Received April 15, 2021, accepted May 26, 2021, date of publication June 23, 2021, date of current version July 12, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3091844

Quality-of-Experience-Aware Incentive Mechanism for Workers in Mobile Device Cloud

SAJEEB SAHA[1], (Senior Member, IEEE), MD. AHSAN HABIB[1], (Senior Member, IEEE), TAMAL ADHIKARY[1], (Member, IEEE), MD. ABDUR RAZZAQUE[1,2], (Senior Member, IEEE), MD. MUSTAFIZUR RAHMAN[1], METEB ALTAF[3], AND MOHAMMAD MEHEDI HASSAN[4], (Senior Member, IEEE)

[1]Green Networking Research Group, Department of Computer Science and Engineering, University of Dhaka, Dhaka 1000, Bangladesh
[2]Department of Computer Science and Engineering, Green University of Bangladesh, Dhaka 1207, Bangladesh
[3]Advanced Manufacturing and Industry 4.0 Center, King Abdulaziz City for Science and Technology, Riyadh 11442, Saudi Arabia
[4]Research Chair of Smart Technologies, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia

Corresponding author: Mohammad Mehedi Hassan (mmhassan@ksu.edu.sa)

This work was supported by the Deanship of Scientific Research at King Saud University through the Vice Deanship of Scientific Research Chairs, Chair of Smart Technologies.

ABSTRACT Mobile device cloud (MDC) is a collaborative cloud computing platform over which neighboring smart devices form an alliance of shared resources to mitigate resource-scarcity of an individual user device for running compute-intensive applications. A major challenge of such a platform is maximizing user quality-of-experience (QoE) at minimum cost while providing attractive incentives to workers' mobile devices. In state-of-the-art works, either a voluntary task execution or merely resource-cost driven mechanism has been applied to minimize the task execution time while overlooking payment of any additional incentive to the worker devices for their quality services. In this paper, we develop a computational framework for MDC where the afore-mentioned challenging problem is formulated as a multi-objective linear programming (MOLP) optimization function that exploits reverse-auction bidding policy. Due to the NP-hardness of MOLP, we offer two greedy worker selection algorithms for maximizing user QoE or minimizing execution cost. In both algorithms, the amount of incentive awarded to a worker is determined following the QoE offered to a user. Theoretical proofs of desirable properties of the proposed incentive mechanisms are presented. Simulation results illustrate the effectiveness of our incentive algorithms compared to the state-of-the-art approaches.

INDEX TERMS Incentive mechanism, mobile device cloud, quality of experience, reverse auction, user satisfaction.

I. INTRODUCTION

With huge advances in recent years, mobile devices (e.g., smartphones, smartwatches, and tablets) have become ubiquitous and are rapidly growing as a dominant computing platform for users. These devices, which are equipped with a plethora of embedded sensors (e.g., GPS, camera, audio, proximity, and temperature), facilitate the running of many compute-intensive mobile applications such as intelligent transportation, natural language translation, augmented reality, and real-time health monitoring [1], [2]. Although the divergence of mobile applications is increasing every day, the execution performance is not yet sufficient due to resource constraints, mainly in terms of limited CPU, memory and battery capacity [3], [4]. To enhance the computation performance of mobile applications, some researchers have introduced mobile device cloud (MDC) [5]–[8] which is an opportunistic computation offloading technology that exploits idle resources of stationary mobile devices; it is also sometimes referred to as Mobile Ad-hoc Cloud [9], [10]. Mobile devices

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

in a large scale stationary location (e.g., theater, shopping mall, stadium, or restaurant) or onboard a vehicle (e.g., bus, train, airplane) may collaboratively form a cloud service infrastructure to perform text translation, healthcare data processing, reality augmentation, etc. in real-time. A study shows that mobile devices are kept in idle state approximately 89 % of the time, and during this period they consume not more than 11 % of the available system resources [11]. The idle resources from such a plethora of mobile devices present untapped computing opportunities [12]. The MDC technology not only mitigates scarcity of computation resources in cloudlet-based offloading mechanisms [3], [13], [14] but also resolves communication latency of remote clouds [1], [15], [16].

Typically, an owner of an application (i.e., a buyer device) offloads an application code to the MDC manager, and then it is executed by different worker devices (i.e., sellers) having a sufficient amount of idle resources. It is important to ensure participation of nearby worker devices in such computation system so as to exploit unused resources efficiently. In state-of-the-art works, authors have focused on designing an MDC framework, where the worker devices participate voluntarily in the task execution process [6], [7], [12]. However, these methods lack to attract a good number of reliable and resource-rich workers in the resource-trading mechanism. To overcome this problem, auction mechanisms have been introduced to select resource-rich workers that minimize user cost [17], [18] or maximize workers' profit [19]–[21]. However, these works suffer from providing quality execution through reliable workers. Moreover, it is still unexplored to investigate the impact of user Quality-of-Experience (QoE) for executing tasks in an MDC environment. The QoE metric quantifies improvement of execution time of an offloaded task observed by the user. Focusing only on minimizing execution cost leads selection of unreliable and/or poor workers, hampering the user QoE. On the other hand, a worker device expects payment in return of task execution that acts as a compensation for dynamic usage of resources (e.g., CPU, memory, and energy) and bandwidth charges. Furthermore, an application user (i.e., buyer) might be motivated to pay more only if s/he receives high quality execution supports from worker devices. Hence, it is a pre-eminent concern to design an incentive mechanism to enhance worker participation, while considering the resource capacity and reputation of worker devices to execute the tasks with the aim of increasing user quality-of-experience (QoE).

In this paper, we focus on the selection of reputed and resourceful worker devices to execute tasks of an application and incentivize them based on the quality of execution. We consider an MDC system consisting of a cloudlet acting as a cloud broker and a set of participating mobile devices (i.e., user and workers), as shown in Fig. 1. The user device has an application (with a set of individual tasks) that requires offloading to the cloudlet for execution. After getting an announcement from the cloudlet, a worker device expresses its willingness to execute a certain task/tasks. Following the

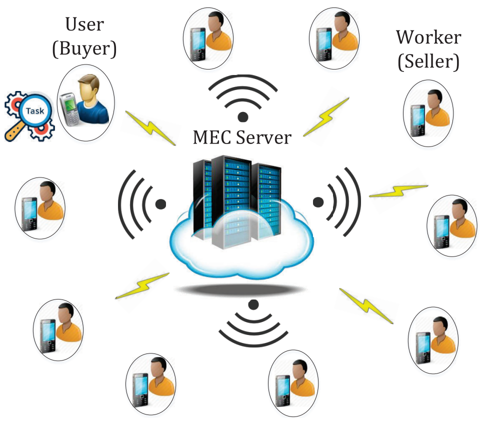


FIGURE 1. Collaborative computation in a mobile device cloud.

reverse-auction bidding policy, the cloudlet then determines an optimal mapping of the tasks to be executed on the worker devices so as to maximize user QoE and minimize execution cost. After successful execution of the assigned tasks, claimed cost with additional incentives according to execution quality are paid to the worker devices. The key contributions of this work are summarized as follows:

- We design a framework for a QoE-aware incentive mechanism to execute applications by workers in MDC. To the best of our knowledge, this is the first work to improve the user-QoE through incentivizing worker devices in addition to their regular bid payment, according to task execution quality.
- We formulate a multi-objective linear programming (MOLP) function that determines the optimal provisioning of application tasks on high performing worker devices with the aim of increasing user QoE with reduced cost.
- Due to the NP-hardness of MOLP, we develop two greedy task-worker assignment algorithms and incentive mechanisms to facilitate resource sharing using reverse-auction theory.
- The novelty of this work lies in paying additional incentives to the workers following their offered qualities of user task execution.
- The reliability and trustworthiness of the workers and the correctness of the incentive mechanisms have been proved theoretically.
- The performances of the proposed incentive mechanisms were evaluated in MATLAB [22], and significant improvements in user QoE and cost reduction were demonstrated.

The remainder of this paper is organized as follows. Section II presents state-of-the-art works in MDC and incentive mechanisms. The computation framework is presented along with our assumptions in Section III. Section IV formulates the worker device selection and incentive disbursement methods and presents a theoretical analysis. Section V presents the simulation environment and experimental results of the proposed incentive mechanisms with comparative analysis. Finally, the paper is concluded in Section VI.

II. RELATED WORKS

The key idea of this work is to provide an incentive to MDC workers according to execution quality of the tasks rendered by them. However, state-of-the-art-works in the literature have only disjointly addressed these two issues, as described below.

A. TASK EXECUTION IN MDC

The increasing number of smartphones has brought an enormous opportunity for code offloading to nearby mobile devices. A significant number of recent works have demonstrated the benefits of exploiting idle resources of nearby mobile devices compared to executing applications on remote clouds. To minimize the task completion time Habak *et al.* proposed FemtoCloud [23], which is a dynamic and self-configuring cluster head-based MDC architecture coordinated by a cloudlet. The improvement was achieved by applying priority-based task assignment and an earliest deadline heuristic on task assignment and result collection, respectively. Mao *et al.* [24] formulated a Lyapunov optimization-based dynamic computation offloading algorithm exploiting the resources of nearby energy harvesting devices to minimize the task execution latency and task failure of offloaded application tasks. However, these works incurred a huge burden on the potential workers due to biased assignment of tasks.

To optimize the application's execution performance, Le *et al.* provided a collaborative infrastructure among nearby mobile devices by leveraging Wi-Fi Direct technology that not only minimizes energy consumption but also expands the hardware capability of mobile devices [8]. The simulation results also reveal that task execution with nearby mobile devices is more beneficial compared to remote cloud server-based execution due to higher communication latency of cellular networks.

Balasubramanian and Karmouch [9] exploited the resources of nearby mobile devices to provide Infrastructure as a Service in the Mobile Ad-hoc Cloud Computing environment. They also outlined necessary architecture and algorithms to create a pool of devices with dedicated resources and efficient utilization of those resources. Later the authors formulated a model to ensure the parallel execution of tasks with a minimum number of worker devices [10]. A composition score was calculated based on the shared resources of the device (CPU, Memory, and Storage) to guarantee the selection of the best available devices for task execution.

Lin *et al.* [25] proposed a code offloading framework named Circa that exhibits the feasibility of code offloading in the proximity of nearby mobile devices exploiting iBeacons, a wireless location-based transmitter system. The system used an efficient and fair task allocation algorithm to disseminate tasks of an application among reliable worker devices.

In [26], Guiguis *et al.* delineated transient clouds, a collaborative computing framework for the offloaded execution of tasks with the help of nearby mobile devices. The authors explored centralized and distributed approaches to allocate tasks among mobile devices according to their capabilities. A modified Hungarian algorithm had been introduced in the centralized approach to balance the workload and Distributed Hash Tables were used to minimize the communication cost in the distributed mechanism. To execute the application tasks of multiple users, Ning *et al.* [27] considered both cloud and edge devices to offload application tasks for minimizing execution delay of latency-sensitive applications and formulated a MILP problem considering the resource contention among the mobile users. In [28], Fernando *et al.* employed a preemptive work-stealing mechanism on a set of worker nodes to balance workload and minimize execution time. Considering user mobility, task properties, resources and energy constraints, Saleem *et al.* [29] proposed a D2D-enabled task assignment and power allocation mechanism to minimize the task execution latency in MEC. To tradeoff between energy consumption and service latency, Yadav *et al.* [30] proposed an energy-efficient computation offloading and resource allocation mechanism considering vehicular node mobility and end-to-end latency deadline in Vehicular Fog Computing. However, the models completely ignores the reliability of the edge worker devices.

All the above-mentioned works significantly minimized execution time by offloading tasks to nearby mobile devices. Although these works focused on minimizing execution cost based on worker' bids and user budget, their worker selection strategies completely ignored user QoE. Moreover, none of these considered any incentive for the worker devices for executing the tasks.

B. INCENTIVE MECHANISM IN MDC

The actual benefit of the emerging MDC technology can only be utilized through effective participation of mobile worker devices in the computation process. Introduction of a reward mechanism can incentivize the use of mobile device resources, increasing motivation for worker devices. In the literature, mobile cloud-based computing systems have exploited a reverse auction mechanism, where a buyer places a request for a service and many sellers bid the minimum value they must be paid for the service, and the bidder with the lowest offer wins the auction.

