Strategy-Proof Online Mechanisms for Weighted AoI Minimization in Edge Computing

Abstract—**Real-time information processing is increasingly needed in many areas, and age of information (AoI), as a new metric, has received considerable attentions to evaluate the performance of real-time information processing systems. In order to reduce the AoI, edge computing is getting prosperous in both academia and industry. Because of the considerable deployment cost and the resulting resource limitation in edge computing, a proper pricing mechanism is necessary to fully utilize the resources and reduce the AoI. However, there are two challenges to conduct this mechanism: 1) the values of the real-time computing tasks are usually private information of users, which may be misreported, and 2) the value discounts with time and hence the traditional mechanisms become invalid. Therefore, we extend the classical Myerson Theorem to the online setting with time discounting tasks values, and accordingly propose an online auction mechanism, called PreDisc, including an allocation rule and a payment rule. In PreDisc, a preemption factor is employed to trade off the newly arrived tasks and ongoing tasks, by assigning the ongoing tasks high priorities. We prove that PreDisc guarantees the economic property of strategy-proofness and achieves a constant competitive ratio. We conduct extensive simulations and the results demonstrate that PreDisc outperforms the traditional mechanisms, in terms of both weighted AoI and revenue of edge service providers. Compared with the optimal solution in offline VCG mechanism, our mechanism consumes much less computation time with only a slight performance loss.**

I. Introduction

In recent years, real-time information processing is prevailing in many areas, such as autonomous vehicles [1], online gaming [2], virtual reality (VR) [3] and multi-robot systems [4]. In order to evaluate the performance of real-time information processing systems, a new metric called age of information (AoI) was proposed in [5], and has received considerable attentions recently [6]–[10]. Different from traditional performance metrics like delay and throughput, the metric of AoI takes the freshness of decision-making information into account. For example, if the user sends tasks with a very low frequency, the system performs well on delay but poorly on AoI, because a lack of timely decision update makes the received decision out of date. Thus, AoI is widely adopted as a more reasonable metric in real-time computing applications.

The traditional centralized cloud computing mode does not satisfy the stringent requirement of AoI in real-time information processing system, because end devices have to send data to remote cloud for processing with a high network delay. Edge computing [11], as a new computing paradigm, is quite attracted to further reduce the AoI in real-time applications. In edge computing, edge servers (also called cloudlets) are deployed near end devices, and such physical proximity can significantly reduce transmission delay and also AoI. For example, in autonomous vehicle systems with cloud computing mode, the transmission time between the vehicle and the remote cloud server is about 150ms, while with the assistance of edge servers, ultra-low latency (less than 1ms) can be achieved [12], [13]. Many real-time applications like online gaming and VR also have improvements in AoI and hence in system performance and user experience by using edge computing mode [2].

Although edge computing achieves attractive performance improvement in terms of reducing AoI, it also introduces additional cost for distributed deployment and maintenance [11], [14]. Due to this cost constraint, the computation resources of edge servers are usually limited, which may result in the degradation of overall service performance [15]. Therefore, on one hand, it is a promising idea to consider the paradigm of edge-cloud collaboration, combining the low latency of edge and the sufficient resources of remote cloud [12], [16]. On the other hand, a proper pricing mechanism is necessary to fully utilize the limited edge resources and to compensate the cost of edge service providers [17].

It is quite challenging to design a pricing mechanism for edge services in real-time information processing systems. The service provider would like to efficiently manage the limited edge resources by assigning large weights (or priorities) to urgent tasks. We measure the extent of task urgency by a concept of *value* (please refer to Section II for a specific definition), which is related to private information of users, such as the driving speed and the surrounding environment in self-driving systems. As the values of tasks are the private information of users, they would manipulate this information, if doing so can increase the priorities of their tasks, resulting in the chaos of market and then the degradation of resource utilization. Therefore, the pricing mechanisms should be carefully designed to resist the strategic behaviors of users. In previous literature, a dynamic pricing rule [18] is studied to minimize the AoI for a crowdsourcing platform, encouraging users to sample the real-time information in different rates. However, the sampling rate can be easily detected, and hence there are no strategic behaviors in this case.

