog computing has garnered significant attention in recent years, resulting in a growing body of research focused on resource allocation and pricing mechanisms in fog computing environments. This section provides an overview of the existing literature. It highlights the limitations of current approaches, emphasizing the need for game theoretical models in optimizing resource allocation and dynamic pricing [8], [9], [10], [11], [12].

Several studies have explored resource allocation techniques in fog computing. Traditional approaches include centralized resource allocation algorithms that optimize resource utilization based on predetermined criteria [5], [13]. However, these methods may suffer from scalability issues and lack adaptability to dynamic fog environments [14], [15]. Some researchers have proposed decentralized resource allocation methods, where fog nodes autonomously negotiate and exchange resources based on their local information. While these approaches address scalability concerns, they often assume full cooperation among fog nodes, which may not hold in practice [16], [17], [18].

Pricing mechanisms are crucial in influencing user and fog node behavior towards efficient resource utilization [19], [20]. Existing pricing models in fog computing range from fixed pricing, where resources are charged at predetermined rates, to dynamic pricing that considers real-time factors such as resource availability and user demand [21], [22], [23]. However, most pricing mechanisms do not explicitly incorporate strategic decision-making by fog nodes and users. As a result, they may not effectively incentivize efficient resource allocation or achieve fairness among participants [24], [25], [26].

Game theory provides a robust framework for modeling strategic interactions among self-interested entities in fog computing [27], [28]. It enables the analysis of decision-making processes and the design of mechanisms that lead to desirable outcomes. Game theoretical approaches have been applied to resource allocation and pricing problems in cloud computing, but their application to fog computing is relatively nascent. By formulating fog computing resource allocation as a non-cooperative game, researchers have started exploring strategies such as coalitional games, Stackelberg games, and Nash bargaining solutions to optimize resource allocation and pricing [4], [29], [30].

In addition to non-cooperative games, [31]there is growing interest in collaborative resource allocation in fog computing [32]. Joint resource allocation involves fog nodes forming coalitions to jointly optimize resource utilization and achieve better system performance [33], [34]. Cooperative game theory provides tools for analyzing coalition formation and allocation of resources among fog nodes [35]. These approaches aim to strike a balance between individual and collective benefits, leading to more efficient resource utilization and enhanced system performance [7], [36], [37], [38].

Some researchers have proposed hybrid approaches that combine game theory with other optimization techniques, such as machine learning and evolutionary algorithms [39], [40]. These approaches leverage the strengths of different methodologies to address resource allocation and pricing challenges in fog computing. For example, reinforcement learning algorithms can be used to learn optimal resource allocation strategies in a dynamic and uncertain fog environment.

While existing research provides valuable insights into resource allocation and pricing in fog computing, there is a need for more comprehensive and robust approaches that explicitly consider the strategic decision-making of fog nodes and users [1], [2], [41]. The application of game theory offers a promising avenue to address these challenges and optimize resource allocation and pricing mechanisms in fog computing environments [2], [6], [14]. The main goal of our work is to provide an effective dynamic pricing scheme that, when combined with resource allocation, overcomes the constraints of the existing pricing schemes used in the current fog computing deployment. We build upon the framework proposed in reference [42], which aims to maximize social welfare through a value-based economic mechanism. Specifically, we focus on designing an efficient dynamic pricing scheme for users facing time constraints in completing their desired tasks within the specified completion time.

The existing research in fog computing has shed light on resource allocation and pricing mechanisms [3], [43], [44]. However, there is still a need for more comprehensive approaches that consider the strategic decision-making of both fog nodes and users. Game theory has emerged as a promising approach to tackle these challenges and optimize resource allocation and pricing in fog computing environments, as highlighted in [45], [46], and [47].

The main goal is to develop an effective dynamic pricing system that integrates smoothly with resource allocation in fog computing installations. Our goal is to overcome the drawbacks of current pricing models by creating a fresh strategy that maximizes resource efficiency and improves consumer happiness. A framework that emphasized maximizing social welfare through a value-based economic mechanism was put forth in [48], [49], and [50]. Using this framework as a foundation, our work broadens its focus to address the problem of users not finishing their jobs by the deadline. The authors recognized this issue as a possible topic for future development [51], [52], [53].

Specifically, we focus on designing an efficient dynamic pricing scheme that caters to users facing time constraints

in completing their desired tasks within the specified completion time. Our approach aims to enhance fog computing systems' overall performance and efficiency by considering the strategic decision-making of fog nodes and users. We aim to optimize the allocation of resources and the pricing mechanism, considering the time constraints users face. Through our research, we aim to provide a more robust and effective solution for resource allocation and dynamic pricing in fog computing, ultimately improving the user experience and maximizing the utilization of fog computing resources.

A. MOTIVATION AND CONTRIBUTIONS

In the changing world of fog computing, important challenges need attention. These challenges include the diverse nature of fog computing environments, which require methods for allocating resources and determining prices. Additionally, fog computing aims to provide services that call for pricing models that accommodate users' time constraints [46], [54]. Moreover, ensuring resource distribution among users and fog nodes while achieving resource utilization remains complex [47], [50].

To contribute to the progress of fog computing, we propose solutions. Firstly, we introduce a pricing scheme that adjusts dynamically to meet users' needs with time constraints while optimizing resource allocation. This pricing model takes into consideration the varying demands of users [6], [49]. Ensures allocation of resources.

Secondly, we seamlessly integrate the dynamic pricing scheme with resource allocation strategies. By doing we enhance resource utilization [52]. Improve user satisfaction. This integration enables the allocation of resources, leading to enhanced system performance.

Furthermore, we explicitly address decision-making by both fog nodes and users. By incorporating principles from game theory, we enhance the performance of the fog computing system. Decision-making allows for efficient resource allocation while ensuring optimal system operation [51], [53].

We aim to elevate the user's experience by tackling the obstacles in fog computing settings. With our efforts, we aspire to enhance the performance and productivity of fog computing systems, ultimately bringing advantages to individuals operating within these environments [43], [54].

This paper makes several contributions to the field of fog computing:

- **Development of GTRADPMFC Mechanism:** We propose a novel mechanism called Game-Theoretic Resource Allocation and Dynamic Pricing Mechanism in Fog Computing (GTRADPMFC) to address resource allocation and dynamic pricing challenges in fog computing environments with limited resources. GTRADPMFC integrates game theory principles with dynamic pricing strategies to optimize resource utilization and enhance overall system performance.
- **Effective Handling of Completion Time Failures:** GTRADPMFC effectively manages resource allocation by establishing pricing in fog computing, considering potential delays in completion time. This mechanism addresses completion time failures by giving users flexibility when they cannot meet their assignment deadlines within the specified time frame, thereby improving user satisfaction and system efficiency.
- **Theoretical Analysis and Simulation Validation:** We conduct theoretical analysis and simulations to evaluate the performance of GTRADPMFC. Theoretical analysis involves rigorous study of the mechanism's properties, complexity, convergence, and optimization guarantees. Simulation results demonstrate that GTRADPMFC improves resource efficiency and overall fog computing system performance compared to existing solutions.
- **Practical Implications and Future Directions:** We discuss the practical implications of our proposed mechanism and identify future research directions for advancing fog computing systems. GTRADPMFC contributes to developing more robust, efficient, and scalable solutions for fog computing environments by addressing key challenges such as dynamic pricing, completion time failures, and resource allocation.

III. SYSTEM MODEL

The proposed model focuses on resource allocation and pricing structures within the fog computing framework, specifically addressing scenarios where users fail to meet their deadlines and resubmit their resource demands. This aspect ensures efficient resource utilization and meeting users' requirements in fog computing environments.

Considering these resubmitted demands, the proposed model aims to develop a scalable algorithm that effectively allocates resources based on user priorities and constraints. It considers the demand components, such as resource requirements and time limits, and incorporates them into resource allocation and pricing decision-making. The fog service provider possesses a set of n resources, each with a specific capacity. These resources, denoted as $R = (R_1, C_1), (R_2, C_2), \ldots, (R_n, C_n)$, encompass natural components such as RAM, cores, HDDs, and others. In the proposed model, each component is represented by a tuple (R_i, C_i), where R_i denotes the i^{th} resource and C_i represents its corresponding available capacity. These components reflect the resources that are available within the fog computing environment.

The resources can vary depending on the specific requirements and capabilities of the fog computing infrastructure. Examples of resources commonly found in fog computing systems include processing power (CPU cores), memory (RAM), storage (disk space), network bandwidth, and other specialized hardware components.

The tuple (R_i, C_i) represents multiple resources and their respective capacities. This information is essential for effective resource allocation and pricing decisions, as it helps

determine the availability and utilization of resources when fulfilling user demands. A user has the flexibility to select any subset of available resources.