In [6], Miluzzo *et al.* provided an outline of an incentive mechanism for resourceful nearby mobile devices that run compute-intensive offloaded tasks of an application in an MDC system. Though the incentive mechanism considered execution cost parameters, such as waiting time and bandwidth usage, few other influential parameters, such as worker reputation, workload, and task interdependency, were not considered in system design. On the contrary, Noor *et al.* [17] proposed a task allocation strategy based on worker reputation to assign tasks to different worker devices. In [31], Yadav and Zhang proposed an adaptive energy-aware heuristic algorithm to detect overloaded hosts and a dynamic VM selection algorithm to minimize the energy consumption and to maximize the quality of service in a mobile cloud computing environment. However, they neither provided any incentive to reputed workers in the system, nor did they consider QoE.

To incentivize worker devices, Duan *et al.* proposed a homogeneous and a heterogeneous reward models for data acquisition and distributed computation applications, respectively [19]. Data acquisition applications exploited a Stackelberg game model and distributed computation applications applied a procurement auction mechanism to incentivize individual workers. In [18], Wang *et al.* considered a game theoretic approach to find an equilibrium point for user cost and worker profit for an optimal allocation of tasks in worker devices. Though the system gives benefits to both buyer and seller devices, it aims to minimize user cost in the context of remote cloud price, which is often not practical. Considering resource capacity and task heterogeneity, Wang *et al.* proposed two winning bid determination algorithms for heterogeneous and homogeneous task models, respectively to execute the offloaded tasks in an MDC environment [20]. However, the bid winning worker devices were paid with the immediate next bid of the corresponding task, leading to overpayment of the workers. Later, Tang *et al.* [21], proposed a broker-based double-sided bidding mechanism to provide incentives to worker devices. The authors presented two game theoretic algorithms to find an equilibrium point that maximized the benefits for both user and worker devices. However, none of the works appraised task dependency and reputation of worker devices to execute tasks.

To encourage the participation of cloud service operators and local edge servers for computational offloading in mobile edge computing Liu *et al.* formulated a Stackelberg game that maximizes the utilities of individual entities [32]. Wang *et al.* proposed an auction model to increase participation of mobile devices that trades resources between task owners and worker devices [33]. Resource allocation and price estimation were determined through a distributed algorithm, while a payment evaluation procedure detected dishonest sellers in the system. Because bids were submitted privately to the selected participants, the optimal result could not be guaranteed from the system.

To stimulate service provisioning by edge clouds, Wang *et al.* utilized market-based pricing model to design a multi-round auction mechanism for the resource trading between edge clouds and mobile devices that incentivizes edge clouds for the allocated resources [34]. In [35], Li and Cai considered the collaborative task offloading problem as a social welfare maximization problem and applied a prime-dual framework to develop an online incentive mechanism for the execution of offloaded tasks. Considering the dynamic participation nature of the users and worker mobile devices, He *et al.* proposed an auction-based incentive mechanism for the execution of offloaded tasks that tried to optimize the long-term system welfare without future information of tasks [36]. However, none of the systems take into account the reliability of the edge devices/clouds in the offloading decision process that made the system vulnerable to the successful execution of application tasks.

All the above works focused on a general objective to minimize execution time and cost of a user where the workers were compensated with the bid amount. There were no consideration of additional payment as incentive following quality execution of a task. Moreover, most of the works not only ignored the reliability of devices while selecting the workers but also neglected the inter-dependency among the tasks. The key philosophy of this work is to select a set of reliable worker devices to execute an application task considering its dependency with others so as to minimize the user execution cost and to maximize the quality of execution. This novel strategy of rewarding worker devices with an additional incentive for offering high quality execution would help our system to attract more workers to participate and makes such MDC system sustainable.

III. SYSTEM MODEL AND ASSUMPTIONS

This section introduces a novel computation framework for an MDC system, interactions among its functional modules and assumptions made for modeling the task execution. In the remainder of the paper, without loss of generality, the terms *buyer* and *seller* devices should be considered synonymous to the terms user and worker devices, respectively.

A. COMPUTATION FRAMEWORK

We consider an MDC system with three different entities: users *(buyers)*, workers *(sellers)* and a cloudlet *(broker)*, which are working in two tiers. Mobile devices that are running applications that require additional resources (CPU, memory, etc.) for code execution, act as *buyers* and devices providing the required computation resources act as *sellers*; both reside at tier one. Interested seller devices bid for different application tasks based on their shareable resources. As presented in Fig. 2, the allocation and distribution of these tasks and resources are done by a cloudlet, which acts as the central controller for task execution and resides in tier two. The cloudlet also acts as a broker to select the winners from a pool of candidate workers and their payment disbursement. Detailed descriptions of four functional modules of our proposed computation framework are appended below.

B. TASK PROFILER (TP)

The **TP** module receives an application with an allotted budget ($\mathcal{P}$) and an execution deadline through the *task receiver* component. It then uses the *workload analyzer* component to split the application into a set of atomic tasks $\mathcal{M}$, where each atomic task $m \in \mathcal{M}$ contains $\mathcal{S}_m$ number of instructions for offloaded execution. The *dependency estimation* component is responsible for determining the interdependency among tasks [37] which is used to identify execution order, facilitating parallel execution. The **TP** then hands the tasks over to the *task advertisement* component for advertising the tasks to

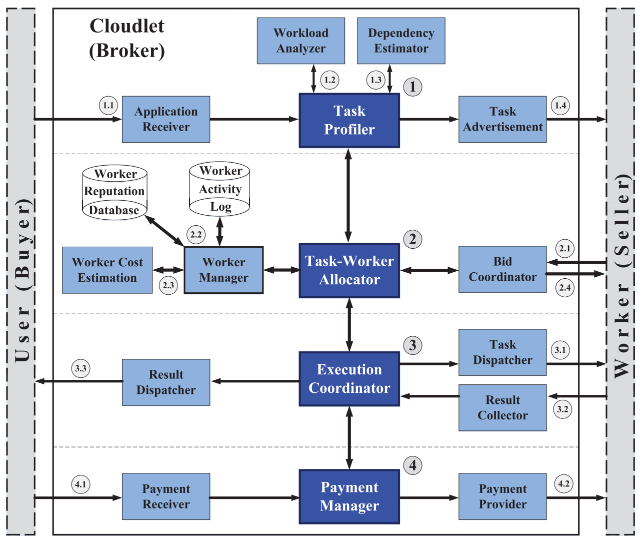


FIGURE 2. **Computation framework of the proposed incentive-driven MDC system.**

the worker devices. Each advertised task is a three-parameters tuple denoted by $< m, \mathcal{S}_m, \mathcal{T}_m >$, where, $m \in \mathcal{M}$ is the task ID, $\mathcal{S}_m$ is its number of instructions and $\mathcal{T}_m$ defines the task deadline. Based on the task size and historical transaction prices, it also calculates maximum and minimum payable amounts (B_m^{max} and B_m^{min}, respectively) for each task to guard against bids from felonious workers [38]. The **TP** shares the lists of advertised tasks information with the The **TWA**.

C. TASK-WORKER ALLOCATOR (TWA)

The **TWA** module is the core of the proposed system, and it controls and coordinates functionalities of other modules. It triggers the **TP** to collect user applications and submits advertisements after profiling the application tasks. It collects worker bids ($\mathcal{B}$) through the *bids coordinator* component. The worker devices send their device-specific information, identified by the tuple new $< k, \mathcal{B}^k, \mathcal{R}^k, \eta^k, \mathcal{H}^k >$, which contains device identifier (k), workload capacity ($\mathcal{H}^k$), associativity time (η^k), and a set of bids ($\mathcal{B}^k$). Parameter $\mathcal{R}^k$ is used to determine the aggregated resources required to execute a single instruction, whereas the workload capacity $\mathcal{H}^k$ indicates the maximum number of instructions that can be handled by the worker [20], [33]. The associativity time η^k indicates the expected amount of time the worker device will stay in the vicinity. Associativity time of a worker device can be calculated based on its contextual information and current GPS location which has been exploited in [39]. A single worker $k \in \mathcal{K}$ is allowed to bid for multiple tasks $m \in \mathcal{M}$, and each bid $\mathcal{B}_m^k \in \mathcal{B}$ is represented by tuple $< k, m, b_m^k, \mathcal{Q}_m^k >$, which contains execution cost (b_m^k) and execution quality ($\mathcal{Q}_m^k$) promised by the worker $k \in \mathcal{K}$. However, to win a bid a worker device must have to satisfy the reputation and computation resource constraint in addition to bid cost. Moreover, a worker device will be entitled to execute multiple tasks only if it has sufficient computational resources and the tasks are sequential in order of execution.

All the collected bids and device information are then transferred to the **TWA** module to determine the winners among the submitted bids. After collecting all the bids from different workers, the **TWA** matches the advertised list of tasks with the corresponding worker bids to ensure that no task remains unbidden. Notifications of winning bids ($\mathcal{W} \subseteq \mathcal{B}$) are also dispatched through the same interface to selected worker devices ($\mathcal{V} \subseteq \mathcal{K}$). The **TWA** module interacts with the *worker manager* component to determine the dependability of workers to execute tasks successfully and to estimate the required cost. The *worker manager* component evaluates submitted bids to determine the quality and reliability of worker device through historical traces containing execution history and reputation information [40], [41]. After accumulating all information, it runs a worker selection algorithm to determine the winners from the set of candidate worker devices, and then it forwards the task-worker mapping list to the *execution coordinator (EC)* to schedule the tasks in order.

D. EXECUTION COORDINATOR (EC)

On reception of the task-worker mapping list from the **TWA**, the **EC** calls the *task dispatcher* component to schedule the tasks in order. This is to be noted that due to different computation and communication resources, the worker devices exhibits heterogeneous execution and communication latencies. However, ordered scheduling of the tasks helps to diminish any synchronization latencies due to such issues. After successful execution, results are collected and transmitted back to the user device with a payment ($\mathcal{P}' \leq \mathcal{P}$) disbursement request. Failed executions are reallocated by the **TWA** to the next available bidder. After the execution of all the submitted tasks, reputation of the allocated worker devices are updated according to their execution results and sent to the **TWA** module to store in the *worker manager* database.

E. PAYMENT MANAGER (PM)

The **PM** receives the agreed amount of payment from the user device through the *payment receiver* component. Upon getting the successful execution notification from the **EC**, the **PM** disburses the individual amount of payment ($\mathcal{P}^v$) to the corresponding winning worker devices $v \in \mathcal{V}$ according to their bids along with incentives, if any, with the help of the *payment provider* component. It also collects a certain percentage of the worker bid cost as the utility ($\mathcal{U}^0$) of the cloudlet, which is acting as a broker, coordinating all these transactions and communication activities on behalf of the user and worker devices.

The proposed incentive mechanisms trade among the *seller* devices to execute application tasks of a *buyer* with minimum cost and maximum quality. The incentive mechanisms should satisfy the following desirable properties:

- **Computational efficiency**: An incentive mechanism is said to be computationally efficient if it can produce an auction decision in polynomial time.

- **Individual rationality**: The incentive mechanism must ensure a positive utility to bid-winning worker devices and the cloudlet to facilitate execution of application tasks, i.e., $\mathcal{U}_m^k > 0,\ \forall k \in \mathcal{K},\ \forall m \in \mathcal{M};\ \mathcal{U}_m^0 > 0,\ \forall m \in \mathcal{M}$, where $\mathcal{U}_m^k$ denotes the utility of worker $k \in \mathcal{K}$ for task $m \in \mathcal{M}$. This basic requirement is mandatory to encourage the participation of worker devices in the system.
- **Truthfulness**: An incentive mechanism is truthful if it can guarantee that only the bidders declaring true costs are eligible to win the auction. No bidder can increase its utility by submitting a bid other than its actual cost.
- **Budget balance**: The incentive mechanism must guarantee that the total amount of payment $\mathcal{P}'$ charged by different worker devices is within the budget allocation $\mathcal{P}$ of a user for a certain application, i.e., $\mathcal{P}' \leq \mathcal{P}$.

F. ASSUMPTIONS

Based on the state-of-the-art works, we have made the following assumptions in this work.

We assume a rooted tree of application tasks as the vertices, and the linkages correspond to dependencies among the tasks. Execution of the tasks begins from the root, where parallel tasks start their execution simultaneously and dependent tasks start execution after completion of the parent task [4], [37], [42]. Hence, the execution performance of an application mostly relies on the number of dependent and parallel tasks. The overall execution delay is calculated considering both execution times and communication latencies involved.