Other than the difficulty in guaranteeing the strategy-proofness[1], the dynamic property and the time discounting values of tasks also bring obstacles to the design of pricing mechanisms. On one hand, since the tasks of users arrive at the edge in an online manner, the edge server needs to schedule them online, without the knowledge of future tasks. The classical Vickrey-Clarke-Groves (VCG) mechanism [19]–[21] could not be directly applied into this online setting, as it needs to calculate the optimal offline allocation and hence is normally computationally intractable. On the other hand, since

[1]In a strategy-proof mechanism, the users would truthfully reveal their private information, *i.e.*, the values of tasks in our context. Please refer to Section II for detailed definition.

the AoIs of real-time decisions increase with time, the values of tasks would discount if they are delayed for execution. The time changing value enables the users to have a large space to further manipulate the mechanisms, i.e., users can win the resource at different time slots by misreporting their values. The existing online mechanisms [22], [23], by which each winning task is charged a predefined payment without considering the time-discounting value, would be no longer strategy-proof, and thus is inapplicable for AoI minimization under strategic environments.

To address these challenges, in this paper, we adopt a cloud-edge collaborative framework to optimize the weighted AoI of real-time decision tasks. The edge servers are employed to conduct urgent tasks, and the remote cloud server is considered as a backup mode to make decisions for users when the edge services are not available. We further propose an online auction mechanism for weighted AoI minimization, where users arrive at the auction dynamically, submit their tasks and corresponding task values to the edge server, and wait for the timely results of decisions before a certain deadline. Based on the reported values, the edge service providers calculate the reductions of weighted AoI for tasks at each time slot, and schedule the tasks to execute, with the goal of minimizing the overall weighted AoIs of all tasks. The edge service provider also determines the prices for users to guarantee the property of strategy-proofness, and then the users pay for the edge service at the required price.

The main contributions of this paper are summarized as follows.

- We creatively convert the problem of AoI minimization into the online optimization of social welfare, and also reveal the corresponding difficulty in mechanism design due to the property of time discounting values of tasks. It also builds a bridge between the AoI research and the mechanism design research.
- We extend the celebrated Myerson theorem [24] to the online pricing problem with the setting of time discounting task values, which provides a fundamental criterion for the mechanism design in AoI optimization problems.
- We propose a Preemption factor-based pricing mechanism with time Discounting task values (PreDisc) to allocate computing resources on the edge server. This mechanism assigns a high virtual value to ongoing tasks to avoid unnecessary preemptions of newly arrived tasks. PreDisc takes a desirable tradeoff between preemption and non-preemption. Our theoretical analysis shows that PreDisc guarantees both strategy-proofness and constant competitive ratio.
- We evaluate the performance of our proposed mechanism with extensive experiments. The evaluation results demonstrate that PreDisc outperforms the existing FCFS and LCFS mechanisms, and is very close to the optimal solution of offline VCG mechanism.

The paper is organized as follows. Section II introduces the model and the basic background knowledge. Section III characterizes the property of strategy-proofness. Section IV and Section V focus on the detailed design of PreDisc. In Section IV, we introduce the allocation and payment rules in PreDisc for tasks with unit edge execution time, and then give an analysis for the upper bound of competitive ratio compared with the offline optimal solution. Section V extends our mechanism to the general cases. In Section VI, we give the simulation results on average AoI and the revenue of edge service provider. Section VII reviews the related works. Finally, we conclude this paper in Section VIII.

II. Preliminaries

In this section, we introduce the model of online auction mechanism with time discounting task values in the context of edge computing, and briefly review the related solution concepts used in this paper from game theory.

A. System Model

We consider a cloud-edge collaborative computing framework including two components to facilitate users to make real-time decisions. A cloud server with adequate computing resources is normally far away from users, so the timeliness cannot be guaranteed if only use the cloud server for decision making. In contrast, a nearby edge server has a timely response for users, but can only support a certain amount of tasks simultaneously due to the limited edge resources.