For instance, a user might ask for two resources as $(R_1, \hat{C}_1)$ and $(R_2, \hat{C}_2)$, where $\hat{C}_i$ denotes the amount of capacity the user is asking for the resource R_i in Fig. 1. The requested capacity can either be less than or equal to the available capacity, $\hat{C}_i \leq C_i$, or it can exceed the available capacity, $\hat{C}_i > C_i$. If a user's demand exceeds the capacity of a particular resource, i.e., $\hat{C}_i > C_i$, the request is rejected due to insufficient supply.

Another user may have a demand that includes multiple resources, such as $(R_1, \hat{C}_1)$, $(R_2, \hat{C}_2)$, and $(R_4, \hat{C}_4)$. When a user specifies a demand as a subset of the available resources provided by the fog provider, the proposed system handles the remaining resources not explicitly mentioned in the user's demand as having zero demand.

For example, let's say the fog provider offers three types of resources: RAM (R1), CPU cores (R2), and storage (R3), with available capacities of 8GB, 16 cores, and 1TB, respectively. If a user submits a demand for only RAM and CPU cores, such as (RAM, 4GB) and (cores, 8), the system will consider the demand for storage as zero.

This treatment of zero demand for the remaining resources ensures that the system accurately reflects the user's specific requirements and avoids allocating unnecessary resources not needed for the user's task.

By considering the subset of resources requested by the user and treating the demand for the remaining resources as zero, the proposed system can effectively allocate resources based on the user's specified needs, maximizing resource utilization and optimizing the overall performance of the fog computing system. The sum of the individual demand components in the user's demand vector yields the total unit demand for the i^{th} user, represented as D_i. The total unit demand for a customer whose demand is expressed as $(R_1, 0)$, $(R_2, \hat{C}_2)$, $(R_3, 0)$, $(R_4, \hat{C}_4)$, and $(R_5, 0)$ would be $D_i = 0 + \hat{C}_2 + 0 + \hat{C}_4 + 0$.

The total unit demand for all users, denoted as D, can be calculated by summing the individual user demands as $D = D_1 + D_2 + \ldots + D_n$.

The total unit demand for all users represents the combined resource requirements of all users in the fog computing system, providing an overview of the overall demand for each resource component. This information is valuable for resource allocation and capacity planning processes to ensure efficient utilization of resources and meet the users' demands. The formula for calculating the unit demand is

$$\sum_{k=1}^{j} D_i \cdot C_j \tag{1}$$

where, the actual capacity allotted for a particular component is represented by $D_i \cdot C_j$.

The sum of the individual needs for each user may be used to compute the total unit demand for n users, yielding

$$\sum_{i=1}^{n} \sum_{k=1}^{j} D_i \cdot C_j \tag{2}$$

Thanks to this description process, we can create a scalable algorithm for resource allocation when consumers miss the deadline and resubmit their demands. Each user adds a deadline to their assignment and the resource characteristics. Each user submits a request as a tuple with the values D_i, t_i, t^i, where D_i represents the demand as previously described, t_i indicates the amount of time needed to accomplish the work, and t^i indicates the deadline. For instance, a user may concurrently request (RAM, 2GB) and (cores, 4), with the caveat that the desired task may take t_i time to perform and must be accomplished by the deadline t^i. Each user also offers a valuation for their desire, represented as γ_i, which reflects their highest possible willingness to pay. Consequently, each user request may be represented by the notation $(D_i, t_i, t^i, \gamma_i)$.

This comprehensive representation of user demands, time constraints, and valuations enables the development of effective resource allocation and pricing mechanisms that consider these factors. By incorporating this information into our algorithm, we can efficiently allocate resources and determine appropriate pricing strategies to optimize user satisfaction and system performance.

After determining the needs, allocating resources and determining the cost of each user's demand is necessary. $P = P_1, P_2, \ldots, P_n$ denotes the consumers' pricing vector. The price for the demand D_i will be denoted as $D_i \cdot P_i$, indicating the price of the allocated resources to fulfill that specific demand.

By considering the demands, time limits, valuations, and pricing, a scalable algorithm can be developed to allocate resources effectively and determine appropriate prices, considering the users' demands and willingness to pay.

IV. PROPOSED MECHANISM

This section provides an overview of GTRADPMFC (Game Theoretic Resource Allocation and Dynamic Pricing Mechanism in Fog Computing), followed by a detailed discussion of the methods involved.

A. SKETCH OF GTRADPMFC

GTRADPMFC is designed to address the resource allocation and pricing challenges in fog computing environments, particularly when users fail to meet task completion deadlines and resubmit their demands. The mechanism incorporates game theory principles to optimize resource allocation and pricing decisions.

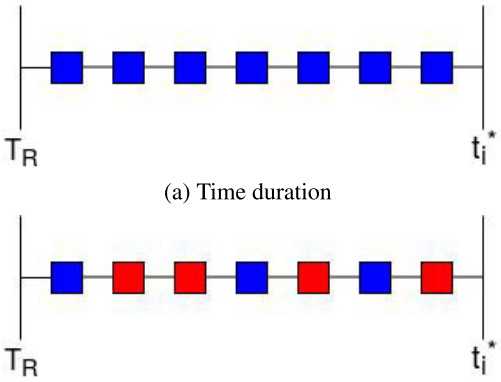
- First, the demands are evaluated before executing the algorithm. accumulated over a set period, say, 30 minutes–and then preserved in a list.

- Next, randomly assign the list of requests and manage them individually in that sequence. All users have an equal opportunity to be handled first since a random list is used. To prevent catching a cold, one can employ an ϵ - Greedy algorithm as an initial approach to address the issue. In that case, we can provide demands made regularly by frequent visitors who drop in. To prevent catching a cold, one can employ an ϵ-Greedy algorithm as an initial approach to address the issue.
- If the original user satisfies the based on the supply, we may allocate them provision; If not, we will reject them. Depending on the supply, the cycle is repeated.
- To determine the pricing for the user being processed right now, we take a long-term look at who has already been processed and gather a representative sample of them. The price computation for an agent (user) is described in detail in Section IV-A1, where a comprehensive formulation is provided. This process enables us to establish a dynamic pricing structure that considers past demand from the initial stages.
- The tentative allocation is verified by checking if the available capacity c_i is less than or equal to the agent's valuation γ_i. If this condition is satisfied, the agent's demand for the current round is accepted, and the price c_i for the i^{th} agent is calculated at this point. However, if the condition is not met, the agent's demand for this round is rejected, and the price calculation is skipped. The following agent in the queue is then processed, which continues iteratively.
- The calculation of c_i is discussed in the upcoming section.

1) PRICE CALCULATION

We examine the deadline t_i^* specified by agent i to calculate the price. Based on this deadline, we determine the time window for agent i, which is defined as follows:

- The beginning of the current time period is indicated by the symbol T_r. Therefore, (T_r, t_i^*) is the window of time for sample collection to calculate the price for the current agent. There are a few occasions where requests are carried out early within this time period. This is considered by making a series of $q = q_1, \ldots, q_{l*}$ random time window selections, as seen in the accompanying picture.
- Now, we choose the red time periods. We received a lot of queries throughout each of these time frames. We gather the winning requests and compile them into a list, $L = L_1, \ldots, L_l*$. The next step is to scan each L_i for requests comparable to the one the present agent is making.
- Similar requests may change based on applications, current demand, the service provider's goals, etc. Say, for instance, that the agent being processed has (6GB, 4 cores). We may consider (6GB, 3 cores), (5GB, 4 cores), and other systems comparable. In this



(a) Time duration

(b) Specified time intervals

FIGURE 1. Interval options.

procedure, we divide L into two distinct lists from all the samples we get. We differentiate between time periods within q that have similar demands and those with different requests using the symbols $L = L^*, L^{**}$.
- To set the price, we adopt a weighted approach that considers the average price of L_1 and L_2. The calculation of the weighted average price is performed as follows.

$$P_i = \left(\frac{\sum_{i=1}^{|L^*|} D_i.P_i}{\sum_{i=1}^{|L^*|} \sum_{j=1}^{k} D_i.C_j} \right) + \gamma \left(\frac{\sum_{i=1}^{|L^{**}|} D_i.P_i}{\sum_{i=1}^{|L^{**}|} \sum_{j=1}^{k} D_i.C_j} \right) \tag{3}$$

Here, $0 < \delta < 1$, and we may adjust it to our goal. For instance, while establishing the agent's price now being processed, $\delta = 0.01$ will give less weight to the different requests, whereas $\delta = 0.5$ would give more weight.

B. DETAILING OF GTRADPMFC

The *Main*() routine, as shown in Algorithm 1, is responsible for collecting and processing user demands.

The first **for loop** collects and stores the demands in the list *list*. Each user is given an equal chance of being processed first by randomly distributing the demands in the list.

The second nested **for loop** iterates through the list *list* one by one. Within this loop, the inner **for loop** is used to check if the current user's demand can be accommodated within the available resources for the duration of their deadline t^*.

If the user's demand cannot be met based on the available resources, the *rest*() function is called, indicating that the user's demand for this round will be denied.

On the other hand, the pricing *Calculation*() subroutine is used to dynamically compute the user's pricing based on the prior demand patterns to see if the user's need can be met.