We assume there will be a sufficient number of worker devices in the system to allocate all the application tasks, and each task will be bid on by at least one worker. The worker devices agree to execute tasks assigned to them, and each device will execute one task at a time [18], [20]. However, it may execute multiple tasks of one application. These worker devices are symmetric, independent, and risk neutral, having no security or privacy violations. A worker device may bid for multiple advertised tasks, where each bid has been generated randomly considering given task size and deadline. When a worker device submits quality information in a bid, it includes communication latency with actual execution time [43].

We assume the system will be running on a trusted platform where all executions will be in a secure environment and all the device-cloudlet interactions will be governed by proper authorization and authentication techniques [44]. We consider a quasi-static behavior for mobility of the user and worker devices to determine the associativity time (η), where the movement of the devices will remain relatively unchanged for a given period of time that is sufficient to execute the allocated task and return the result to the cloudlet [8], [26], [39]. In this work, current GPS location and contextual information have been used to determine the associativity time of a worker mobile device [39].

The major notations used in this paper are listed in Table 1.

TABLE 1. List of notations.

Symbol	Description
$\mathcal{M}$	Set of advertised tasks
$\mathcal{K}$	Set of candidate worker devices
$\mathcal{V}$	Set of winning worker devices
$\mathcal{B}$	Set of bids submitted by worker devices
$\mathcal{W}$	Set of winning bids
b_m^k	Bid of worker $k \in \mathcal{K}$ for executing task $m \in \mathcal{M}$
η^k	Associativity time of worker $k \in \mathcal{K}$
$\mathcal{H}^k$	Maximum resource capacity of worker $k \in \mathcal{K}$
$\mathcal{S}_m$	Number of instructions in task $m \in \mathcal{M}$
$\mathcal{T}_m$	Execution deadline of task $m \in \mathcal{M}$
$\mathcal{T}_m^k$	Execution time of task $m \in \mathcal{M}$ in worker $k \in \mathcal{K}$
$\mathcal{L}_m^k$	Communication latency of task $m \in \mathcal{M}$ in worker $k \in \mathcal{K}$
$\mathcal{R}_m$	Resource requirement for executing task $m \in \mathcal{M}$
$\mathcal{C}_m^k$	Claimed cost of worker $k \in \mathcal{K}$ for task $m \in \mathcal{M}$
$\mathcal{P}$ / $\mathcal{P}'$	Allocated / Final payment
$\mathcal{P}_m^k$	Payment of worker $k \in \mathcal{K}$ for task $m \in \mathcal{M}$
λ	Cloudlet Utility percentage for coordination
$\mathcal{U}_m^0$	Utility of cloudlet for serving task $m \in \mathcal{M}$
$\mathcal{U}_m^k$	Utility of worker $k \in \mathcal{K}$ for task $m \in \mathcal{M}$
σ_m^k	Incentive amount of worker $k \in \mathcal{K}$ for task $m \in \mathcal{M}$
Ω^k / Ω^v	Previous / Final reputation of worker $k \in \mathcal{K}$, $v \in \mathcal{K}$
Ω_m^k	Earned reputation of worker $k \in \mathcal{K}$, $m \in \mathcal{M}$
$\mathcal{Q}_m^k$ / $\mathcal{Q}_m^{k'}$	Offered / Provided quality of worker $k \in \mathcal{K}$, $m \in \mathcal{M}$

IV. PROPOSED INCENTIVE MECHANISM

Successful execution of application tasks greatly depends on the selection of high performing and reputed worker devices. The execution time of an application task also significantly varies from one worker to another due to their resource heterogeneities, which offers varied QoE for users. This section first details the design of an optimal selection process of worker devices considering user QoE and task execution cost. Due to the NP-hardness of the optimal solution, we then develop greedy algorithms for task assignments on suitable workers so as either to maximize QoE or minimize execution cost. Finally, this section ends by presenting a QoE-aware incentive payment mechanism and theoretical proofs of its properties.

A. OPTIMAL SELECTION OF WORKER DEVICES

The competency of our proposed QoE-aware incentive mechanism mostly relies on efficient selection of worker devices so that the overall execution quality is increased and cost is decreased. The efficiency of application task allocations by the **TWA** module on different worker devices also depends on their reputations and resource availabilities. In addition to that, interdependency among the tasks and costs demanded by the worker devices for task execution are also important. Thus, provisioning of application tasks on worker devices is a multi-objective, multi-constrained problem. The next subsections describe methods for measuring user QoE and execution cost metrics, followed by formulation of an optimization framework for the problem.

1) USER QUALITY-OF-EXPERIENCE (QoE)

As discussed earlier, each task has an associated deadline $\mathcal{T}_m$ determined by the cloudlet [45], within which the output of a task must be available to the cloudlet. Execution quality of an application, as termed as service level agreement (SLA) quality, is defined as the ratio of task execution time on the worker device to that on user device. Therefore, quality for a single task $m \in \mathcal{M}$ being offloaded to device $k \in \mathcal{K}$ can be calculated as

$$\mathcal{Q}_m^k = 1 - (\frac{\mathcal{T}_m^k + \mathcal{L}_m^k}{\mathcal{T}_m}), \quad \mathcal{Q}_m^k \in (0, 1], \tag{1}$$

where $\mathcal{T}_m^k$ denotes task execution time, and $\mathcal{L}_m^k$ indicates communication latency between worker device $k \in \mathcal{K}$ and the cloudlet during input and output transmission for task $m \in \mathcal{M}$. The task execution time $\mathcal{T}_m^k$ is calculated by $\frac{\mathcal{S}_m}{\mu^k}$, where μ^k represents the CPU speed of a worker device $k \in \mathcal{K}$. The ratio $\frac{\mathcal{T}_m^k+\mathcal{L}_m^k}{\mathcal{T}_m} < 1$ because execution delay cannot exceed the deadline for any task. Now, combining the quality of all tasks, the QoE observed by the user for an application can be expressed as

$$\mathcal{Q} = \frac{1}{|\mathcal{M}|} \sum_{m=1}^{|\mathcal{M}|} \mathcal{Q}_m^k, \quad \mathcal{Q} \in (0, 1]. \tag{2}$$

The target of our QoE-aware incentive mechanism is to increase the value of $\mathcal{Q}$ for a user application.

2) COST OF EXECUTION

A worker device participating in task execution incurs a cost due to usage of a certain amount of resources (CPU, memory, bandwidth, etc.). Therefore, a payment for the used resources is necessary to incentivize the worker device, promoting this service model [20]. The cost to execute each task $m \in \mathcal{M}$ is determined by the amount of resources utilized by the task during task execution time. The amount of computation resources (CPU clock speed) required to execute task $m \in \mathcal{M}$ with task size $\mathcal{S}_m$ can be given as

$$\mathcal{R}_m = \mathcal{S}_m \times \mathcal{R}^k, \tag{3}$$

where $\mathcal{R}^k$ is the resources required by candidate device $k \in \mathcal{K}$ to execute a single instruction. Then, the total cost is calculated by considering cloudlet utility along with execution cost. Now, considering $\mathcal{C}^k$ to be the cost of utilizing a unit resource in device $k \in \mathcal{K}$, the cost of executing task $m \in \mathcal{M}$ can be determined by

$$\mathcal{C}_m^k = \mathcal{R}_m \times \mathcal{C}^k. \tag{4}$$

After successful completion of a task, the corresponding worker devices earn payments with incentives in line with their execution qualities. From application task profiling and worker selection to task dissemination, the cloudlet controls and coordinates the whole process of execution in the MDC system. Thus, our worker devices pay a certain proportion of their bid amounts to the cloudlet as a coordinating utility. The utility of the cloudlet for executing task $m \in \mathcal{M}$ by candidate device $k \in \mathcal{K}$ is

$$\mathcal{U}_m^0 = b_m^k \times \lambda, \tag{5}$$

where b_m^k is the price of bidding by worker device $k \in \mathcal{K}$ to execute task $m \in \mathcal{M}$ that contains a marginal profit with accumulated costs for used resources and cloudlet payment, and λ is the utility percentage that will be given to the cloudlet for coordinating this task assignment and execution. The value of this utility percentage is a system design parameter, and it may vary from one cloudlet to another over a given period. The total utility of the cloudlet $\mathcal{U}^0$ for providing necessary support to execute all tasks of an application is scaled by

$$\mathcal{U}^0 = \sum_{m=1}^{|\mathcal{M}|} \mathcal{U}_m^0. \quad \forall k \in \mathcal{K} \tag{6}$$

Therefore, the utility of candidate $k \in \mathcal{K}$ for executing a single task $m \in \mathcal{M}$ is

$$\mathcal{U}_m^k = b_m^k - \mathcal{C}_m^k - \mathcal{U}_m^0, \tag{7}$$

where $\mathcal{C}_m^k$ is the actual cost of executing task $m \in \mathcal{M}$ on device $k \in \mathcal{K}$. Accumulating the cost of each task $m \in \mathcal{M}$ in the application, we can get total bidding cost of application execution on the candidate devices, quantified as

$$\mathcal{C}_\mathcal{M} = \sum_{m=1}^{|\mathcal{M}|} \sum_{k=1}^{|\mathcal{K}|} b_m^k \times y_m^k, \tag{8}$$

where binary variable $y_m^k \in \{0, 1\}$ takes value 1 if task $m \in \mathcal{M}$ is executed on worker device $k \in \mathcal{K}$ and 0 otherwise. Hence, the normalized bidding cost $\mathcal{C}$ of the user is gained by

$$\mathcal{C} = \frac{\mathcal{C}_\mathcal{M}}{\mathcal{P}}, \quad \mathcal{C} \in [0, 1], \tag{9}$$

where $\mathcal{P}$ is user-sanctioned budget for execution of the complete application and is the summation of the maximum allowable bid costs for each task $m \in \mathcal{M}$, i.e., $\mathcal{P} = \sum_{m=1}^{|\mathcal{M}|} B_m^{max}$. While selecting worker devices, the proposed incentive mechanism aims to minimize this normalized execution cost for an application.

3) OPTIMAL OBJECTIVE FUNCTION

The selection of an optimal set of candidate devices for offloading tasks of an application can now be formulated as

$$Maximize : \mathcal{Z} = \underset{\mathcal{W} \in P(\mathcal{B})}{\text{argmax}} \sum_{\forall w \in \mathcal{W}} \{\beta \times \mathcal{Q} - (1-\beta) \times \mathcal{C}\}, \tag{10}$$

subject to:

$$\mathcal{C}_\mathcal{M} \leq \mathcal{P} \tag{11}$$
$$\eta^k \geq max(\mathcal{T}_n^k) + \mathcal{T}_m^k(1 - \Phi_{n,m}), \quad \forall n \in \Pi_m \tag{12}$$
$$\Omega^k \geq \gamma, \quad \forall k \in \mathcal{K} \tag{13}$$
$$\sum_{m=1}^{|\mathcal{M}|} \mathcal{S}_m \times y_m^k \leq \mathcal{H}^k, \quad \forall k \in \mathcal{K}, \quad \forall m \in \mathcal{M} \tag{14}$$
$$\mathcal{U}_m^k > 0, \quad \forall k \in \mathcal{K}, \forall m \in \mathcal{M}; \quad \mathcal{U}^0 > 0, \quad \forall m \in \mathcal{M} \tag{15}$$
$$\sum_{k \in \mathcal{K}} y_m^k = 1, \quad \forall m \in \mathcal{M}. \tag{16}$$

In the above formulation, the aim of the objective function (10)[1] is to excel in task QoE while reducing execution cost. It chooses a bid set $\mathcal{W}$ from the power set of bids $P(\mathcal{B})$, which optimizes the said parameters. Here, β is the relative weight parameter, which works as a control knob. Its value can be tuned to obtain different tradeoff levels between application QoE and execution cost required by various types of applications. Setting $\beta = 1$ translates it into a quality maximization problem, and $\beta = 0$ makes it a cost minimization problem, while other values correspond to various quantified levels of tradeoff between the two. The cloudlet determines an appropriate value of β following user demands.