A user communicates with the cloud server periodically in a normal mode. For instance, as illustrated by the blue dashed line in Fig. 1, user 1 sends a task to the cloud server at each time interval Δt, and receives a decision feedback after a time period T_c. The overall processing time from the cloud server (including the network delay and the computing time) is usually long, because the remote cloud is far away from the user. We assume that the overall cloud processing time T_c is identical for all users as the time differences among users are negligible compared with the large overall processing time. The *Age of Information (AoI)* for a user at a certain time is defined as the difference between this time and the generating time of the latest received decision-making result. For example, in Fig. 1, at time t_3, the AoI of user 1 is T_c, since the newly received decision is generated before a time period T_c. After time t_3, the AoI increases over time, reaches the highest AoI $T_c + \Delta t$ at time t_4, and then drops to T_c since the next decision is received. Thus, without the involvement of an edge server, the AoI fluctuates steadily at a relatively high level (see the blue solid line in Fig. 1).

When a user encounters an emergency task (*e.g.*, for a self-driving system, some urgent traffic situations need timely decisions), the user also sends the task to a nearby edge server, looking for a quick response. For example, in Fig. 1, at time t_2, user 2 sends the task to both the cloud and the edge server, and receives a quick feedback from the edge server after time T_e (also from the cloud server after time T_c). The communication time between users and the edge server has an ultra-low latency, so it is reasonable to omit the communication time. We consider T_e as only the computing time on the edge server. Also due to the property of ultra-low latency, T_e is always smaller than T_c, and hence AoI can be reduced with the help of edge server, as shown by the red solid line in

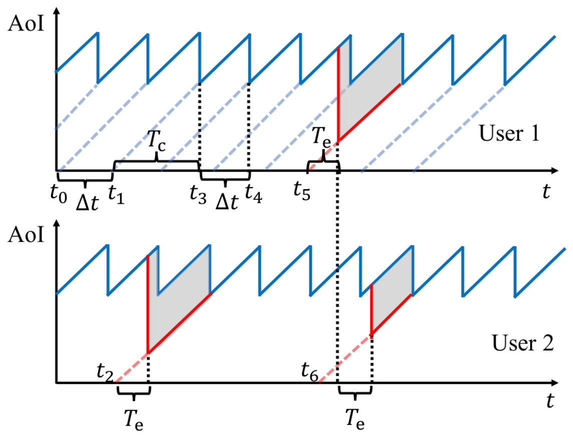


Fig. 1. The illustration of AoI for two users.

Fig. 1. Furthermore, as only emergency tasks are uploaded to the edge server, we assume the emergency tasks from the same user are non-overlapping with each other. For tasks with different computing complexity, we consider a transparent service mode, that is, the edge server completes them within an identical execution time via allocating different computing resources. Thus, the task i from the user j would demand $m_{i,j}$ units of resources, which is a public information determined by the edge server. We assume the resource demand $m_{i,j}$ to be discrete for simplicity in our model. Since computing resources at the edge are costly and limited, there may be conflicts among users in resource usage. For example, at time t_5 in Fig. 1, user 1 sends a task to an edge server, and shortly after that, user 2 also sends a task to the same edge server at time t_6. However, the edge server does not have enough computing resources to satisfy the demands for both users, and needs to schedule the tasks from these two users in an appropriate way. In addition, if T_e is larger than 1, a more emergency task may also preempt another ongoing task for a better utilization of edge resources[2].

As the timeliness of decisions is critical for the success of real-time information processing applications, *e.g.*, it may influence the safety of self-driving cars or the user experience in interactive gaming, the objective of each user j is to minimize the weighted average AoI over time, denoted as $\overline{A}_j$. The weight captures the extent of emergency of tasks, and can also be interpreted as the amount of money the user is willing to pay to exchange for the decrease of AoI. For each urgent task i, the value (*i.e.*, weight) $v_{i,j}$ is reported by the user j, and it may depend on many kinds of factors. For example, in an autonomous vehicle system, the urgency of a task depends on the driving speed, the vehicle performance, the surrounding environment, and so on. Since these factors are only accessible by the user, she is able to misreport the value $v_{i,j}$ for her own interest, e.g., declaring a large value to increase the priority of her task, and reduce the weighted AoI. Such a selfish behavior would degrade the system performance of the edge service, as a more urgent task may be preempted by a non-urgent task with a misreported high value. With such a consideration, we leverage an auction mechanism to incentivize the users to truthfully reveal their private information, and to efficiently allocate the limited edge resources to minimize the weighted average AoI of all users.