In Algorithm 2, the function *Price()* determines the price that the current user will pay. When the system starts operating for a failed user, the *if* condition handles the initial prices.

Before proceeding to dynamic pricing, the variable W represents the number of rounds to be completed. The variable $\hat{q}_i$ is a system-generated threshold price, which can

Algorithm 1 Main

```
begin
    Let T_r be current time.
    Let x represent the time from when the collection of demand starts.
    for ∀z ∈ {T_r − x, T_r} do
        list ← list ∪ list_z
    end for
    l ← rand(list)          /* random list */
    for i = 1 to |list| do
        boolean = true
        for k = 1 to |list_i(D_i)| do
            if list_i(D_i · Ĉ_k) ≤ C_k   ∀t ∈ (T_r, t*) then
                C_k ← (C_k − list_i(D_i · Ĉ_k))
            else
                boolean = false
            end if
        end for
        if boolean = false then
            reset()
        end if
        Price()
    end for
    return
end
```

Algorithm 2 Price Calculation

```
The price is first established using the formula ∑∑ D_i · C_i depending on the current demand for several rounds.
if no. of rounds ≤ W then ▷ /* W is set by the system */
    P_i ← P̂_i          ▷ /* P̂_i is system generated threshold price */
else
    s ← rand(T_r, t_i*)
    L* ← process(s)
    L** ← process(s)
    Q_i ← ( ∑_{i=1}^{|L*|} D_i.P_i / ∑_{i=1}^{|L*|} ∑_{j=1}^{k} D_i.C_j ) + δ ( ∑_{i=1}^{|L**|} D_i.P_i / ∑_{i=1}^{|L**|} ∑_{j=1}^{k} D_i.C_j )  ▷ /* set price by Equation 3 */
    if p_i ≤ δ_i then
        final allocation P_i = Q_i
    else
        reject
    end if
end if
return
```

be adjusted dynamically based on the demand from the previous round.

To determine the adjusted threshold price, we can multiply $\hat{q}_i$ by ϵ, where $0 \leq \epsilon \leq 1$, or multiply it by $(1+\epsilon)$, depending on the demand.

The **else** section of the code snippet initially gathers the samples from the time frame (T_r, t_i^*). The samples are collected in s, and then they are divided into two lists: L^* (representing similar requests) and L^{**} (representing different requests). The price for the user is then determined using the equation specified in the code snippet.

The current user's price, denoted as P_i, is checked against its maximum willingness to pay, γ_i, using the condition $P_i \leq \gamma_i$. If this condition is met, the user's final assignment is made. Otherwise, its request is rejected.

C. TIME COMPLEXITY

When all the conditions in the nested loops are met for each element in the list, Algorithm 1 exhibits a linear time complexity of $O(n)$. Conversely, in the worst-case scenario, where none of the conditions are met, and the reset function needs to be called for every element, the time complexity is $O(n \times m)$, and the average-case time complexity remains $O(n \times m)$.

In the best-case scenario, where the number of rounds is equal to or less than a predetermined value W, Algorithm 2 simply assigns a value to P_i and exhibits a constant time complexity of $O(1)$. The worst-case time complexity arises when the number of rounds exceeds W, entailing more intricate calculations. Assuming that the operations within the branches are $O(n)$, the worst-case time complexity is also $O(n)$. Nevertheless, the actual time complexity may fluctuate depending on the distribution of the input data and the complexity of the operations, necessitating specific details for a more precise analysis.

V. ANALYSIS

The predicted number of users that can be assigned from the requests obtained during the time window $T_r - x, T_r$ is first determined using the two probability models. This calculation allows us to assess the efficiency of resource allocation achieved by the proposed algorithm. By determining the expected number of successfully allocated users, we can evaluate how effectively the algorithm utilizes available resources to meet user demands.

Furthermore, we provide information on the accuracy of the proposed algorithm, emphasizing its performance and effectiveness. This assessment evaluates how closely the algorithm's allocation decisions align with user demands. By analyzing the algorithm's accuracy, we gain insights into its ability to make precise resource allocation decisions and efficiently utilize available resources.

Lemma 1: GTRADPMFC are truthful

Proof: We analyze the price calculation process to demonstrate that the i^{th} user cannot benefit from deviating from their actual demand θ_i to a reported demand $\hat{\theta}_i$. For any arbitrary user i, the price calculation is performed as follows.

The system collects the users' demands, including user i's reported demand $\hat{\theta}_i$. Based on the collected demands, the system calculates the prices for each user. This price calculation considers the overall demand pattern and other relevant factors.

The price for user i, denoted as p_i, is determined using the calculated prices.

User i's utility, u_i, is calculated as the difference between their true valuation and the payment, which is given by $u_i = \gamma_i - p_i$.

If user i deviates from its actual demand θ_i to a reported demand $\hat{\theta}_i$, the price calculation and utility calculation are still based on the actual demands of other users. Therefore, the reported demand $\hat{\theta}_i$ does not affect the price calculation or the resulting utility u_i.

As a result, user i's utility with the reported demand, denoted as $\hat{u}_i$, remains the same as the utility with the actual demand: $\hat{u}_i = u_i$.

By following this analysis, we can demonstrate that if the i^{th} user deviates from its actual demand to a reported demand, it cannot benefit since the price calculation and resulting utility remain unaffected by the reported demand.

$P_i \leftarrow \left(\frac{\sum_{i=1}^{|L^*|} D_i.P_i}{\sum_{i=1}^{|L^*|}\sum_{j=1}^{k} D_i.C_j}\right) + \delta\left(\frac{\sum_{i=1}^{|L^{**}|} D_i.P_i}{\sum_{i=1}^{|L^{**}|}\sum_{j=1}^{k} D_i.C_j}\right)$, is independent of γ_i.

If the user's accurate valuation γ_i is revealed, its utility u_i is calculated as the difference between the value and the payment, which is $\gamma_i - P_i$.

If the user deviates and reports a value different from γ_i, its utility $\hat{u}_i$ will not change because the price calculation, which determines the payment to be made by the user, is independent of the reported value for the demand. Consequently, the difference between the user's accurate valuation and the payment remains the same.

Therefore, we have $u_i = \hat{u}_i$, indicating that the user's utility does not change regardless of the reported value. This ensures that the user cannot gain any advantage through deviation from their actual valuation, as its utility remains unaffected. □

Lemma 2: We can calculate the anticipated number of users who will be successfully assigned by taking into consideration the i^{th} user's request fulfillment probability, which is $\frac{1}{i}$. If n is the total number of users, the estimated number of users that will be effectively assigned may be expressed as $\leq \log_2 n + 1$. This constraint shows that the expected number of correctly assigned users rises together with the number of users, albeit at a declining pace. The bound's logarithmic nature shows that as the user base increases, the efficiency of the suggested algorithm in allocating resources increases.

Proof: The number of users who have made demands during the time period from $T_r - x$ to T_r is denoted as l. However, not all of these requests can be fulfilled due to limitations in resource availability.

To determine the number of requests that can be successfully fulfilled, we consider the order in which users are processed from the list l. The probability of a user's request being fulfilled depends on its position in the list, denoted by i. The probability of the i^{th} user's request being fulfilled is calculated as $\frac{1}{i}$.

We establish a random variable S_i for each user's request to keep track of the number of successful requests. The number of successful requests for the i^{th} user is represented by S_i. The sum of all the S_i values may determine the overall number of successful requests.

$$\begin{aligned} S &= S_1 + \cdots + S_n \\ &= \sum_{i=1}^{n} S_i \end{aligned} \tag{4}$$

Taking expectations from both sides, we get

$$\begin{aligned} \mathbb{E}[S] &= \mathbb{E}[\sum_{i=1}^{n} S_i] \\ &= \sum_{i=1}^{n} \mathbb{E}[S_i], \text{ by linearity of expectation} \\ &= \sum_{i=1}^{n} \left(\frac{1}{i} \cdot 1 + \left(1 - \frac{1}{i}\right) \cdot 0\right) \\ &= \sum_{i=1}^{n} \frac{1}{i} \\ &= H_n \\ &\leq log_2 n + 1 \end{aligned} \tag{5}$$

□

The probability model discussed previously could not be helpful if the service provider has a lot of resources to distribute. In such cases, an alternative approach can be adopted.

In order to handle this, we consider the amount of allocations already made and the user who is now being processed (i^{th} user). As more allocations are made, the probability of not being successfully allocated increases. Therefore, we can use $\frac{i}{n}$ to represent this probability.

The interpretation is as follows. When considering the first user, the probability of not being allocated is $\frac{1}{n}$, which is a very small amount. As we progress through the list of users, the probability of not being allocated gradually increases. Based on this interpretation, we may now provide our following lemma, which extends this probabilistic framework.

Lemma 3: We may analyze the anticipated number of users successfully allotted based on the probability model where the likelihood of the i^{th} user's request not being fulfilled is $\frac{i}{n}$.

Proof: The random variables S_i may determine the total number of allocations as follows.