The budget constraint defined in (11) means that the total payment of workers must not exceed the user-sanctioned payment. The availability constraint (12) specifies the minimum amount of time a selected candidate device must stay in the system. Here, Π_m is the set of parents of a task $m \in \mathcal{M}$ and $\Phi_{n,m}$ is the percentage of dependency of a child task $m \in \mathcal{M}$ on its parent $n \in \Pi_m$ [37]. The reputation constraint (13) ensures that to win an auction for any task $m \in \mathcal{M}$, the reputations of each selected candidate must be greater than a certain minimum threshold. The taskload constraint (14) refers to the fact that a candidate device's total assigned taskload must be less than or equal to its specified maximum capacity $\mathcal{H}^k$. The utility constraint (15) ensures that each device executing a task of an application will earn positive revenue. The cloudlet will also earn positive utility for supporting the execution service for each individual task. Similarly, the atomicity constraint (16) confirms that a single task will not be assigned to multiple candidate devices, and each task should be executed only once.

Theorem 1: The proposed worker device selection problem in (10) is NP-hard.

Proof: The optimization framework in (10) is a MOLP because since it contains two conflicting objectives (i.e., maximizing quality and minimizing cost) with combinatorial and continuous constraints. The worker selection problem can be reduced to a multiprocessor scheduling problem (an NP-complete scheduling problem) [46] by leveraging the constraints and considering that all workers offer equal quality. Hence, the proposed MOLP problem is NP-hard and cannot provide a polynomial time solution.

In a practical MDC platform, a typical application containing approximately $10 - 15$ individual tasks may generate thousands of bids. To find boundary values of worker and tasks in a typical MDC environment, we simulate the objective function in NEOS optimization server (2× Intel Xeon e5-2698 @ 2.3GHZ CPU and 92GB RAM) with $\beta = 0.5$. The graphs in Fig. 3 show that the computation time for a higher number of tasks and workers exponentially increases due to exploring an enormous number of task-worker assignments. For 20 tasks and 30 workers,

[1]Please note that (10) is not a typical multi-objective optimization problem; rather, it brings a tradeoff between the two conflicting objective parameters - quality and cost.

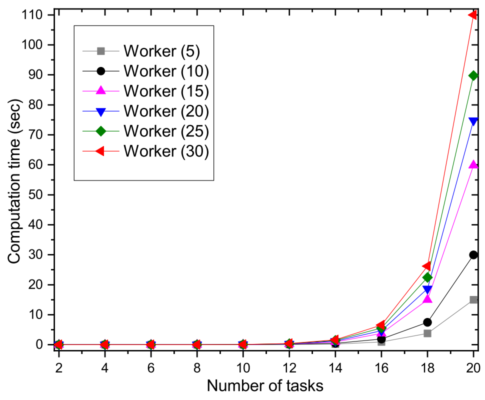


FIGURE 3. Computation time for optimal selection of workers.

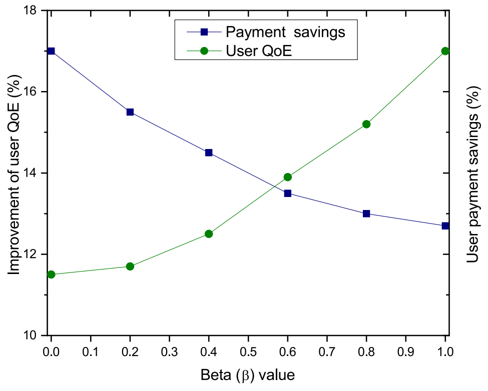


FIGURE 4. Impact of weight parameter (β).

the run time exceeds 100 seconds, which might not be tolerable for a decision-making algorithm. In a separate experiment, the impact of weight parameter (β) was studied for 12 tasks and 20 worker devices as shown in Fig. 4. The graphs show that with the increasing β value, the user QoE increases and the user payment saving amount decreases. Our further studies on QoE and payment savings have been discussed in Section V.

Since, the above formulation turns to be an NP-hard problem, we develop two greedy solutions for assigning tasks to workers to overcome the problem.

B. GREEDY SELECTION OF WORKER DEVICES

To support real-time processing of user applications, this section introduces light-weight greedy worker selection algorithms focusing on either maximizing execution quality or minimizing execution cost. Algorithm 1 selects workers that maximize task execution quality while maintaining total cost within the allocated budget. On the other hand, Algorithm 2 selects workers that demand minimum cost while maintaining the required quality of execution. Detailed descriptions of the algorithms are given in the following subsections.

1) MAXIMIZING TASK EXECUTION QUALITY

Algorithm 1 takes a set of bids $\mathcal{B}$ and a set of tasks $\mathcal{M}$ as input and produces a set of winning bids $\mathcal{W}$ as output. It uses a priority queue of bids $\mathcal{B}'$ in descending order of

Algorithm 1 Worker Selection for Maximizing Quality

INPUT: *Set of bids from all worker devices for all tasks,* $\mathcal{B} \leftarrow \bigcup_{\forall k \in \mathcal{K}, \forall m \in \mathcal{M}} \mathcal{B}_m^k$

OUTPUT: *Set of winning bids,* $\mathcal{W}$

```
1.  Set $\mathcal{W} \leftarrow \phi$, $\mathcal{M}' \leftarrow \mathcal{M}$
2.  Set $\mathcal{R}_u^k \leftarrow 0 \ \forall k \in \mathcal{K}$
3.  Set Priority Queue, $\mathcal{B}' \leftarrow \phi$
4.  for all Bids $\mathcal{B}_m^k \in \mathcal{B}$ do
5.      if $((b_m^k > B_m^{min}$ && $b_m^k < B_m^{max})$ && $(\Omega^k \geq \gamma)$ && $(\mathcal{Q}_m^k > 0))$ then
6.          $\mathcal{B}'.push(\mathcal{B}_m^k)$ {Insert item following priority on higher values of $\mathcal{Q}_m^k$}
7.      end if
8.  end for
9.  while $(\mathcal{B}' \neq \phi$ && $\mathcal{M}' \neq \phi)$ do
10.     $\mathcal{F} \leftarrow \mathcal{B}'.pop()$
11.     if $(\mathcal{R}_m + \mathcal{R}_u^k < \mathcal{H}^k$ && $\mathcal{F}.k \cap \mathcal{M}' \neq \phi)$ then
12.         $\mathcal{W} \leftarrow \mathcal{W} \cup \mathcal{F}$
13.         $\mathcal{R}_u^k \leftarrow \mathcal{R}_m + \mathcal{R}_u^k$
14.         $\mathcal{M}' \leftarrow \mathcal{M}' \setminus k \mid k \in \mathcal{F}$
15.     end if
16.     $\mathcal{B}' \leftarrow \mathcal{B}' \setminus \mathcal{F}$
17. end while
18. return $\mathcal{W}$
```

their execution qualities (line 3). First, the algorithm prepares a candidate bid set after pruning the bids that cannot meet the minimum reputation and fall outside of the bid boundary. These bids are sorted with higher quality values and stored in the priority queue (lines $4-8$). Then, the task of the topmost bid is assigned to the corresponding worker device that fulfills the required resource requirements (lines $10-11$). After successful task assignment, the corresponding task and the bid is removed from the queue and the procedure is repeated until all the tasks assignment are completed (lines $9-17$).

2) MINIMIZING TASK EXECUTION COST

The algorithm selects worker devices that can satisfy execution constraints, ascertain minimum execution quality, and reduce overall execution costs. Delay tolerant applications, such as text translation, audio video transmission, online forum, and blogging can compromise in terms of execution quality, providing us the opportunity to minimize execution cost.

Algorithm 2 follows similar steps of Algorithm 1, except the priority queue $\mathcal{B}'$ is based on minimum execution cost (line 6). The algorithm selects all bids that provide minimum task execution cost and ensures at least the minimum quality (lines $4-17$).

What we unfold next is the process of providing additional incentives to the high performing workers following their offered qualities of task executions. Note here that the incentive mechanisms are applicable for workers assigned tasks either by MOLP system or greedy algorithms.

Algorithm 2 Worker Selection for Minimizing Cost

INPUT: *Set of bids from all worker devices for all tasks,* $\mathcal{B} \leftarrow \bigcup_{\forall k \in \mathcal{K}, \forall m \in \mathcal{M}} \mathcal{B}_m^k$

OUTPUT: *Set of winning bids,* $\mathcal{W}$

```
1.  Set $\mathcal{W} \leftarrow \phi$, $\mathcal{M}' \leftarrow \mathcal{M}$
2.  Set $\mathcal{R}_u^k \leftarrow 0 \ \forall k \in \mathcal{K}$
3.  Set Priority Queue, $\mathcal{B}' \leftarrow \phi$
4.  for all Bids $\mathcal{B}_m^k \in \mathcal{B}$ do
5.      if $((b_m^k > B_m^{min}$ && $b_m^k < B_m^{max})$ && $(\Omega^k \geq \gamma)$ && $(\mathcal{Q}_m^k > 0))$ then
6.          $\mathcal{B}'.push(\mathcal{B}_m^k)$ {Insert item following priority on lower values of $b_m^k$}
7.      end if
8.  end for
9.  while $(\mathcal{B}' \neq \phi$ && $\mathcal{M}' \neq \phi)$ do
10.     $\mathcal{F} \leftarrow \mathcal{B}'.pop()$
11.     if $(\mathcal{R}_m + \mathcal{R}_u^k < \mathcal{H}^k$ && $\mathcal{F}.k \cap \mathcal{M}' \neq \phi)$ then
12.         $\mathcal{W} \leftarrow \mathcal{W} \cup \mathcal{F}$
13.         $\mathcal{R}_u^k \leftarrow \mathcal{R}_m + \mathcal{R}_u^k$
14.         $\mathcal{M}' \leftarrow \mathcal{M}' \setminus k \mid k \in \mathcal{F}$
15.     end if
16.     $\mathcal{B}' \leftarrow \mathcal{B}' \setminus \mathcal{F}$
17. end while
18. return $\mathcal{W}$
```

C. QUALITY-OF-EXPERIENCE-AWARE INCENTIVE MECHANISM

Upon successful execution of all tasks of an application, payment is calculated for disbursement to the corresponding bid winners $v \in \mathcal{V}$. A worker device is paid the bid amount (b_m^v) and an additional incentive based on the executed task quality $(\mathcal{Q'}_m^v)$, which is measured after task execution. This incentive might play a vital role in motivating worker devices to submit rational bids and increasing participation of valued workers in the bidding process.

The entitlement of an incentive depends on two parameters. Firstly, the bid cost must be less than the maximum budget for the task (i.e., $b_m^v < B_m^{max}$). Secondly, the provided execution quality of a task must be higher than the committed quality (i.e., $\mathcal{Q'}_m^v > \mathcal{Q}_m^v$). If these two criteria are fulfilled then we calculate the amount of bonus quality provided by the worker device, i.e., $(\frac{\mathcal{Q'}_m^v - \mathcal{Q}_m^v}{\mathcal{Q}_m^v})$. A worker device is given a portion of the unused budget $(B_m^{max} - b_m^v)$ as an incentive for the provided bonus quality. Otherwise, the worker will not be entitled to any incentive amount. Thus, we calculate the incentive amount σ_m^v as

$$\sigma_m^v = \begin{cases} (B_m^{max} - b_m^v) \times \dfrac{\mathcal{Q'}_m^v - \mathcal{Q}_m^v}{\mathcal{Q}_m^v}, & \text{if } b_m^v < B_m^{max} \\ & \&\& \ \mathcal{Q'}_m^v > \mathcal{Q}_m^v. \\ 0, & \text{otherwise.} \end{cases} \quad (17)$$

Now, the cloudlet is responsible for computing the required payment to the worker devices. A detailed description of the

Algorithm 3 Incentive Payment for Winner Devices

INPUT: *Set of winning bids*, $\mathcal{W}$, from Algo 1 or Algo 2.
OUTPUT: *Payment* $\mathcal{P}^v$ *for winner device,* $v \in \mathcal{V}$,
Total amount of payment, $\mathcal{P}'$ *for the user application.*

```
1.  Set $\mathcal{P}^v \leftarrow 0, \mathcal{P}' \leftarrow 0$
2.  $\mathcal{V} = \{v \mid v$ is a winning device listed in $\mathcal{W}\}$
3.  for all winning devices $v \in \mathcal{V}$ do
4.      for all tasks $m \in \mathcal{M}$ do
5.          Calculate incentive amount, $\sigma_m^v$ using Eqn. (17)
6.          $\mathcal{P}_m^v \leftarrow b_m^v + \sigma_m^v$
7.          $\mathcal{P}^v \leftarrow \mathcal{P}^v + \mathcal{P}_m^v$
8.          $\mathcal{P}' \leftarrow \mathcal{P}' + \mathcal{P}_m^v$
9.      end for
10. end for
11. return $\mathcal{P}^v, \mathcal{P}'$
```

payment strategy for bid-winning worker devices is presented in Algorithm 3.