B. Problem Formulation

We consider the edge server with W units of reusable homogeneous resources in a finite time horizon, which can be further divided into T time slots with equal length: $\mathbb{T} = \{1, 2, \cdots, T\}$. Suppose the set of tasks[3] produced by user j is U_j, task $i \in U_j$ arrives at time slot $a_{i,j}$, and then it should be completed before a deadline at departure time $d_{i,j} = a_{i,j} + T_c$, because the decision made by the edge server becomes useless when the decision from the cloud server is also received after T_c time slots. We denote $\mathbb{T}_{i,j}$ as the set of all time slots in $[a_{i,j}, d_{i,j}]$ for task i of user j, and denote the set of other time slots $t \notin \cup_{i\in U_j}\mathbb{T}_{i,j}$ by $\widetilde{\mathbb{T}}_j$. To calculate the weighted average AoI $\overline{A}_j$ of user j, we denote the age at time t as $A_j(t)$, the age produced by the cloud server (*i.e.*, the blue solid line in Fig. 1) as $C_j(t)$, and the age produced by the edge server (*i.e.*, the red solid line in Fig. 1) as $E_j(t)$. If the AoI at time slot t is not produced by the edge server, we set $E_j(t) = +\infty$. With these definitions, we can have

$$A_j(t) = \min\{E_j(t), C_j(t)\},$$

and we can then get the weighted average AoI,

$$\begin{aligned}\overline{A}_j &= \frac{1}{T}\left(\sum_{t\in\widetilde{\mathbb{T}}_j} 1 \times A_j(t) + \sum_{i\in U_j}\sum_{t\in\mathbb{T}_{i,j}} v_{i,j}\times A_j(t)\right)\\ &= \frac{1}{T}\left(\sum_{t\in\widetilde{\mathbb{T}}_j} C_j(t) + \sum_{i\in U_j}\sum_{t\in\mathbb{T}_{i,j}} v_{i,j}\times C_j(t)\right.\\ &\quad \left. - \sum_{i\in U_j}\sum_{E_j(t)<C_j(t)} v_{i,j}\times (C_j(t) - E_j(t))\right).\end{aligned}$$

As the first two items in the brackets are constants and the tasks are non-overlapping, we only need to maximize the third item $\sum_{E_j(t)<C_j(t)} v_{i,j} \times (C_j(t) - E_j(t))$ for each task independently, which represents the reduction of the weighted AoI during the current task interval, *i.e.*, each of the shadow areas in Fig. 1. For easy presentation, we duplicate a user, also called as an agent, for each task, and use v_i directly to denote $v_{i,j}$. Suppose the edge server starts to execute the task i at time $t_{i,j}$ without an interruption in the following T_e time slots, we can then obtain

$$\sum_{E_i(t)<C_i(t)} v_i \times (C_i(t) - E_i(t)) = v_i \times \sum_{\substack{t>t_{i,j}+T_e\\ t<a_i+T_c}} (C_i(t) - E_i(t))$$

[2]In such a case, we assume the preempted task needs to start over if it is selected next time.

[3]As we focus on the emergency tasks on the edge in the following sections, we do not distinguish “task” and “emergency task" from now on.

which is a function with respect to starting time $t_{i,j}$. Thus, we denote

$$\begin{aligned} v_i(t) &= v_i \times f_i(t) \\ &= v_i \times \sum_{\substack{t'>t+T_e \\ t'<a_i+T_c}} (C_i(t') - E_i(t')) \end{aligned} \quad (1)$$

as the task value if it starts to execute at time slot t, with $a_i \leq t \leq a_i + T_c - T_e$. We do not restrict $C_i(t')$ and $E_i(t')$ to any specific (e.g., linear) format, but only require that each of these two functions has the identical format for all tasks. Since we have $E_i(t) < C_i(t)$ during the considered time interval, we can get that $f_i(t)$ is non-negative and non-increasing, meaning that the task value is discounting over time. Some possible function $f_i(t)$ could be $f_i(t) = \eta^{(t-a_i)}$ or $f_i(t) = 1 - \beta(t - a_i)$, where parameters $\eta \in (0,1)$ and $\beta \in (0,1)$ are shared by all agents. We note that all users are associated with the identical format of function $f_i(t)$, but could have different parameter of the arrival time a_i, Without loss of generality, we normalize $f_i(a_i) = 1$.