Let N be the total number of users making requests. For the i^{th} user, the probability of its request being successfully allocated is $(1 - \frac{i}{N})$. Therefore, the random variable S_i follows a Bernoulli distribution with probability of success $(1 - \frac{i}{N})$.

To calculate the total number of successful allocations, we sum up the values of S_i for all users.

Total number of successful allocations $= S_1 + S_2 + \ldots + S_N$.

The expected value of this sum can be calculated as

Expected number of successful allocations $= E(S_1 + S_2 + \ldots + S_N)$.

Using the linearity of expectation, this can be rewritten as

Expected number of successful allocations $= E(S_1) + E(S_2) + \ldots + E(S_N)$.

Since each S_i follows a Bernoulli distribution, the expected value of S_i is equal to its probability of success, as given by

Expected number of successful allocations $= (1 - \frac{1}{N}) + (1 - \frac{2}{N}) + \ldots + (1 - \frac{N}{N})$.

Simplifying this expression, we get

Expected number of successful allocations $= N - \frac{1+2+\ldots+N}{N}$.

Using the formula for the sum of consecutive integers, we have

Expected number of successful allocations $= N - \frac{\frac{N(N+1)}{2}}{N}$.

Simplifying further, we get

Expected number of successful allocations $= \frac{N}{2}$. □

VI. SIMULATION AND EXPERIMENTAL RESULTS

This section illustrates a detailed analysis of the experimental simulations and results.

A. SIMULATION SETUP

1) THE PRODUCTION OF SAMPLE DATA

To simulate fog computing resource allocation dynamics, we need to prepare training data using synthetic data generation techniques. The given data structure is used to save information regarding available resources and historical price allocations by fog service providers.

To save time, flat rates are used for the lowest resource requirements. This pricing strategy is periodically changed to reflect ongoing market dynamics. This gradual introduction of cost ensures that the simulation accurately represents the dynamics of fog computing environments.

The simulation uses data that cover a particular timeframe, from the beginning at the first timestamp (0 seconds since epoch) until the end at a finite time endpoint (e.g., 15 seconds since epoch). For the data, there are different resource types: $r1$ represents RAM, $r2$ represents HDD, and $r3$ represents CPU. This hypothesis considers the expense hierarchy: $r3 > r1 > r2$. This hierarchy is an important component for comparing input files with user queries.

Furthermore, Table 1 shows the corresponding availability of resources and pricing structure, which serve as the basis for simulating resource allocation strategies in fog computing.

2) USER REQUESTS

Table 2 lists resource calls submitted by users whose burst times have expired according to the set time limitation. Each demand must be allocated with uninterruptible resources that respond instantly (0 to 30 seconds since epoch). Each demand is reviewed to ensure the availability of resources and pricing set by fog service providers.

TABLE 1. Resource availability and pricing structure.

Resource Type	Availability	Pricing
$r1$ (RAM)	High	Variable
$r2$ (HDD)	Medium	Variable
$r3$ (CPU)	Low	Variable

TABLE 2. User requests.

User ID	Resource Type	Demand
1	$r1$ (RAM)	High
2	$r2$ (HDD)	Low
3	$r3$ (CPU)	Medium

Through a simulation of user requests that mimics actual scenarios and by matching user demands with resource availability, we capture the complexity of interaction between the two elements.

Table 4 represents the resources requested by the users who have exceeded their previous specified burst times. All the requests are the ones accepted in the time frame 0 to 30 seconds since epoch.

3) SYSTEM DETAILS

The algorithm is implemented using Python and runs on a server provided by Google colab. While implementing the algorithm, we have used two libraries, **random** and **matplotlib**, to implement randomization and plot graphs, respectively.

B. SIMULATION OUTLINE

We feed the sample data and the requests to GTRADPMFC. It randomly chooses instances of sample data, bifurcates the randomly chosen data into similar data (L^*) and dissimilar data (L^{**}), and then calculates q_i, $\forall \delta \in \{0, 0.1, 0.2, 0.3\}$. Along with that, the first minimum and the second minimum are also calculated.[1]

Since the first minimum and second minimum are independent of δ, they remain the same all over the simulation. Using eq. 3, p_i for $\delta \leftarrow 0$, is calculated as

$$p_i = \left(\frac{\sum_{i=1}^{|L^*|} D_i.P_i}{\sum_{i=1}^{|L^*|}\sum_{j=1}^{k} D_i.C_j}\right) + 0 \times \left(\frac{\sum_{i=1}^{|L^{**}|} D_i.P_i}{\sum_{i=1}^{|L^{**}|}\sum_{j=1}^{k} D_i.C_j}\right) = \left(\frac{\sum_{i=1}^{|L^*|} D_i.P_i}{\sum_{i=1}^{|L^*|}\sum_{j=1}^{k} D_i.C_j}\right) \quad (6)$$

p_i is similarly similarly for other δ values.

C. RESULTS

It is this part that tackles the examination of how well the GTRADPMFC algorithm does in fog resource allocation management. In our research, we will use a number of

[1]The calculation of minimums includes finding the first and second minimum price from the list of requests in L^* and returning the per unit price of the minima found.

TABLE 3. Previous data.

Resources requested	t_i (s)	t_i^* (seconds since epoch)	Price ($)	Request Time (seconds since epoch)
$[r_1:5, r_2:25, r_3:1]$	10	30	15	10
$[r_1:10, r_2:30, r_3:2]$	10	40	20	10
$[r_1:15, r_2:35, r_3:3]$	10	50	25	10
$[r_1:20, r_2:40, r_3:4]$	10	60	30	10
$[r_1:25, r_2:45, r_3:5]$	10	70	35	10
$[r_1:30, r_2:50, r_3:6]$	10	80	40	10
$[r_1:35, r_2:55, r_3:7]$	10	90	45	10
$[r_1:6, r_2:26, r_3:2]$	10	35	17	15
$[r_1:11, r_2:31, r_3:3]$	10	40	22	15
$[r_1:16, r_2:36, r_3:4]$	10	60	27	15
$[r_1:21, r_2:41, r_3:5]$	10	65	32	15
$[r_1:26, r_2:46, r_3:6]$	10	70	37	15
$[r_1:7, r_2:27, r_3:3]$	10	40	19	20
$[r_1:12, r_2:32, r_3:4]$	10	50	24	20
$[r_1:17, r_2:37, r_3:5]$	10	65	29	20
$[r_1:22, r_2:42, r_3:6]$	10	70	34	20
$[r_1:27, r_2:47, r_3:7]$	10	45	39	20
$[r_1:37, r_2:52, r_3:8]$	10	60	44	20
$[r_1:8, r_2:28, r_3:4]$	10	38	24	25
$[r_1:13, r_2:33, r_3:5]$	10	40	29	25
$[r_1:18, r_2:38, r_3:6]$	10	50	34	25
$[r_1:23, r_2:43, r_3:7]$	10	55	39	25
$[r_1:28, r_2:53, r_3:8]$	10	60	44	25

TABLE 4. Resources requested by the users.

Resources requested	t_i (s)	t_i^* (seconds since epoch)	Price ($)	Request Time (seconds since epoch)
$[r_1:5, r_2:25, r_3:2]$	10	35	15	15
$[r_1:17, r_2:36, r_3:3]$	10	50	35	20
$[r_1:22, r_2:41, r_3:6]$	10	70	50	25
$[r_1:5, r_2:25, r_3:3]$	10	25	20	30
$[r_1:10, r_2:30, r_3:3]$	10	65	30	25
$[r_1:20, r_2:50, r_3:3]$	10	80	60	10
$[r_1:12, r_2:32, r_3:1]$	10	90	20	15