After completion of all tasks, the cloudlet picks a worker device and calculates incentives of each task executed by it using (eq. 17), total payment for the task ($\mathcal{P}_m^v$), and payment for all the executed tasks ($\mathcal{P}^v$). These steps are repeated for all the worker devices (lines $3-7$). Finally, the cloudlet calculates the total actual payment ($\mathcal{P}'$) of the user and disburses payments to individual workers upon reception of actual payment from the user (line 8). We call the incentive payment mechanism *IMaxQ* when the workers are selected to maximize task execution quality (in Algorithm 1) and *IMinC* when the workers are selected to minimize task execution cost (in Algorithm 2).

D. UPDATING REPUTATIONS OF WORKER DEVICES

Reputations of worker devices are updated by the ***TWA*** module based on the execution results of the tasks; this is important for selecting winning devices in future task assignments. To encourage truthful bidding, successful execution with the offered quality provides a positive reputation, whereas failure in maintaining the offered quality or deadline results in a penalty to protect against dishonest activity of unqualified workers.

To calculate the reputation of a winning device $v \in \mathcal{V}$, the cloudlet first calculates the quality enhancement indicator $\mathcal{E}_m^v$ of the executed task based on qualities $\mathcal{Q}_m^v$ and $\mathcal{Q'}_m^v$ offered in SLA and provided by the worker, respectively.

$$\mathcal{E}_m^v = \begin{cases} 1, & \text{if } \mathcal{Q'}_m^v \geq \mathcal{Q}_m^v \\ (1 - \dfrac{\mathcal{Q}_m^v - \mathcal{Q'}_m^v}{\mathcal{Q}_m^v}), & \text{if } 0 < \mathcal{Q'}_m^v < \mathcal{Q}_m^v \\ -1, & \text{unsuccessful.} \end{cases} \tag{18}$$

The task quality enhancement indicator ($\mathcal{E}_m^v$) is used to calculate the reputation/penalty value gained for executing the current task $m \in \mathcal{M}$ as

$$\Omega_m^v = \alpha \times \mathcal{E}_m^v \times (-1)^{x_m^v} \times (-1)^{y_m^v}, \tag{19}$$

where α is a weight parameter used to put emphasis on the currently availed reputation/penalty. In this work, the value of α was set to 0.1, placing only a small significance on the reputation achieved for executing the current task. The binary variable $x_m^v \in \{1, 0\}$ is used to represent service level agreement (SLA) retention status. x_m^v is set to 1 if worker device $v \in \mathcal{V}$ successfully executes task $m \in \mathcal{M}$, maintaining $\mathcal{Q'}_m^v \geq \mathcal{Q}_m^v$; otherwise, it is set to 0. Similarly, binary variable $y_m^v \in \{1, 0\}$ takes value 1 if task $m \in \mathcal{M}$ successfully executed on the worker device $v \in \mathcal{V}$ and 0 otherwise. Finally, the reputation of a worker device is updated for future task assignments considering previous reputation as

$$\Omega^v = max(0, min(\Omega'^v + \Omega_m^v, 1)). \tag{20}$$

Equation (20) bounds the reputation for a worker device $v \in \mathcal{V}$ between 0 and 1. In this work, the initial reputation of a worker device was considered as 1.0 to encourage new workers to participate in the MDC system. Note that (19) and (20) jointly ensure that on-time successful execution increases the reputation of a worker and that delayed or failed execution causes a penalty for the device. In this way, the dynamic reputation update of a worker facilitates our proposed incentive mechanisms to select high performing workers for task execution, increasing user QoE as well as incentives for workers.

E. AN ILLUSTRATIVE EXAMPLE

This section presents an illustrative example on operation processes of the proposed *IMaxQ*, *IMinC* and *Optimal* solutions. Consider a scenario depicted in Fig. 5, where the cloudlet divides an application into three tasks: $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$ with allowable maximum execution costs 3, 5, and 7 units, respectively. We also assumed that five worker devices $\mathcal{K}_1 - \mathcal{K}_5$ available in the system bid for the designated tasks shown by edges (in Fig. 5, labeled with bid cost and committed quality). The instruction sizes and deadlines of the tasks are labeled at the top and available resource capacities and reputations of the worker devices are labeled at the bottom.

The task assignment to different workers, their incentives, and user cost savings have been shown in following Table 2. It is to be noted here that the worker $\mathcal{K}_2$ could not win any bid due to its poor reputation. Furthermore, as expected theoretically, the proposed *IMinC* has brought out the maximum cost savings for the user and *IMaxQ* has offered the lowest while the *Optimal* solution (with $\beta = 0.5$) has worked out with competitive savings. We also notice that, in all algorithms, tasks were executed satisfying the expected QoE within the allocated budget. Finally, the amount of incentive (σ_m^v) awarded to a worker is directly proportional to the quality ($\mathcal{Q'}_m^v$) it offers in executing a user task.

F. THEORETICAL PROOF OF DESIRABLE PROPERTIES

In this section, we provide theoretical proofs of the desirable properties of the proposed incentive mechanisms, including

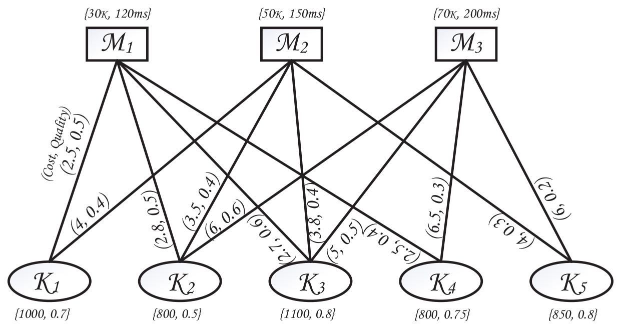


FIGURE 5. An example scenario for task assignment.

TABLE 2. Bid winners, their incentives, and user cost savings.

	$\mathcal{K}$	$\mathcal{M}$	$\mathcal{B}_m^{max}$	b_m^v	$\mathcal{Q}_m^v$	$\mathcal{Q'}_m^v$	σ_m^v	$Savings(\%)$
IMaxQ	3	$\mathcal{M}_1$	3.0	2.7	0.6	0.7	0.05	
	1	$\mathcal{M}_2$	5.0	4.0	0.4	0.5	0.25	10.0%
	4	$\mathcal{M}_3$	7.0	6.5	0.3	0.3	0	
IMinC	1	$\mathcal{M}_1$	3.0	2.5	0.5	0.6	0.1	
	3	$\mathcal{M}_2$	5.0	3.8	0.4	0.5	0.3	12.0%
	5	$\mathcal{M}_3$	7.0	6.0	0.2	0.3	0.5	
Optimal	4	$\mathcal{M}_1$	3.0	2.5	0.4	0.4	0	
	1	$\mathcal{M}_2$	5.0	4.0	0.4	0.5	0.25	11.67%
	5	$\mathcal{M}_3$	7.0	6.0	0.2	0.3	0.5	

computational efficiency, individual rationality, truthfulness, and budget balance.

Lemma 1: The proposed incentive mechanisms run in polynomial time, meaning they are computationally efficient.

Proof: In Algorithm 1, lines 4–8 run for all bids to check the eligibility of bids and to prune bids that do not maintain the required constraints, incurring a runtime complexity of $O(|\mathcal{B}|)$. Inside the loop, candidate bids are pushed in a priority queue (line 6) that has a runtime complexity of $O(|\mathcal{B}|log|\mathcal{B}|)$, totaling $O(|\mathcal{B}| \times |\mathcal{B}|log|\mathcal{B}|)$. Lines $9-17$ add complexity of $O(|\mathcal{B}|)$, and the remaining statements are executed in constant time; hence, the overall runtime complexity of Algorithm 1 is $O(|\mathcal{M}| \times |\mathcal{K}| \times |\mathcal{M}| \times |\mathcal{K}|log|\mathcal{M}| \times |\mathcal{K}| + |\mathcal{M}| \times |\mathcal{K}|) \approx O(|\mathcal{M}|^2 \times |\mathcal{K}|^2\ log|\mathcal{M}| \times |\mathcal{K}|)$, where $\mathcal{M}$ is the set of application tasks and $\mathcal{K}$ is the set of available worker devices. The runtime complexity of Algorithm 2 is the same as that of Algorithm 1. Similarly, Algorithm 3 has a runtime complexity of $O(|\mathcal{M}| \times |\mathcal{K}|)$.

Thus, the complexities of the algorithms are bounded by $|\mathcal{M}|$ and $|\mathcal{K}|$, so they are computable within polynomial time.

Lemma 2: The proposed incentive mechanisms are individually rational to the cloudlet and worker devices.

Proof: Consider the two following scenarios:

1) **If a worker device** $k \in \mathcal{K}$ **wins no bid,** it will have no resulting cost, $\mathcal{C}'^k = 0$, payment $\mathcal{P}^k = 0$, and $\mathcal{U}^k = 0$. In this case, the utility of the cloudlet $\mathcal{U}^0 = 0$.
2) **If a worker device** $k \in \mathcal{K}$ **wins a bid**, its utility is calculated as $\mathcal{U}_m^k = \mathcal{P}_m^k - \mathcal{C}_m^k - \mathcal{U}_m^0$, where payment is determined as $\mathcal{P}_m^k = b_m^k + \sigma$. According to (5), cloudlet utility $(\mathcal{U}_m^0)$ will be nonzero for a successful task execution by a worker device. Moreover, for a successful execution by a worker device, payment will be at least equal to the bid price. As worker devices bid for a task $m \in \mathcal{M}$ including cloudlet payment $(\mathcal{U}_m^0)$ and its resource cost $\mathcal{C}_m^k$, bid price must be higher than $\mathcal{U}_m^0 + \mathcal{C}_m^k$, which has been ensured through constraint (15). Therefore, in conclusion, it can be clearly seen that the utility of a worker device and cloudlet will be nonzero.

Lemma 3: The proposed incentive mechanisms are truthful.

Proof: Let $\mathcal{U}_m^k$ and b_m^k denote the utility and bid price, respectively, of a worker device $k \in \mathcal{K}$ if it bids with actual cost $\mathcal{C}_m^k$, and let $\bar{\mathcal{U}}_m^k$ and $\bar{b}_m^k$ denote the illegitimate utility and bid price, respectively, caused by bad intension of a worker, where $b_m^k \neq \bar{b}_m^k$. To prove this lemma, we must show that only a truthful bid price can provide the maximum utility of a worker device, i.e., $\mathcal{U}_m^k \geq \bar{\mathcal{U}}_m^k, \forall b_m^k \neq \bar{b}_m^k$. In this case, we first consider Algorithm 2 ($IMinC$).

Assume that $\bar{b}_m^k > b_m^k$, a win of $\bar{b}_m^k$ means that the worker wins when it bids $\bar{b}_m^k$, and a loss of $\bar{b}_m^k$ means that the worker loses when it bids $\bar{b}_m^k$. Algorithm 2 confirms that a worker with minimum bid cost wins the bid. Hence, $\bar{b}_m^k$ loses the bid in the presence of another bid that equals b_m^k, and thus the utility $\bar{\mathcal{U}}_m^k = 0$ and $\mathcal{U}_m^k > \bar{\mathcal{U}}_m^k$.

If $\bar{b}_m^k = b_m^k$, then a win of $\bar{b}_m^k$ means that b_m^k also wins. In this case, $\mathcal{U}_m^k = \bar{\mathcal{U}}_m^k$ signifies that the actual bid is made by a legitimate worker.

Lastly, if $\bar{b}_m^k < b_m^k$, then $\bar{b}_m^k$ wins the bid according to the worker selection criteria. However, in this case, the expected utility constraint $\bar{\mathcal{U}}_m^k < \mathcal{U}_m^k$ is preserved.

Similarly, for Algorithm 1 ($IMaxQ$), workers offering higher quality may submit overpriced bids. In this case, maximum bid cost $(\mathcal{B}_m^{max})$ for a task $m \in \mathcal{M}$ is used to guard against such dishonest activity and reject such bids during the candidate worker set generation phase, which results in utility $\mathcal{U}_m^k = 0$ for worker $k \in \mathcal{K}$. Moreover, the incentive (σ) for qualified execution induces extra profit to deserving workers. For this reason, workers are motivated in bidding with actual cost to increase the opportunities for a winning bid.

It is to be noted here that a worker might win a task by misreporting both the quality and bid but it would be penalized with a negative reputation due to failure in achieving the committed quality.