With the concept of task value, we can further formulate the problem of weighted AoI minimization as follows. There are n agents $\mathbb{N} = \{1, 2, \cdots, N\}$ arriving at the system in a random order. Each agent $i \in \mathbb{N}$ arrives at time a_i, and demands for m_i resources to execute her task before a departure time d_i. For simplicity of notations, we also denote $d_i' = a_i + T_c - T_e$ as the latest starting time for task i to be able to be completed in time. Each agent i has an intrinsic task value v_i and a time-varying task value $v_i(t)$ once she is allocated m_i units of resources from the time t for T_e consecutive time slots. We note $v_i = v_i(a_i)$ as $v_i(a_i) = v_i \times f_i(a_i)$ and $f_i(a_i) = 1$. As discussed in the previous section, the agent i's time-varying value function can be expressed as

$$v_i(t) = \begin{cases} v_i \times f_i(t), & t \in [a_i, d_i'], \\ 0, & \text{otherwise,} \end{cases} \quad (2)$$

where $f_i(t)$ is a time discounting function defined in (1). We note that the arrival time a_i is critical for the edge to make the correct decision. For example, if a self-driving car uploads a task with an incorrect timestamp, it may receive a false driving command, which endangers the safety. Thus, once an agent $i \in \mathbb{N}$ enters the system, the information of arrival time a_i and the resource demand m_i are truthfully revealed. The agent submits a declared intrinsic value (bid) $\hat{v}_i$, which may not be necessarily equal to her true intrinsic value v_i, to a trusted auctioneer (the edge server). We call the true value v_i of agent i as her *type* as in mechanism design, and use vector $\hat{\boldsymbol{v}} = (\hat{v}_1, \hat{v}_2, \cdots, \hat{v}_N)$ to denote the declared types (*i.e.*, the bidding profile) of all agents.

The procedure of online auction mechanism for edge resource allocation is described as follows. We denote $\mathbb{N}_a$ as the set of active agents, who is able to complete its task if starting at the current time slot t, *i.e.*, we have $i \in \mathbb{N}_a$ if $a_i \leq t \leq d_i'$. At each time slot $t \in \mathbb{T}$, the auctioneer first calculates the bid $\hat{v}_i(t)$ for each active agent $i \in \mathbb{N}_a$, by replacing her declared type $\hat{v}_i$ with the true intrinsic value v_i in (2). Given the bidding profile of the active agents $\mathbb{N}_a$ at time t: $\hat{\boldsymbol{v}}(t) = (\hat{v}_1(t), \hat{v}_2(t), \cdots, \hat{v}_{|\mathbb{N}_a|}(t))$, the auctioneer then allocates the total W units of resources, including the idle resources and those in use by existing tasks, to the active agents. We note that to further improve the utilization of resources, the newly arrived agents with high bids could interrupt some ongoing tasks with low bids. The agent i is called a winning agent if she is allocated m_i units of resources for T_e continuous time slots without an interruption before the deadline d_i; otherwise she is called a losing agent. We use $x_i(\hat{\boldsymbol{v}}) = 1$ to denote that the agent i is a winner when the declare value profile is $\hat{\boldsymbol{v}}$; otherwise $x_i(\hat{\boldsymbol{v}}) = 0$. Finally, according to the declared value profile $\hat{\boldsymbol{v}}$ of agents, the auctioneer determines the payment $p_i(\hat{\boldsymbol{v}})$ for each agent i at her departure time d_i. The payments of the losing agents are set to zeros. We use vector $\boldsymbol{x}(\hat{\boldsymbol{v}}) = (x_1(\hat{\boldsymbol{v}}), x_2(\hat{\boldsymbol{v}}), \cdots, x_N(\hat{\boldsymbol{v}}))$ and $\boldsymbol{p}(\hat{\boldsymbol{v}}) = (p_1(\hat{\boldsymbol{v}}), p_2(\hat{\boldsymbol{v}}), \cdots, p_N(\hat{\boldsymbol{v}}))$ to represent the allocation rule and payment rule in an online auction, respectively.