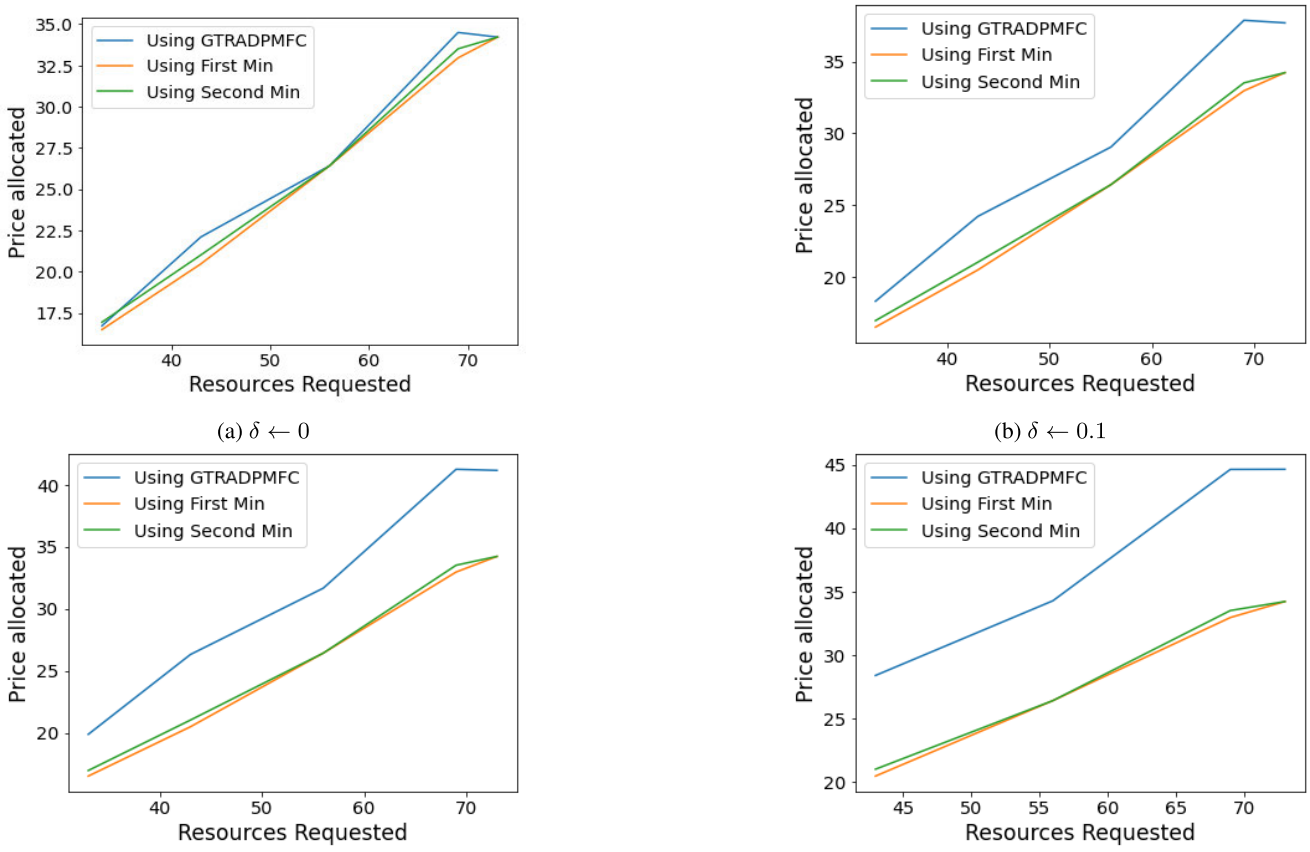
TABLE 5. Final result.

Resources requested	t_i	t_i^*	Price Allocation($)					
			GTRADPMFC				Minima	
	(s)	(s)	$\delta \leftarrow 0$	$\delta \leftarrow 0.1$	$\delta \leftarrow 0.2$	$\delta \leftarrow 0.3$	First Minimum	Second Minimum
$[r_1:5, r_2:25, r_3:2]$	10	35	NA	NA	NA	NA	NA	NA
$[r_1:17, r_2:36, r_3:3]$	10	50	26.42	29.04	31.66	34.28	26.42	26.42
$[r_1:22, r_2:41, r_3:6]$	10	70	34.5	37.87	41.25	44.62	32.96	33.51
$[r_1:5, r_2:25, r_3:3]$	10	25	16.73	18.3	19.87	NA	16.5	16.95
$[r_1:10, r_2:30, r_3:3]$	10	65	22.11	24.21	26.31	28.4	20.48	21.02
$[r_1:20, r_2:50, r_3:3]$	10	80	34.22	37.69	41.16	44.63	34.22	34.22
$[r_1:12, r_2:32, r_3:1]$	10	90	NA	NA	NA	NA	NA	NA

important indicators including execution duration, resource utilization, and cost efficiency. Moreover, we do a comparative summary of the pros and cons of GTRADPMFC with the current systems to highlight the efficiency of GTRADPMFC in real scenarios of fog services.

The evaluation process is done under the execution time of GTRADPMCFC for different data sizes and request loads. Knowing the crossbar switch's computational efficiency is vital for analyzing it at the level of scalability and real amenability to the cloud. On the other side, we, however, consider resource utilization, which is certainly an important aspect of productive fog service use. By examining how R and M DMFC resource allocation roles tuned to the changing load in an instant, we are able to realize how well it stands for meeting user needs and minimum wasting of resources.

Fig. 2 shows the price allocated to the resources requested according to the previous sample data collected. For $\delta \leftarrow 0$, from Fig. 2a, we can infer that the price allocation will be the same in all three scenarios, i.e., GTRADPMFC, First Minimum, and Second Minimum. The price allocated also increases with the increase in the requested resources. The decrease in prices allocated at $\sum_{j=1}^{k} d_i \cdot c_j \leftarrow 73$ is due to



(a) $\delta \leftarrow 0$ (b) $\delta \leftarrow 0.1$

(c) $\delta \leftarrow 0.2$ (d) $\delta \leftarrow 0.3$

FIGURE 2. Price allocated using GTRADPMFC, first minimum, and second minimum.

the request being more storage-centric (i.e., asks for more of HDD than RAM than CPU) unlike for $\sum_{j=1}^{k} d_i \cdot c_j \leftarrow 69$. After that, for $\delta \leftarrow 0.1$, from Fig. 2b, GTRADPMFC provides higher price allocation when compared to First Minimum and Second Minimum. The points at which the graphs of the second minimum and first minimum coincide are the points in which only one previous sample data similar to the request is found. Hence, the first minimum is the same as the second minimum.[2]

For $\delta \leftarrow 0.2$, GTRADPMFC allocates greater prices than the other two right from the start, and so does for $\delta \leftarrow 0.3$. Hence, GTRADPMFC starts proving itself as a profitable option for service providers from $\delta \leftarrow 0.1$ and keeps increasing with the increase of δ.

The downfall of the requests fulfilled in Fig. 3, at $\delta \leftarrow 0.3$, describes that the price proposed by the algorithm has increased beyond the price proposed by the users.

Fig. 4 illustrates the comparative performance of GTRADPMFC (represented by the red line) and FogPrime [41] (represented by the blue line) in a periodic scenario. GTRADPMFC exhibits a performance advantage over FogPrime [41]. Specifically, GTRADPMFC begins with an initial performance score of 85, which is higher than FogPrime's score of 82. This lead is consistently maintained by GTRADPMFC throughout the periods. At the end of the scenario, GTRADPMFC achieves a performance score of 95, while FogPrime [41] scores 89. This highlights the superior performance of GTRADPMFC and establishes it as the preferred choice for this specific scenario.

Fig. 5 illustrates the comparison of space complexity between GTRADPMFC and FogPrime [41], focusing on the memory requirements over time. The red line, representing GTRADPMFC, exhibits an initial space complexity of 80MB, which is more efficient than FogPrime's 90MB. This advantage in space efficiency is consistently maintained throughout the duration of the analysis, with GTRADPMFC concluding at 55MB and FogPrime [41] at 70MB. Fig. 5 provides clear evidence of GTRADPMFC's superior memory efficiency, rendering it the optimal choice for scenarios where memory conservation is a priority.

Fig. 6 compares the performance of GTRADPMFC and FSPs [53] across multiple data points. GTRADPMFC is found to be the better option, with a blue solid line indicating a consistent and impressive improvement in performance from 10 to 20. FSPs [53], represented by a green dashed line, start with a slightly higher performance of 12 but show less significant improvement and more fluctuations. The graph

[2]This wouldn't happen when the sample data increases and becomes denser.

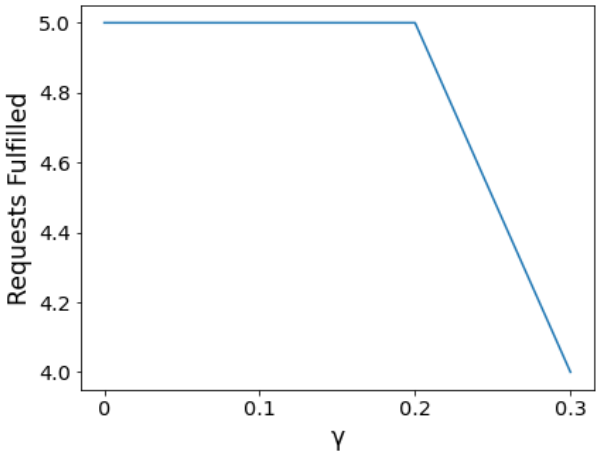


FIGURE 3. Requests fulfilled.

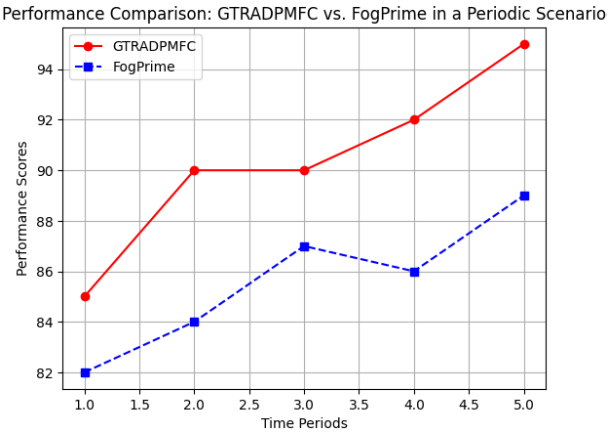


FIGURE 4. Performance difference.