Lemma 4: The proposed incentive mechanisms are budget balanced.

Proof: The total amount of payment for a user to execute all tasks is calculated as $\mathcal{P}' = \sum_{m=1}^{|\mathcal{M}|} \mathcal{P}_m^k$, including their incentive (lines 4–7 in Algorithm 3). If all payments of tasks are equal to their maximum allowable bid prices (including incentive σ, if any), then the total payment will be $\mathcal{P}' = \mathcal{C}_\mathcal{M} = \mathcal{P}$ (according to (8)); otherwise, the payment will be $\mathcal{C}_\mathcal{M} \leq \mathcal{P}' \leq \mathcal{P}$. This means that the total cost of an application execution will always be less than or equal to the budget allocated by the user, i.e., $\mathcal{P}' \leq \mathcal{P}$. Hence, the proposed incentive mechanisms maintain the budget balance property.

V. PERFORMANCE EVALUATION

The performances of the proposed **IMaxQ** and **IMinC** algorithms were compared with two state-of-the-art works (**Min-Cost** [20] and **First-Fit** [47]) through numerical simulations in MATLAB [22]. The ***Min-Cost*** algorithm employed a least cost per unit resource mechanism to select the winning bids. The workers that offer a minimum bid are selected for the execution of tasks and the corresponding worker devices were paid with the immediate next bid amount of the task. On the other hand, ***First-Fit*** is a baseline method extracted from [47] that greedily allocates tasks on worker devices with sufficient resources without considering execution quality, cost, or device reputation.

A. SIMULATION SETUP

We assumed that a number of user devices (buyers) issue application code execution requests to a nearby cloudlet on which there are 10 − 100 connected worker devices *(seller)*, randomly distributed in a $50 \times 50\ m^2$ area. A user application ranging the size from $100K - 500K$ instructions is supposed to execute in an MDC environment. Since the application will be distributed to a number of worker devices, it is split into a random size of smaller tasks containing instructions from 5000 − 125000. Now, to execute these instructions, worker devices are elected by the greedy algorithms in the system. We consider the capacity of a worker is randomly chosen from the range of $10,000 - 1,000,000$ units. For simplicity, we assumed 1 unit of resources is required to execute each instruction, where 1 unit of cost was estimated for every 10,000 units of resources. The maximum amount of bid ($\mathcal{B}_m^{max}$) for a task has been calculated based on the task size and the cost per unit of resources which has been varied according to the task size. The value of ($\mathcal{B}_m^{min}$) has been set to 0 in all experiments. To generate the simulation data, we have implemented an experimental testbed based on our preliminary works [12], [37]. In the experiments, the reputation threshold of a candidate worker was set to 0.6. The percentage of cloudlet utility λ, has been set to 20 %. All simulation experiments were conducted on a PC with an Intel Core i5 2.2 GHz processor and 8 GB memory running Windows 8.1. This is to note here that both the greedy algorithms run in polynomial time and the runtime of both algorithms is few milliseconds which has been ignored in the calculation of task execution quality in the simulation results.

To compare with the state-of-the-art works, at first, we estimate the execution time for the whole application in the user device. Then we compute improvement of QoE, payment savings, and user satisfaction based on execution time, bid amount and number of successful execution (without resubmission), respectively. Details of measuring each performance metric are explained in section V-B. This is to note that each simulation experiment was run for 500*s*, and the results collected from 50 runs with different random seed values were averaged to plot each data point with corresponding confidence interval in the graphs. A summary of the values and ranges of different simulation parameters is given in Table 3.

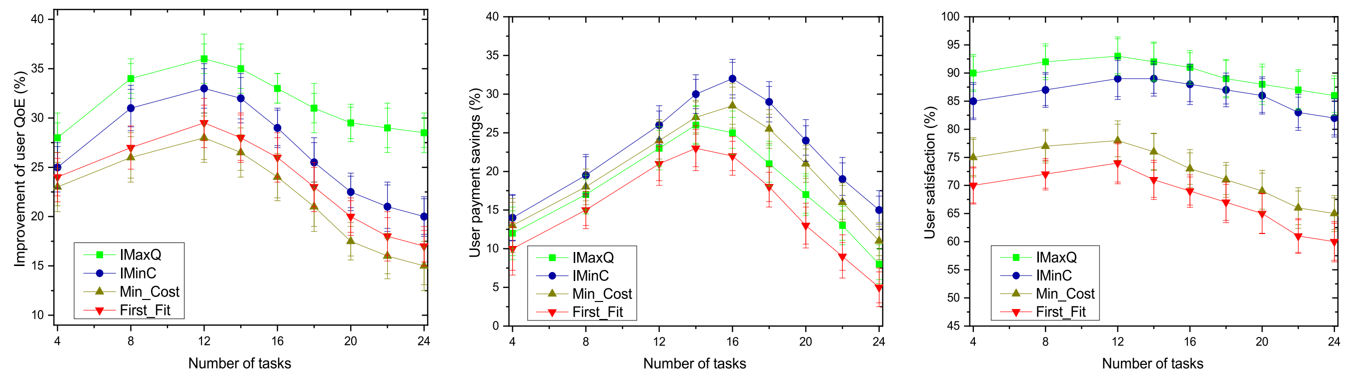
TABLE 3. Values of simulation parameters.

Parameter	Description
Number of tasks	$4 - 24$
Number of workers	$10 - 100$
Application size	$100 - 500K$ instructions
Task size	$5 - 125K$ instructions
Cost per unit resource	10^{-4} units
Total budget	$10 - 50$ units
Worker available resource	$10000 - 1000000$ units
Arrival rate of tasks	$1 - 3$ tasks/second (poisson)
Arrival rate of workers	$2 - 5$ workers/second (poisson)
Task deadline	$500 - 1500\ ms$
Reputation threshold (γ)	0.6
Cloudlet utility (λ)	20 % of worker bid cost
Simulation area	$50 \times 50\ m^2$
Simulation time	$500\ sec$

B. PERFORMANCE METRICS

We focused on the following performance metrics to evaluate the proposed methods and to compare them with other state-of-the-art methods.

- *Improvement of user QoE:* The user Quality-of-Experience (QoE) metric measures the task quality improvement of an offloaded task observed by the user. This gives the average percentage improvement of execution time in MDC compared to that of the user device. To compute it, at first, we estimate the execution time of the whole application in the user device. Then, we distribute the application among the selected worker devices based on the proposed greedy allocation algorithms. After getting the execution results, we compute execution time (including communication latency) for corresponding worker devices and the improvement of user QoE using Eq. 2.
- *User payment savings:* The user payment savings parameter reflects the surplus amount of the budget after the completion of payment and incentive of the worker devices. This is defined as the proportion of unused payment compared to the sanctioned budget of a user for a certain application execution; it is expressed as a percentage, i.e., $(1 - \frac{\mathcal{P}'}{\mathcal{P}}) \times 100\ \%$.
  To compute it, at first, we estimate the maximum budget (including cloudlet cost, bandwidth cost, resource compensation, etc.) for the whole user application to be executed in the MDC environment. Based on the bid amount in auction and execution time, the cloudlet disburses payments to the winner worker devices including incentives. It is to note that the eligibility of incentives is determined by Eq. 17. Finally, we compute the payment savings with respect to estimated budget and total expenditure including incentives and bid cost.
- *User satisfaction:* This is the percentage of successfully executed tasks (without resubmission) out of all offloaded tasks.



(a) Improvement of user QoE (b) User payment savings (c) User satisfaction

FIGURE 6. Impact of varying number of tasks.

C. RESULTS

This subsection provides the experimental results and discussion on the comparative performances of the studied systems.

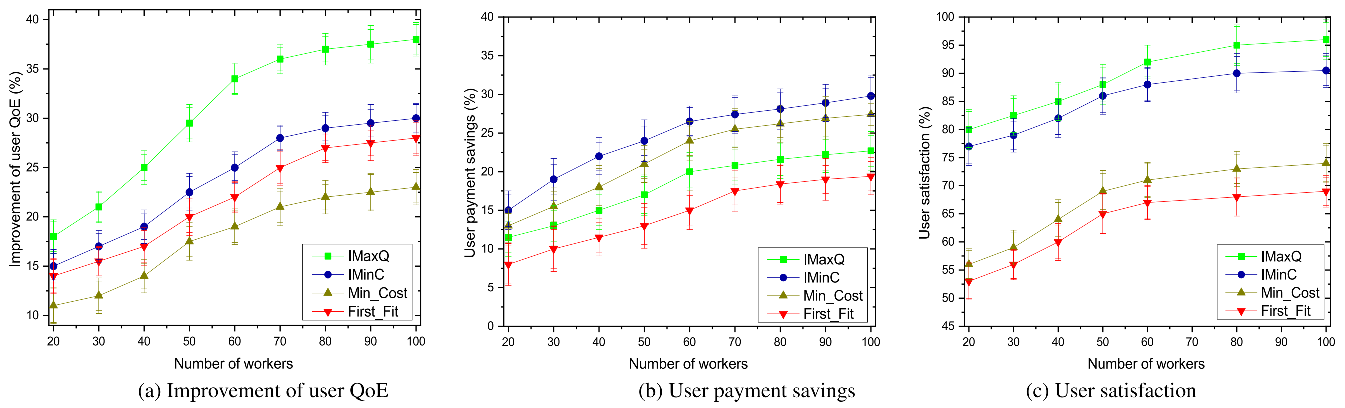
1) IMPACT OF VARYING NUMBER OF TASKS

In this experiment, we fix the number of worker devices to $|\mathcal{K}| = 50$ and application size to 200K, and we vary the number of tasks from $4 - 24$. The number of instructions in each task may vary from other tasks and due to fixed size application, the number of instructions of an individual task is reduced with increasing number of tasks.

Fig. 6(a) shows the improvement of user QoE achieved by the proposed incentive algorithms compared to existing approaches. We observe that the user QoE increases with the number of tasks, and it reaches its peak point when the task population is close to 12. This trend is reasonable since a few capable workers are able to bid and win the tasks at the beginning due to relatively large instruction size of the tasks. When size of an individual task is decreased, the number of capable worker devices for executing the task is increased that facilitated more parallel execution and enabled their executions on highly qualified workers till the saturation point. After that, admission of an additional number of tasks in the system forces it to allocate mid-level or even poor capacity and/or low-quality workers to allocate for the execution of tasks, causing degradation of user-QoE. The graphs also depict that the proposed *IMaxQ* offers the highest quality (35 %) as it is expected theoretically. Since *IMinC* and *Min-Cost* prefer workers with low cost rather than high quality and thus their performances are significantly less compared to *IMaxQ*. Similarly, due to the random selection of workers, *First-Fit* provides the worst user QoE among all.

On the other hand, in Figure 6(b), *IMinC* offers the highest user payment savings with an increasing number of tasks compared to all other incentive algorithms. In the beginning, a few tasks containing higher instruction sizes were bid by a limited number of capable workers at a high cost. Due to this monopolistic competition, such execution leads to small savings for the user. As the number of tasks increases, the amounts of savings also rises in all algorithms. This is due to the fact that more low-cost offering workers were able to compete that assisted better worker selection and execution quality with relatively lower bid cost, resulting higher cost savings for the user. Nevertheless, further increase of tasks ($|M| >= 16$) causes a sharp fall in savings since the system is bound to select workers with high bid costs to accommodate the increasing number of tasks. By comparison, *IMinC* provides the highest payment savings (31 %) and *Min-Cost's* user payment savings percentage is somewhat lower than that of *IMinC* because it pays worker devices with the immediate next bid winner's cost. *IMaxQ* cannot provide low cost execution due to the selection of higher quality workers. Due to the random selection of workers by *First-Fit*, it provides the least payment savings.

We also measured the satisfaction level of application users offered by the studied systems through on-time execution of tasks, as shown in Fig. 6(c). We observed that the user satisfaction rates for all studied algorithms increased with the number of tasks until peaking at approximately $|M| = 12$, and then they started to decrease slowly. Initially, the size of each task was relatively bigger due to a small number of tasks in an application. As a result, a few number of resource-rich workers were able to win the bid and provide successful execution. Dividing the application into more number of tasks instigated more quality workers with a better reputation get a chance to win bids, resulting in higher successful execution. Further increasing the number of tasks implies that each task contains a fewer number of instructions, demanding allocation to more workers of the system. This consequence diminishes user satisfaction due to allocations of tasks to relatively poor workers. Hence, user satisfaction is increased slightly at the beginning and is reduced gradually after reaching a pick point. In comparison, *IMaxQ* and *IMinC* provide better user satisfaction than *Min-Cost* and *First-Fit*, and *IMaxQ* provides the best. This is due to the consideration of worker reputation, which allocates a task to reliable devices in addition to available computation resources. Moreover, lack of consideration



(a) Improvement of user QoE (b) User payment savings (c) User satisfaction

FIGURE 7. **Impact of varying number of worker devices.**

of task dependency incurs unnecessary waiting time for a task (dependent on a different one for data input) and hampers user satisfaction in *Min-Cost* and *First-Fit*.