The *utility* u_i of each agent $i \in \mathbb{N}$ is defined as the difference between her value on the allocated resources and the payment:

$$u_i(\hat{\boldsymbol{v}}) = \begin{cases} v_i \times f_i(t_i(\hat{\boldsymbol{v}})) - p_i(\hat{\boldsymbol{v}}), & i \in \mathbb{W}, \\ 0, & \text{otherwise,} \end{cases} \quad (3)$$

where $\mathbb{W}$ is the set of winning agents, and $t_i(\hat{\boldsymbol{v}})$ is the starting time of the winner $i \in \mathbb{W}$ to execute her task when the declared type profile is $\hat{\boldsymbol{v}}$.

As we have shown at the beginning of this section, minimizing the weighted average AoI is equivalent to maximizing the sum of time-varying task values, which is defined as the *social welfare* in the context of auction mechanism as follows.

Definition 1 (Social Welfare)**.** *The social welfare in an online auction mechanism with time discounting values is the sum of winners' values at their corresponding winning time slots,* i.e.,

$$SW = \sum_{i \in \mathbb{W}} v_i \times f_i(t_i(\hat{\boldsymbol{v}})). \quad (4)$$

Other than social welfare, *revenue*, which is defined as the total payment collected from agents, is also a widely used objective in mechanism design. As revenue only reflects the interest of the edge service provider rather than the whole system, we adopt social welfare as the optimization objective in this work, which is beneficial for the long term development of real-time edge service systems. We also evaluate the revenue of the proposed mechanisms in the evaluation results.

In contrast to the optimization goal of the edge service provider, the agents are rational and selfish, and have incentives to maximize their own utilities by strategically reporting their private intrinsic values. To illustrate this strategic behavior in the setting of time discounting task values, we provide a simple example: Suppose agent 1 with $v_1 = 10$ and agent 2 with $v_2 = 8$ send tasks to the edge server at the same time. The edge can only serve one agent and the execution time is $T_e = 1$. We adopt a simple resource allocation rule as the more urgent tasks (tasks with higher values) first, and the payment rule as charging the winners a uniform price 1. Under these rules, the solution would be to execute task 1 at the first time slot and then task 2 at the following time slot. If the values of tasks do not discount over time, then agent 2 has no incentive to misreport her value, because the payment is independent on her bid and her utility is always $8 - 1 = 7$. However, if the values of tasks shrink by half after each time slot, the strategic behaviors may occur. Suppose agent 2 reports her value truthfully, her utility would be $4 - 1 = 3$, and the social welfare is 14. But if agent 2 misreports a value 11, she would be served before agent 1 and obtain a higher utility $8 - 1 = 7$, while the social welfare drops to 13. We also observe from this example that the traditional payment rule to guarantee the strategy-proofness derived from the classical Myerson theorem [24], *i.e.*, the payment is independent on the resource allocation time, no longer holds in the setting of time discounting values. This is because the users can change the resource allocation times, resulting in different utilities in the setting of time-varying task values, by misreporting their values. Therefore, a new proper auction mechanism is necessary for this setting to resist such strategic behaviors and still achieve the optimal social welfare.

C. Solution Concepts

A strong solution concept from mechanism design is *dominant strategy*, where *strategy* is defined as the type reported by a user.

Definition 2 (Dominant Strategy [25])**.** *A strategy $\hat{v}_i$ is agent i's dominant strategy, if for any strategy $\hat{v}'_i \neq \hat{v}_i$ and any other agents' strategy profile $\hat{\boldsymbol{v}}_{-i}$, we have*

$$u_i(\hat{v}_i, \hat{\boldsymbol{v}}_{-i}) \geq u_i(\hat{v}'_i, \hat{\boldsymbol{v}}_{-i}).$$

Intuitively, a dominant strategy of an agent is a strategy that maximizes her utility, regardless of what strategy profile the other agents choose.

The concept of dominant strategy is the basis of *incentive-compatible* mechanism, in which truthfully revealing private information is a dominant strategy for every agent. An accompanying concept is *individual-rationality*, which means that every agent participating in the auction expects to gain no less utility than staying outside. We now can introduce the definition of a *strategy-proof mechanism*.

Definition 3 (Strategy-Proofness [26])**.** *A mechanism is strategy-proof when it satisfies both incentive-compatibility and individual-rationality.*

The objective of this work is to design a strategy-proof online auction mechanism in the setting of time discounting task values.