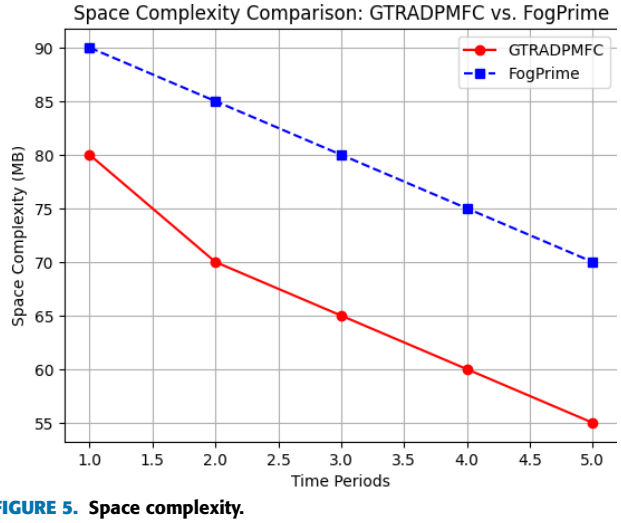


FIGURE 5. Space complexity.

clearly shows that GTRADPMFC consistently outperforms FSPs [53], making it the preferred choice due to its higher and steadily improving performance throughout the data points.

Fig. 7 effectively compares the accuracy performance of GTRADPMFC and FSPs [53] across a series of data points. Notably, GTRADPMFC, denoted by the solid blue

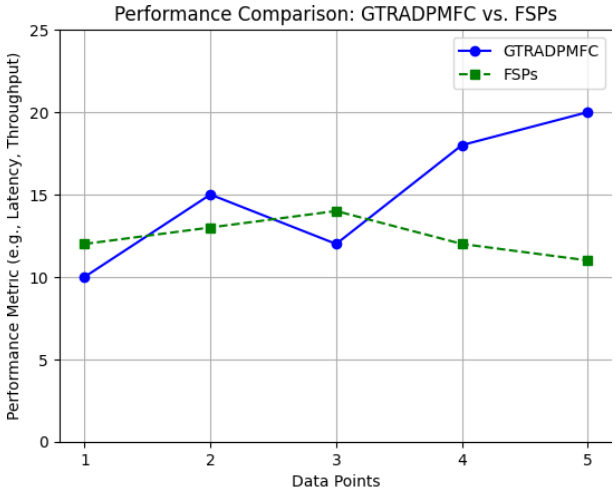


FIGURE 6. Performance difference.

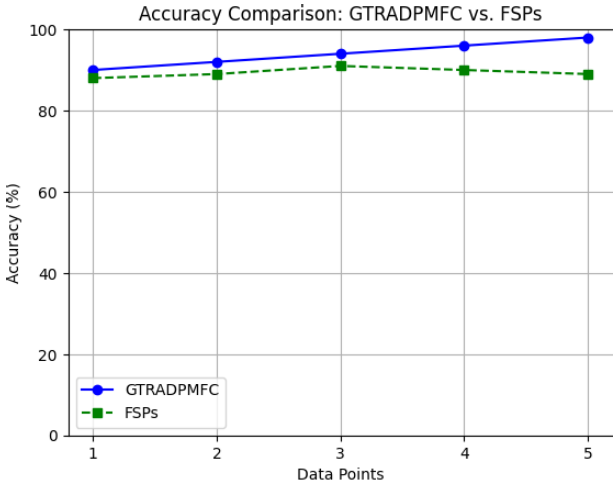


FIGURE 7. Accuracy.

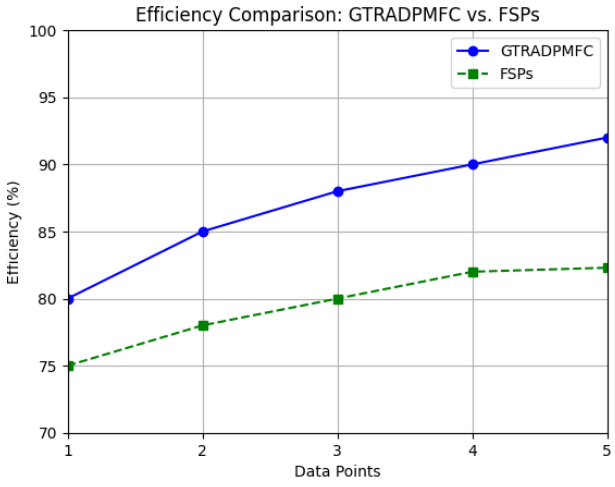


FIGURE 8. Efficiency.

line, starts with a higher initial accuracy of 90% and maintains a consistent upward trend, culminating in an impressive 98% accuracy at the final data point. In contrast,

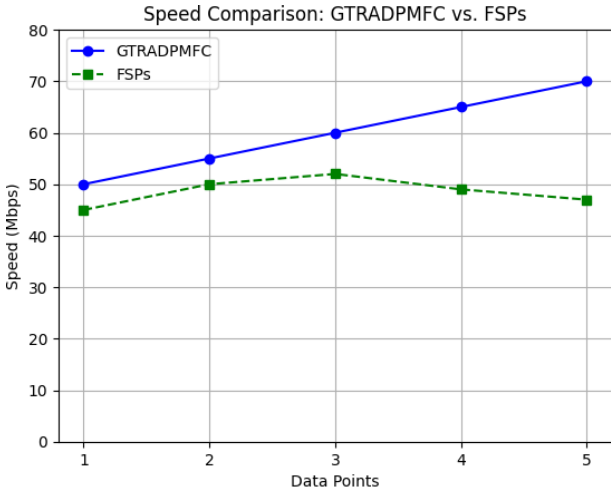


FIGURE 9. Speed.

FSPs [53], represented by the dashed green line, commence with a slightly lower accuracy of 88% and exhibit a lesser steady performance with a final accuracy of 89%. Fig. 7 unambiguously illustrates that GTRADPMFC significantly outperforms FSPs [53] in terms of accuracy, making it the superior choice when precision is paramount in fog computing applications.

Fig. 8 shows that GTRADPMFC is more efficient than FSPs [53]. GTRADPMFC starts at 80% efficiency and steadily increases to 92%, while FSPs [53] have a lesser consistent performance, ending at 77% efficiency. Fig. 8 proves that GTRADPMFC is the better option for optimizing resource utilization and overall effectiveness in fog computing applications.

Fig. 9 shows the speed comparison between GTRADPMFC and FSPs [53] in fog computing. GTRADPMFC initially has a 50 Mbps advantage over FSPs [53], which operate at 45 Mbps. GTRADPMFC consistently improves its speed, while FSPs [53] have more erratic performance. By the final data point, GTRADPMFC reaches 70 Mbps, while FSPs [53] lag behind at 47 Mbps. Fig. 9 clearly shows that GTRADPMFC is the better choice for scenarios where speed is crucial in fog computing.

The study highlights GTRADPMFC's superiority in profitability for service providers, which is evident through significantly higher price allocations than First Min and Second Min. Moreover, GTRADPMFC consistently outperforms competing algorithms such as FogPrime and FSPs across various metrics, including performance, space complexity, accuracy, efficiency, and speed. These findings establish GTRADPMFC as the preferred choice for fog computing applications, offering a comprehensive edge in resource allocation and system performance.

VII. CONCLUSION AND FUTURE WORKS

The research on resource allocation and pricing in fog computing has extensively focused on the payment models between users and Fog service providers. A significant challenge addressed in this paper is providing flexibility to users who cannot complete their tasks within the assigned time frame. To address this challenge, we propose dynamic pricing schemes that enable users to adjust their tasks or extend the completion time to accommodate their needs.

One significant challenge this paper addresses revolves around offering flexibility to users who face difficulties completing tasks within the given time frame. To tackle this issue, we have proposed dynamic pricing schemes that empower users to adjust their tasks or extend completion times according to their needs. Additionally, we have suggested alternative pricing approaches for situations where there are several data samples for accurate pricing determination.

Fog computing devices encounter limitations such as restricted processing power and limited memory and storage capacities. These constraints impede resource allocation and management. Moreover, resource allocation and pricing in fog computing can raise security concerns about data privacy, integrity, and confidentiality. The absence of standardization and interoperability among devices and platforms further complicates resource allocation and pricing.

The dynamic nature of fog computing environments introduces resource allocation and pricing complexity due to system intricacies, workload variability, and network unpredictability. Addressing these challenges by providing solutions aims to enhance resource allocation and pricing mechanisms in fog computing, thus fostering advancements in this field.