2) IMPACT OF VARYING NUMBER OF WORKER DEVICES

Availability of worker devices creates an opportunity to select more appropriate resources for executing a task, resulting in better system performance. To investigate the impact of worker devices, we fix the number of tasks to $|\mathcal{M}| = 20$ and application size to 200K and vary the number of worker devices from $20 - 100$.

Figure 7(a) shows that increasing the number of worker devices drives the average user QoE to rise sharply, and then it slows until it reaches saturation. This is due to the fact that with a fixed set of tasks ($|\mathcal{M}| = 20$), increasing worker devices create an opportunity to allocate resources from more qualified workers, resulting in better user QoE. However, increasing worker devices cannot remarkably improve the user QoE when the system contains many qualified workers ($|\mathcal{K}| > 60$) above the user demand. Our proposed *IMaxQ* and *IMinC* algorithms show better performances compared than the other. Task allocation to reputed workers with rich computing resources provides the finest result for *IMaxQ*, while *IMinC* sacrifices quality slightly to minimize execution cost.

A completely opposite phenomenon is illustrated in Figure 7(b), wherein the selection of worker devices with minimum bid cost provides superior performance for *IMinC* and *Min-Cost* compared to *IMaxQ* and *First-Fit*, while *IMinC* provides the maximum savings for a user. This figure also illustrates that user payment savings gradually increase due to the availability of a large number of worker devices offering relatively lower execution cost. However, availability of too many qualified workers ($|\mathcal{K}| > 60$) cannot improve the user savings further as the system reaches saturation.

Similarly, increasing worker devices creates more opportunity to improve successful task execution rate, which facilitates higher user satisfaction, as shown in Figure 7(c). From this figure, it is also evident that our proposed *IMaxQ* and *IMinC* algorithms maintain significantly superior results than the other algorithms due to the consideration of task dependency, worker reputation, and available computing resources.

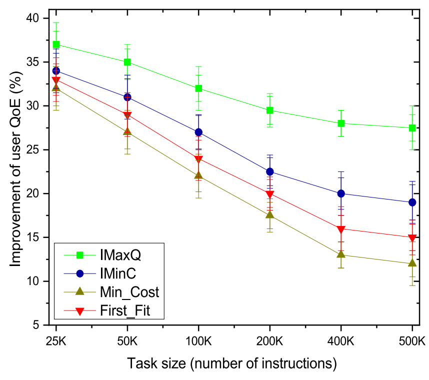
3) IMPACT OF VARYING TASK SIZE

In this experiment, we fixed the number of worker devices to $|\mathcal{K}| = 50$ and the number of tasks to $|\mathcal{M}| = 20$ and classified the application size into three categories: small ($25K - 50K$), medium ($100K - 200K$) and large ($400K - 500K$) tasks.

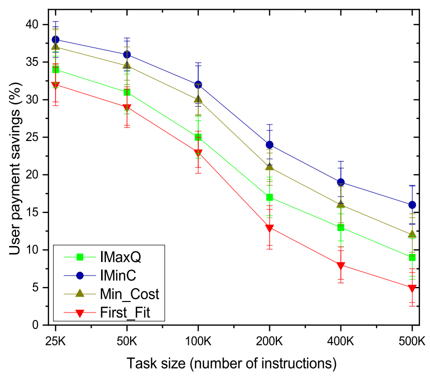
As shown in Figure 8(a) and Figure 8(b), increasing the size of a task results in degradation of both user QoE and payment savings. This is due to the fact that increasing task size reduces the percentage of qualified worker devices to execute the task, resulting in relatively resource-poor workers being hired. For this reason, user QoE and user payment savings decrease with growing task size. However, due to the consideration of worker reputation, task dependency, and available computation resources, our proposed *IMaxQ* and *IMinC* algorithms still outperform the *Min-Cost* and *First-Fit* models, while *IMaxQ* has superior performance for user QoE and *IMinC* provides maximum cost savings. For the same reason, user satisfaction also decreases with growing application task size, as illustrated in Fig. 8(c). However, for allocation of tasks to reputed worker devices, our proposed *IMaxQ* and *IMinC* provide significantly better results compared to other algorithms.

4) INCENTIVES FOR WORKER DEVICES

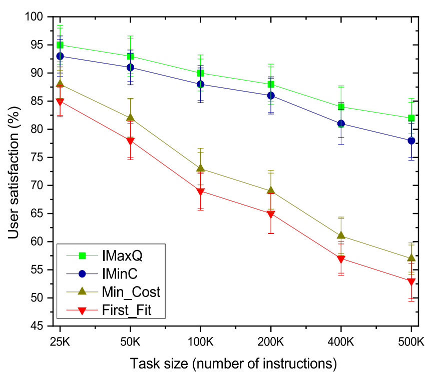
In this experiment, we plotted the average amount of incentives received by different winning worker devices for executing a single application having 20 units of budget with fifty worker devices ($|\mathcal{K}| = 50$). Figure 9(a) shows the impact of increasing the number of tasks on the distribution of amounts of the worker bids, incentives and savings. Initially, size of an individual task was relatively bigger. As a result, a few workers were able to win the bid with high bid cost, resulting small payment savings and incentives. Dividing the application into more number of small-sized tasks invited more quality workers with lower bid cost, resulting higher payment



(a) Improvement of user QoE



(b) User payment savings

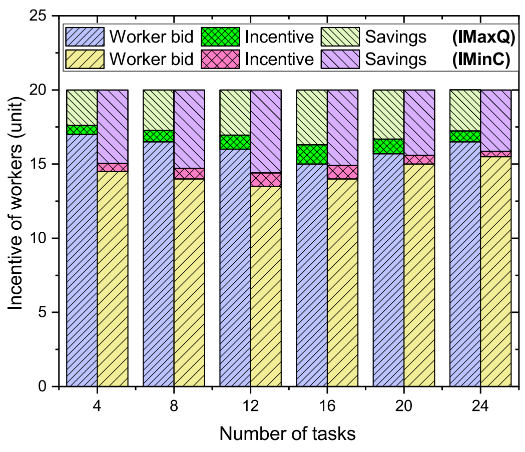


(c) User satisfaction

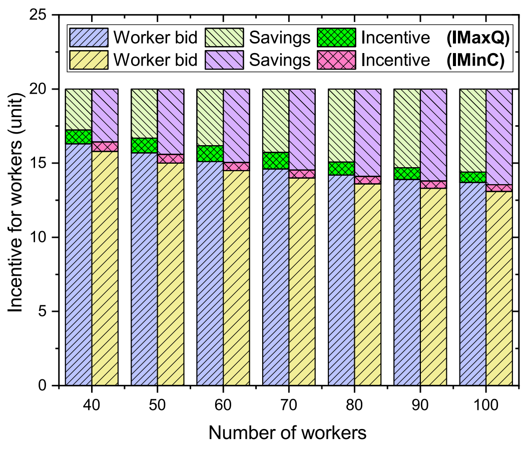
FIGURE 8. Impact of varying task sizes.

savings and incentives. However, further increasing number of tasks diminishes payment savings and incentives due to execution of tasks with relatively high bid cost workers. More specifically, in such situation, number of good quality workers is not enough to execute so many tasks. For the same reason, the graph follows similar trend for user payment savings. Comparatively, *IMaxQ* pays more incentive than *IMinC* as it prefers higher quality. Incentive performances for the *Min-Cost* and *First-Fit* algorithms are not presented here because these algorithms do not pay any additional incentives to workers above their bid amounts.

Similarly, increasing the number of worker devices causes more competition among the workers with fixed number of tasks ($|\mathcal{M}| = 20$), resulting in higher user QoE with small bid cost, as illustrated in Figure 9(b). Therefore, the user gets the opportunity to pay a higher incentive for the worker resources. However, further increasing the number of workers (>70) resulted high quality execution from the worker devices though the incentive amount is decreased. At this point, due to high competition among the workers, the difference between the SLA quality and the provided quality of the workers becomes very little and hence the incentive amount is decreased even though the payment savings is increased. Though *IMaxQ* pays higher incentives than *IMinC*, the savings amount in Figure 9(b) is much higher compared to its counterpart shown in Figure 9(a). It is worth noting that although our proposed *IMaxQ* and *IMinC* algorithms provide incentives to workers, the savings are still higher than from other state-of-the-art-works due to the payment strategy. Our proposed resource allocation algorithms provide payment as the workers bid price, whereas other mechanisms use the next winner's bid cost to pay a worker, causing higher payment. However, our proposed allocation algorithms incentivize only the qualified workers, resulting in savings. Our in-depth look into the simulation trace files also reveals that, due to the payment of additional incentives, the participation of resourceful workers also increases gradually with the growing number of worker devices and execution of tasks with these worker devices upswing user-QoE.



(a) Impact of increasing number of tasks



(b) Impact of increasing number of workers

FIGURE 9. Incentive for workers.

In summary, the amount of incentive received by a worker device is proportional to how much less time it takes for it to execute a task than the user deadline. Our in-depth analysis on results in the simulation trace file revealed that approximately 40 % of the workers are incentivized with an ample amount for offering excellent qualities of execution. Moreover, approximately 30 % of the workers are incentivized insignificantly due to bidding higher cost and offering just above SLA quality. The remaining workers are given negligible (or zero) incentives for just-in-time execution. Such a mechanism develops a win-win environment for users and workers, increasing the sustainability of the MDC system.

VI. CONCLUSION

In this work, we developed a QoE and Incentive-aware dynamic resource allocation framework for MDC, where mobile worker devices are incentivized in line with their execution qualities of user application tasks. Furthermore, consideration of resource capacity, reputation, and bid cost of worker devices helped our allocation algorithms to harvest maximum benefits (in terms of quality or cost) through efficient utilization and management of the idle resources. Incentivizing the worker devices proportional to offered task execution quality significantly increased the participation of the worker devices and improved the quality of experience of the users. The simulation results clearly indicated that the proposed system can maximize user QoE by as much as 35 % while minimizing cost by up to 30 %. The results also revealed that increasing the number of workers in the system can steadily improve user QoE, payment savings, and satisfaction, whereas larger task sizes can adversely affect MDC performance.

In this work, we have considered a fixed budget for the execution of an application that shrinks the worker selection due to budget constraints. In the future, given that the user budget for execution of application tasks is dynamically adjustable rather than being fixed, developing a resource allocation algorithm in an MDC environment that can further increase user QoE could be an interesting problem to solve. Moreover, other multi-objective evolutionary approaches can be considered to find an optimal solution with more number of instances to minimize the runtime complexity of the formulated MOLP objective function.

REFERENCES

[1] Q. Shen, X. Liang, X. Shen, X. Lin, and H. Y. Luo, "Exploiting geo-distributed clouds for a e-health monitoring system with minimum service delay and privacy preservation," *IEEE J. Biomed. Health Informat.*, vol. 18, no. 2, pp. 430–439, Mar. 2014.

[2] M. Golkarifard, J. Yang, Z. Huang, A. Movaghar, and P. Hui, "Dandelion: A unified code offloading system for wearable computing," *IEEE Trans. Mobile Comput.*, vol. 18, no. 3, pp. 546–559, Mar. 2019.

[3] K. Xie, X. Wang, G. Xie, D. Xie, J. Cao, Y. Ji, and J. Wen, "Distributed multi-dimensional pricing for efficient application offloading in mobile cloud computing," *IEEE Trans. Services Comput.*, vol. 12, no. 6, pp. 925–940, Nov. 2019.

[4] Y. Zhang, J. Yan, and X. Fu, "Reservation-based resource scheduling and code partition in mobile cloud computing," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2016, pp. 962–967.

[5] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura, "Serendipity: Enabling remote computing among intermittently connected mobile devices," in *Proc. 13th ACM Int. Symp. Mobile Ad Hoc Netw. Comput. (MobiHoc)*, 2012, pp. 145–154.

[6] E. Miluzzo, R. Cáceres, and Y.-F. Chen, "Vision: MClouds–computing on clouds of mobile devices," in *Proc. 3rd ACM Workshop Mobile Cloud Comput. Services (MCS)*, 2012, pp. 9–14.

[7] A. Mtibaa, K. A. Harras, and A. Fahim, "Towards computational offloading in mobile device clouds," in *Proc. IEEE 5th Int. Conf. Cloud Comput. Technol. Sci.*, Dec. 2013, pp. 331–338.

[8] M. Le and Y.-W. Kwon, "Utilizing nearby computing resources for resource-limited mobile devices," in *Proc. Symp. Appl. Comput.*, Apr. 2017, pp. 572–575.

[9] V. Balasubramanian and A. Karmouch, "An infrastructure as a service for mobile ad-hoc cloud," in *Proc. IEEE 7th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2017, pp. 1–7.

[10] B. Venkatraman, F. A. Zaman, and A. Karmouch, "Optimization of device selection in a mobile ad-hoc cloud based on composition score," in *Proc. 2nd Int. Conf. Commun. Syst., Comput. IT Appl. (CSCITA)*, 2017, pp. 257–262.

[11] A. Shye, B. Scholbrock, G. Memik, and P. A. Dinda, "Characterizing and modeling user activity on smartphones: Summary," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 38, no. 1, pp. 375–376, 2010.

[12] S. Saha, M. A. Habib, and M. A. Razzaque, "Compute intensive code offloading in mobile device cloud," in *Proc. IEEE Region Conf. (TENCON)*, Nov. 2016, pp. 436–440.

[13] X. Guo, L. Liu, Z. Chang, and T. Ristaniemi, "Data offloading and task allocation for cloudlet-assisted ad hoc mobile clouds," *Wireless Netw.*, vol. 24, no. 1, pp. 79–88, Jan. 2018.

[14] F. Lu, L. Gu, L. T. Yang, L. Shao, and H. Jin, "Mildip: An energy efficient code offloading framework in mobile cloudlets," *Inf. Sci.*, vol. 513, pp. 84–97, Mar. 2020.

[15] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 945–953.

[16] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. Wireless Commun.*, vol. 11, no. 6, pp. 1991–1995, Apr. 2012.

[17] S. A. Noor, R. Hasan, and M. M. Haque, "CellCloud: A novel cost effective formation of mobile cloud based on bidding incentives," in *Proc. IEEE 7th Int. Conf. Cloud Comput.*, Jun. 2014, pp. 200–207.

[18] X. Wang, X. Chen, W. Wu, N. An, and L. Wang, "Cooperative application execution in mobile cloud computing: A stackelberg game approach," *IEEE Commun. Lett.*, vol. 20, no. 5, pp. 946–949, May 2016.

[19] L. Duan, T. Kubo, K. Sugiyama, J. Huang, T. Hasegawa, and J. Walrand, "Motivating smartphone collaboration in data acquisition and distributed computing," *IEEE Trans. Mobile Comput.*, vol. 13, no. 10, pp. 2320–2333, Oct. 2014.

[20] X. Wang, X. Chen, and W. Wu, "Towards truthful auction mechanisms for task assignment in mobile device clouds," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, May 2017, pp. 1–9.

[21] L. Tang, S. He, and Q. Li, "Double-sided bidding mechanism for resource sharing in mobile cloud," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 1798–1809, Feb. 2017.

[22] H. Moore, *MATLAB for Engineers*. London, U.K.: Pearson, 2017.

[23] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, "Femto clouds: Leveraging mobile devices to provide cloud service at the edge," in *Proc. IEEE 8th Int. Conf. Cloud Comput.*, Jun. 2015, pp. 9–16.

[24] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[25] X. Lin, J. Jiang, C. H. Y. Li, B. Li, and B. Li, "Circa: Collaborative code offloading among multiple mobile devices," *Wireless Netw.*, vol. 26, no. 2, pp. 823–841, Feb. 2020.

[26] M. Guiguis, Q. Gu, T. Penner, L. Tammineni, T. Langford, A. Rivera-Longoria, A. Johnson, and B. V. Slyke, "Assignment and collaborative execution of tasks on transient clouds," *Ann. Telecommun.*, vol. 73, nos. 3–4, pp. 251–261, Apr. 2018.

[27] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4804–4814, Jun. 2019.

[28] N. Fernando, S. W. Loke, and W. Rahayu, "Computing with nearby mobile devices: A work sharing algorithm for mobile edge-clouds," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 329–343, Apr. 2019.

[29] U. Saleem, Y. Liu, S. Jangsher, Y. Li, and T. Jiang, "Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 360–374, Jan. 2021.

[30] R. Yadav, W. Zhang, O. Kaiwartya, H. Song, and S. Yu, "Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 14198–14211, Dec. 2020.

[31] R. Yadav and W. Zhang, "MeReg: Managing energy-SLA tradeoff for green mobile cloud computing," *Wireless Commun. Mobile Comput.*, vol. 2017, pp. 1–11, Dec. 2017.

[32] Y. Liu, C. Xu, Y. Zhan, Z. Liu, J. Guan, and H. Zhang, "Incentive mechanism for computation offloading using edge computing: A Stackelberg game approach," *Comput. Netw.*, vol. 129, pp. 399–409, Dec. 2017.

[33] X. Wang, Y. Sui, J. Wang, C. Yuen, and W. Wu, "A distributed truthful auction mechanism for task allocation in mobile cloud computing," *IEEE Trans. Services Comput.*, vol. 14, no. 3, pp. 628–638, May 2021.

[34] Q. Wang, S. Guo, Y. Wang, and Y. Yang, "Incentive mechanism for edge cloud profit maximization in mobile edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.

[35] G. Li and J. Cai, "An online incentive mechanism for collaborative task offloading in mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 624–636, Jan. 2020.

[36] J. He, D. Zhang, Y. Zhou, and Y. Zhang, "A truthful online mechanism for collaborative computation offloading in mobile edge computing," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4832–4841, Jul. 2020.

[37] S. Saha, M. A. Habib, T. Adhikary, M. A. Razzaque, and M. M. Rahman, "Tradeoff between execution speedup and reliability for compute-intensive code offloading in mobile device cloud," *Multimedia Syst.*, vol. 25, no. 5, pp. 577–589, Oct. 2019.

[38] K. Xu, Y. Zhang, X. Shi, H. Wang, Y. Wang, and M. Shen, "Online combinatorial double auction for mobile cloud computing markets," in *Proc. IEEE 33rd Int. Perform. Comput. Commun. Conf. (IPCCC)*, Dec. 2014, pp. 1–8.

[39] S. Sarker, M. A. Razzaque, M. M. Hassan, A. Almogren, G. Fortino, and M. Zhou, "Optimal selection of crowdsourcing workers balancing their utilities and platform profit," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8602–8614, Oct. 2019.

[40] B. Kantarci and H. T. Mouftah, "Reputation-based sensing-as-a-service for crowd management over the cloud," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2014, pp. 3614–3619.

[41] Q. Wu, X. Zhang, M. Zhang, Y. Lou, R. Zheng, and W. Wei, "Reputation revision method for selecting cloud services based on prior knowledge and a market mechanism," *Sci. World J.*, vol. 2014, Feb. 2014, Art. no. 617087.

[42] J. Liu, E. Ahmed, M. Shiraz, A. Gani, R. Buyya, and A. Qureshi, "Application partitioning algorithms in mobile cloud computing: Taxonomy, review and future directions," *J. Netw. Comput. Appl.*, vol. 48, pp. 99–117, Feb. 2015.

[43] S. Saha and M. S. Hasan, "Effective task migration to reduce execution time in mobile cloud computing," in *Proc. 23rd Int. Conf. Autom. Comput. (ICAC)*, Sep. 2017, pp. 1–5.

[44] T. Dbouk, A. Mourad, H. Otrok, H. Tout, and C. Talhi, "A novel ad-hoc mobile edge cloud offering security services through intelligent resource-aware offloading," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 4, pp. 1665–1680, Dec. 2019.

[45] Y.-H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," *IEEE Trans. Mobile Comput.*, vol. 16, no. 11, pp. 3056–3069, Nov. 2017.

[46] D. Dereniowski and W. Kubiak, "Shared multi-processor scheduling," *Eur. J. Oper. Res.*, vol. 261, no. 2, pp. 503–514, Sep. 2017.

[47] T. Sigwele, A. S. Alam, P. Pillai, and Y. F. Hu, "Evaluating energy-efficient cloud radio access networks for 5G," in *Proc. IEEE Int. Conf. Data Sci. Data Intensive Syst.*, Dec. 2015, pp. 362–367.

SAJEEB SAHA (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees from the University of Dhaka, Bangladesh, in 2007 and 2009, respectively, where he is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering. His research interests include mobile device cloud, mobile cloud computing, the Internet of Things, and energy optimization. He is a member of the ISOC, the IEEE Computer Society, and the IEEE Communications Society.

MD. AHSAN HABIB (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees from the Islamic University of Technology, Bangladesh, in 2003 and 2012, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, University of Dhaka, Bangladesh. His research interests include wireless sensor networks, mobile cloud computing, and the Internet of Things. He is a member of the IEEE Computer Society, ACM, IEB, and ISOC.

TAMAL ADHIKARY (Member, IEEE) received the B.Sc. and M.Sc. degrees from the University of Dhaka, Bangladesh, in 2012 and 2013, respectively. His research interests include mobile device cloud, mobile cloud computing, the Internet of Things, and energy optimization. He is a member of the Internet Society (ISOC), the IEEE Computer Society, and the IEEE Communications Society.

MD. ABDUR RAZZAQUE (Senior Member, IEEE) received the B.S. and M.S. degrees from the University of Dhaka, Bangladesh, in 1997 and 1999, respectively, and the Ph.D. degree from Kyung Hee University, South Korea, in 2009. From 2010 to 2011, he was a Research Professor with Kyung Hee University. He worked as a Visiting Scholar with Stratford University, USA, in 2017. His research interests include the area of modeling, analysis and optimization of wireless networking protocols and architectures, the Internet of Things, and cloud computing. He is a member of the IEEE Communications Society and the IEEE Computer Society. He is also the TPC chair of many conferences, an Editor of *Journal of Network and Computer Applications (JNCA)*, and an Associate Editor of IEEE Access.

MD. MUSTAFIZUR RAHMAN received the B.Sc. degree (Hons.) in applied physics and electronics and the M.Sc. degree in computer science from the University of Dhaka, Bangladesh, in 1997 and 1999, respectively, and the Ph.D. degree in computer engineering from Kyung Hee University, South Korea, in 2009. He is currently a Professor with the Department of Computer Science and Engineering, University of Dhaka. His research interest includes multiple access protocols.

METEB ALTAF received the Ph.D. degree from Brunel University London, London, U.K., in 2009. Since 2009, he joined the King Abdulaziz City for Science and Technology (KACST), as an Assistant Research Professor. He was appointed as the Director Assistant of Administrative Affairs and Scientific Affairs in the National Center for Robotics and Intelligent Systems. After that he was appointed as the Director of the National Robotics Technology and Intelligent Systems Center, before it become known as the National Center for Robotics Technology and the Internet of Things. Then, he promoted as a Research Associate Professor. In the meantime, he became the Director of the Innovation Center for Industry 4.0 at the KACST. He is currently the Director of the Advanced Manufacturing and Industry 4.0 Center, and lecturing at the Department of Biomedical Technology, King Saud University, and supervised more than 20 research projects locally and internationally, as technology transfer project. During his career life, he published number of articles in different well-known ISI journals and in well-recognized conferences.

MOHAMMAD MEHEDI HASSAN (Senior Member, IEEE) received the Ph.D. degree in computer engineering from Kyung Hee University, South Korea, in February 2011. He is currently a Full Professor with the Department of Information Systems, College of Computer and Information Sciences (CCIS), King Saud University (KSU), Riyadh, Saudi Arabia. He has authored and coauthored around more than 180 publications, including refereed IEEE/ACM/Springer/Elsevier journals, conference papers, books, and book chapters. Recently, his four publications have been recognized as the ESI Highly Cited Papers. His research interests include cloud computing, edge computing, the Internet of Things, body sensor networks, big data, deep learning, mobile cloud, smart computing, wireless sensor networks, 5G networks, and social networks. He has served as the chair and a technical program committee member for international conferences.