REFERENCES

[1] P. Shroff and A. Bandyopadhyay, "A novel matching framework for one-sided markets in fog computing," *Int. J. Comput. Digit. Syst.*, vol. 10, pp. 1–10, Sep. 2020.
[2] A. Bandyopadhyay, V. Kumar Singh, S. Mukhopadhyay, U. Rai, F. Xhafa, and P. Krause, "Matching IoT devices to the fog service providers: A mechanism design perspective," *Sensors*, vol. 20, no. 23, p. 6761, Nov. 2020.
[3] A. Bandyopadhyay, S. Mukhopadhyay, and U. Ganguly, "On free of cost service distribution in cloud computing," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2017, pp. 1974–1980.
[4] L. Mashayekhy, M. M. Nejad, and D. Grosu, "Cloud federations in the sky: Formation game and mechanism," *IEEE Trans. Cloud Comput.*, vol. 3, no. 1, pp. 14–27, Jan. 2015.
[5] M. Babaioff, E. Timnat, Y. Mansour, N. Nisan, G. Noti, C. Curino, N. Ganapathy, I. Menache, O. Reingold, and M. Tennenholtz, "ERA: A framework for economic resource allocation for the cloud," in *Proc. 26th Int. Conf. World Wide web Companion—WWW Companion*, 2017, pp. 635–642, doi: 10.1145/3041021.3054186.
[6] A. Bandyopadhyay, T. S. Roy, V. Sarkar, and S. Mallik, "Combinatorial auction-based fog service allocation mechanism for IoT applications," in *Proc. 10th Int. Conf. Cloud Comput., Data Sci. Eng. (Confluence)*, Jan. 2020, pp. 518–524.
[7] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031–5044, May 2019.
[8] V. Abhishek, I. A. Kash, and P. Key, "Fixed and market pricing for cloud services," in *Proc. IEEE INFOCOM Workshops*, Mar. 2012, pp. 157–162.
[9] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, "Deconstructing Amazon EC2 spot instance pricing," *ACM Trans. Econ. Comput.*, vol. 1, no. 3, pp. 1–20, Sep. 2013, doi: 10.1145/2509413.2509416.
[10] Y. Ai, M. Peng, and K. Zhang, "Edge computing technologies for Internet of Things: A primer," *Digit. Commun. Netw.*, vol. 4, no. 2, pp. 77–86, Apr. 2018.

[11] J. Du, C. Jiang, A. Benslimane, S. Guo, and Y. Ren, "SDN-based resource allocation in edge and cloud computing systems: An evolutionary Stackelberg differential game approach," *IEEE/ACM Trans. Netw.*, vol. 30, no. 4, pp. 1613–1628, Aug. 2022.

[12] J. Du, C. Jiang, J. Wang, Y. Ren, and M. Debbah, "Machine learning for 6G wireless networks: Carrying forward enhanced bandwidth, massive access, and ultrareliable/low-latency service," *IEEE Veh. Technol. Mag.*, vol. 15, no. 4, pp. 122–134, Dec. 2020.

[13] S. Arisdakessian, O. A. Wahab, A. Mourad, H. Otrok, and N. Kara, "FoG-Match: An intelligent multi-criteria IoT-fog scheduling approach using game theory," *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, pp. 1779–1789, Aug. 2020.

[14] A. Bandyopadhyay, F. Xhafa, S. Mallik, P. Krause, S. Mukhopadhyay, V. K. Singh, and U. Maulik, "A framework for allocation of IoT devices to the fog service providers in strategic setting," in *Proc. Int. Conf. P2P, Parallel, Grid, Cloud Internet Comput.*, 2019, pp. 340–351.

[15] A. Bandyopadhyay, F. Xhafa, S. Mukhopadhyay, V. K. Singh, and A. Sharma, "An auction framework for DaaS in cloud computing and its evaluation," *Int. J. web Grid Services*, vol. 15, no. 2, p. 119, 2019.

[16] A. Botta, W. Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and Internet of Things: A survey," *Future Generat. Comput. Syst.*, vol. 56, pp. 684–700, Mar. 2016.

[17] S. Aknine, N. Bouchareb, and N. E. Zarour, "Resource management policies to increase provider's gain in a cloud coalition," *Int. J. Grid Utility Comput.*, vol. 7, no. 3, p. 163, 2016.

[18] A. Brogi and S. Forti, "QoS-aware deployment of IoT applications through the fog," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1185–1192, Oct. 2017.

[19] S. Chawla, N. R. Devanur, A. E. Holroyd, A. R. Karlin, J. B. Martin, and B. Sivan, "Stability of service under time-of-use pricing," in *Proc. 49th Annu. ACM SIGACT Symp. Theory Comput.*. New York, NY, USA: Association for Computing Machinery, Jun. 2017, pp. 184–197, doi: 10.1145/3055399.3055455.

[20] W. Chen, I. Paik, and Z. Li, "Cost-aware streaming workflow allocation on geo-distributed data centers," *IEEE Trans. Comput.*, vol. 66, no. 2, pp. 256–271, Feb. 2017.

[21] P. Cong, G. Xu, T. Wei, and K. Li, "A survey of profit optimization techniques for cloud providers," *ACM Comput. Surv.*, vol. 53, no. 2, pp. 1–35, Mar. 2020, doi: 10.1145/3376917.

[22] N. Dimitri, "Pricing cloud IaaS computing services," *J. Cloud Comput.*, vol. 9, no. 1, p. 14, Dec. 2020.

[23] Y. Ding, K. Li, C. Liu, and K. Li, "A potential game theoretic approach to computation offloading strategy optimization in end-edge-cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 6, pp. 1503–1519, Jun. 2022.

[24] G. Feng and R. Buyya, "Maximum revenue-oriented resource allocation in cloud," *Int. J. Grid Utility Comput.*, vol. 7, no. 1, p. 12, 2016.

[25] M. R. Habes and H. B. Souici, "Towards a fairer negotiation for dynamic resource allocation in cloud by relying on trustworthiness," *Int. J. Grid Utility Comput.*, vol. 8, no. 3, p. 185, 2017.

[26] C. Jiang, Y. Chen, Q. Wang, and K. J. R. Liu, "Data-driven auction mechanism design in IaaS cloud computing," *IEEE Trans. Services Comput.*, vol. 11, no. 5, pp. 743–756, Sep. 2018.

[27] I. Lee, "Pricing schemes and profit-maximizing pricing for cloud services," *J. Revenue Pricing Manage.*, vol. 18, no. 2, pp. 112–122, Apr. 2019, doi: 10.1057/s41272-018-00179-x.

[28] N. C. Luong, Y. Jiao, P. Wang, D. Niyato, D. I. Kim, and Z. Han, "A machine-learning-based auction for resource trading in fog computing," *IEEE Commun. Mag.*, vol. 58, no. 3, pp. 82–88, Mar. 2020.

[29] H. Madiha, L. Lei, A. A. Laghari, and S. Karim, "Quality of experience and quality of service of gaming services in fog computing," in *ICMSS*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 225–228.

[30] P. Modisane and O. Jokonya, "Evaluating the benefits of cloud computing in small, medium and micro-sized enterprises (SMMEs)," *Proc. Comput. Sci.*, vol. 181, pp. 784–792, Oct. 2021.

[31] S. F. Abedin, A. K. Bairagi, Md. S. Munir, N. H. Tran, and C. S. Hong, "Fog load balancing for massive machine type communications: A game and transport theoretic approach," *IEEE Access*, vol. 7, pp. 4204–4218, 2019.

[32] A. K. Bairagi, S. F. Abedin, N. H. Tran, D. Niyato, and C. S. Hong, "QoE-enabled unlicensed spectrum sharing in 5G: A game-theoretic approach," *IEEE Access*, vol. 6, pp. 50538–50554, 2018.

[33] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 416–464, 1st Quart., 2018.

[34] T. Ni, Z. Chen, L. Chen, S. Zhang, Y. Xu, and H. Zhong, "Differentially private combinatorial cloud auction," *IEEE Trans. Cloud Comput.*, vol. 11, no. 1, pp. 412–425, Jan. 2023.

[35] T. M. Ho, N. H. Tran, S. M. Ahsan Kazmi, and C. S. Hong, "Dynamic pricing for resource allocation in wireless network virtualization: A Stackelberg game approach," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2017, pp. 429–434.

[36] A. C. Oliveira, C. Fetzer, A. Martin, D. Le Quoc, and M. Spohn, "Optimizing query prices for data-as-a-service," in *Proc. IEEE Int. Congr. Big Data*, Jun. 2015, pp. 289–296.

[37] C. Pahl, A. Brogi, J. Soldani, and P. Jamshidi, "Cloud container technologies: A state-of-the-art review," *IEEE Trans. Cloud Comput.*, vol. 7, no. 3, pp. 677–692, Jul. 2019.

[38] G. Portella, E. Nakano, G. N. Rodrigues, and A. C. M. A. Melo, "Utility-based strategy for balanced cost and availability at the cloud spot market," in *Proc. IEEE 12th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2019, pp. 214–218.

[39] H. Zhang, H. Jiang, B. Li, F. Liu, A. V. Vasilakos, and J. Liu, "A framework for truthful online auctions in cloud computing with heterogeneous user demands," *IEEE Trans. Comput.*, vol. 65, no. 3, pp. 805–818, Mar. 2016.

[40] R. Zhou, Z. Li, C. Wu, and Z. Huang, "An efficient cloud market mechanism for computing jobs with soft deadlines," *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 793–805, Apr. 2017.

[41] S. C. Misra and A. Mondal, "FogPrime: Dynamic pricing-based strategic resource management in fog networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 8, pp. 8227–8236, Aug. 2021.

[42] A. Bandyopadhyay, U. Datta, A. Banik, P. Biswas, and V. Sarkar, "Geopositioning of fog nodes based on user device location and framework for game theoretic applications in an fog to cloud network," in *Recent Trends in Computational Intelligence Enabled Research*. New York, NY, USA: Academic, 2021, pp. 233–244.

[43] A. Sinha, V. Mishra, A. Bandyopadhyay, S. Swain, and S. Chakraborty, "Fair resource allocation in fog computing by using a game theoretic approach," in *Proc. Int. Conf. Data Anal. Insights*, 2023, pp. 125–134.

[44] A. Bandyopadhyay, S. Mukhopadhyay, and U. Ganguly, "Allocating resources in cloud computing when users have strict preferences," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2016, pp. 2324–2328.

[45] A. Bandyopadhyay, F. Xhafa, and S. Mukhopadhyay, "An auction framework for DaaS in cloud computing," in *Advances in Internet, Data web Technologies*. Cham, Switzerland: Springer, 2018, pp. 732–741.

[46] A. Bandyopadhyay, V. K. Singh, S. Mukhopadhyay, U. Rai, and A. Bandyopadhyay, "An efficient framework for resource allocation and dynamic pricing scheme for completion time failure in cloud computing," in *Advances in Networked-Based Information Systems*. Cham, Switzerland: Springer, 2022, pp. 143–153.

[47] M. Peng, S. Yan, K. Zhang, and C. Wang, "Fog-computing-based radio access networks: Issues and challenges," *IEEE Netw.*, vol. 30, no. 4, pp. 46–53, Jul. 2016.

[48] G. Zhang, F. Shen, Z. Liu, Y. Yang, K. Wang, and M.-T. Zhou, "FEMTO: Fair and energy-minimized task offloading for fog-enabled IoT networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4388–4400, Jun. 2019.

[49] N. Chen, Y. Yang, T. Zhang, M.-T. Zhou, X. Luo, and J. K. Zao, "Fog as a service technology," *IEEE Commun. Mag.*, vol. 56, no. 11, pp. 95–101, Nov. 2018.

[50] G. Zhang, F. Shen, Y. Yang, H. Qian, and W. Yao, "Fair task offloading among fog nodes in fog computing networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[51] K. Wang, Y. Tan, Z. Shao, S. Ci, and Y. Yang, "Learning-based task offloading for delay-sensitive applications in dynamic fog networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11399–11403, Nov. 2019.

[52] N. Chen, Y. Yang, J. Li, and T. Zhang, "A fog-based service enablement architecture for cross-domain IoT applications," in *Proc. IEEE Fog World Congr. (FWC)*, Oct. 2017, pp. 1–6.

[53] Y. Jie, C. Guo, K. R. Choo, C. Z. Liu, and M. Li, "Game-theoretic resource allocation for fog-based industrial Internet of Things environment," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3041–3052, Apr. 2020.

[54] M. Aazam and E.-N. Huh, "Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT," in *Proc. IEEE 29th Int. Conf. Adv. Inf. Netw. Appl.*, Mar. 2015, pp. 687–694.

ANJAN BANDOPADHYAY received the M.Tech. degree in information security from the Department of Information Technology, NIT, Durgapur, West Bengal, India, and the Ph.D. degree from NIT. He is an Assistant Professor with the Kalinga Institute of Industrial Technology, Bhubaneswar, Odisha, India. In Visvesvaraya, he received the Ph.D. Fellowship under MHRD. He has published many conference and journal articles in esteemed conferences and journals. He is broadly interested in algorithmic game theory (mechanism design). His research interests include cloud computing, fog computing, metaverse, the IoT, healthcare, crowd sourcing, and image processing. He has bagged several Best Paper Awards in many conferences, such as 3PGCIC. He is also a reviewer of many journals and conferences.

SUJATA SWAIN received the B.Tech. degree in computer science and engineering from BPUT University, India, and the M.Tech. and Ph.D. degrees in computer science and engineering from the Indian Institute of Technology Roorkee, India. She is an Assistant Professor with the School of Computer Science and Engineering, Kalinga Institute of Industrial Technology (Deemed to be University), Bhubaneswar. Her research interests include service oriented computing, metaverse, medical imaging, and healthcare systems.

RAJ SINGH is currently pursuing the B.Tech. degree with the Kalinga Institute of Industrial Technology, Bhubaneswar, with a profound passion for cutting-edge technologies. His research interests include cloud and fog computing, the Internet of Things, machine learning, and game theory.

PRITAM SARKAR is currently pursuing the B.Tech. degree with the Kalinga Institute of Industrial Technology, Bhubaneswar, and possesses a deep enthusiasm for state-of-the-art technologies. His research pursuits revolve around cloud and fog computing, the Internet of Things, machine learning, and game theory.

SIDDHARTHA BHATTACHARYYA (Senior Member, IEEE) received the first bachelor's degree in physics and the second bachelor's and master's degrees in optics and optoelectronics from the University of Calcutta, Kolkata, India, in 1995, 1998, and 2000, respectively, and the Ph.D. degree in computer science and engineering from Jadavpur University, Kolkata, in 2008.

He is currently a Senior Researcher with the VSB Technical University of Ostrava, Ostrava, Czech Republic. In addition to this, he is also a Scientific Advisor with the Algebra University College, Zagreb, Croatia. Prior to this, he was the Principal of Rajnagar Mahavidyalaya, Birbhum, India, and the RCC Institute of Information Technology, Kolkata; a Professor with the Department of Computer Science and Engineering, CHRIST (Deemed to be University), Bengaluru, India; and a Senior Research Scientist with the Faculty of Electrical Engineering and Computer Science, VSB Technical University of Ostrava. He has coauthored six books and co-edited 100 books and has authored or coauthored more than 400 research publications in international journals and conference proceedings. He holds five patents. His research interests include soft computing, pattern recognition, multimedia data processing, hybrid intelligence, social networks, and quantum computing. He is a member of FRSA (U.K.), FIET (U.K.), FIEI (I), FIETE, LFOSI, SMIEEE, SMACM, SMAAIA, and SMIETI.

LEO MRSIC received the Diploma degree in insurance from the Department of Foreign Trade, the master's degree in business economics from the IT Department, Faculty of Economics, Zagreb, and the Doctorate of Social Sciences in graph theory and the application of statistical models in business from the Department of Information Sciences, Faculty of Humanities and Social Sciences, Zagreb.

He is an expert/a scientist with proven experience in managing companies and teams (micro teams to large corporations) and a strong orientation on digital innovation. He has led and participated in more than 150 projects. He has particularly emphasized orientation on strategy, planning, and efficient business use of technical/ICT capacities, evaluation, and creation of business value and impact from the use of technologies in business. He is a holder of academic title as an Associate Professor in the field of information sciences (5.04/HR), an Associate Professor in the field of informatics (5.13/SLO), and an Assistant Professor in the field of computing (2.07/SLO); and a teaching title as a Professor of professional studies with tenure in the field of economics (5.01/HR). He is a permanent court expert in the fields of finance, accounting, bookkeeping, and informatics (12 years) with a large number (more the 150) of successfully completed complex expertise procedures. He is an IPMA A Certified Project Director with more than 100 successfully completed complex projects. He is the Vice-Dean for Science and Research with the Algebra University College, Zagreb; the Director of Algebra LAB Research and Innovation Center; the Head of the Digital Transformation Center, the Rudolfovo Public Research Center (SLO), and the BDV i-Silver Data Center, Algebra LAB; a mentor on several incubator programs (Algebra LAB and Uplift); and the Head of the Data Science Study Program, Algebra University College, with a delivered total of more than 4,500 equivalent working hours of teaching. He is an Associate and a Mentor with the Faculty of Information Studies, Novo Mesto; the Faculty of Social Studies, Nova Gorica; the Faculty for Media, Ljubljana; the Rudolfovo Public Research Center, Slovenia; and the Police Academy, Ministry of the Interior of the Republic of Croatia.

Dr. Mrsic is a member of the National Council for Higher Education, Science and Technological Development. He is a consultant on projects in the region related to strengthening education capacity (MRMS and CEP) and the projects to attract subsidies/funds from local (CES) and international investment programs (competence building and innovation). He is a registered Consultant with GOPA Consulting Group Germany, for the areas of business consulting, application of analytical/statistical methods, analysis of the labor market, and support to educational policy making. Since 2022, he has been an official observer/an expert of labor market analysis and education as part of MSWG with the European Commission in Brussels.