III. Characterizing Strategy-Proofness

In this section, we present a characterization theorem for strategy-proof online auction mechanisms with time discounting values. This can be considered as a generalization of the well-known Myerson theorem [24]. Specifically, we claim that the necessary and sufficient condition for a payment rule that truthfully implement an allocation rule in the setting of time discounting values is that the function $F(\hat{\boldsymbol{v}}) = f(t(\hat{\boldsymbol{v}})) \times x(\hat{\boldsymbol{v}})$ must satisfy a monotonicity criterion. We first give the definition of this monotone criterion.

Definition 4 (Monotonicity)**.** *The function $F_i(\hat{\boldsymbol{v}}) = f_i(t_i(\hat{\boldsymbol{v}})) \times x_i(\hat{\boldsymbol{v}})$ is monotone, if for any two types of $\hat{v}_i$ and $\hat{v}'_i$ with $\hat{v}_i > \hat{v}'_i$ and the reported types of the other agents $\hat{\boldsymbol{v}}_{-i}$, we have $F_i(\hat{v}_i, \hat{\boldsymbol{v}}_{-i}) \geq F_i(\hat{v}'_i, \hat{\boldsymbol{v}}_{-i})$.*

We take a closer look at this monotone condition $F_i(\hat{v}_i, \hat{\boldsymbol{v}}_{-i}) \geq F_i(\hat{v}'_i, \hat{\boldsymbol{v}}_{-i})$, which could be realized in two detailed cases. One is that the allocation result $x_i(\cdot)$ changes from $x_i(\hat{v}'_i, \hat{\boldsymbol{v}}_{-i}) = 0$ to $x_i(\hat{v}_i, \hat{\boldsymbol{v}}_{-i}) = 1$. The other case is that the allocation result $x_i(\cdot)$ remains the same, i.e., $x_i(\hat{v}_i, \hat{\boldsymbol{v}}_{-i}) = x_i(\hat{v}'_i, \hat{\boldsymbol{v}}_{-i}) = 1$[4] and $f_i(t_i(\hat{v}_i, \hat{\boldsymbol{v}}_{-i})) \geq f_i(t_i(\hat{v}'_i, \hat{\boldsymbol{v}}_{-i}))$, which further implies $t_i(\hat{v}_i, \hat{\boldsymbol{v}}_{-i}) \leq t_i(\hat{v}'_i, \hat{\boldsymbol{v}}_{-i})$ under the assumption of non-increasing function $f_i(t)$. The first case is consistent with the monotonicity of allocation rule in the classical Myerson Theorem, meaning that the bidder with a higher value is more likely to win the auction. The second case comes from the new feature of online mechanism, which further requires the agent with a higher value to be allocated at an earlier time slot. The intuition behind this monotone condition in online setting is that the winning user could be allocated resources at an earlier time slot if she increases her declared type.

We now present our main result: the necessary and sufficient condition for the existence of strategy-proof online auction mechanisms with time discounting values.

Theorem 1. *There exists a payment rule $\boldsymbol{p}(\hat{\boldsymbol{v}})$ such that the online auction mechanism $(\boldsymbol{x}(\hat{\boldsymbol{v}}), \boldsymbol{p}(\hat{\boldsymbol{v}}))$ in the setting of time discounting values is strategy-proof if and only if the function $F_i(\hat{\boldsymbol{v}}) = f_i(t_i(\hat{\boldsymbol{v}})) \times x_i(\hat{\boldsymbol{v}})$ is monotone for each agent $i \in \mathbb{N}$.*

We complete the proof by analyzing the following two lemmas.

Lemma 1. *If the function $F_i(\hat{\boldsymbol{v}}) = f_i(t_i(\hat{\boldsymbol{v}})) \times x_i(\hat{\boldsymbol{v}})$ is monotone for each agent, the online auction mechanism associated with a carefully designed payment rule $\boldsymbol{p}(v)$ is strategy-proof.*

Proof. We set the payment rule as

$$p_i(\hat{\boldsymbol{v}}) = \sum_{k=1}^{K} v_i^k \times \Delta_i^F(v_i^k), \tag{5}$$

where the sequence $v_i^1, v_i^2, \cdots, v_i^K$ is a list of K values, which are the breakpoints of function $F_i(\hat{\boldsymbol{v}})$ when the value increases from 0 to the true value v_i. In general, we assume $v_i^{k_1} \leq v_i^{k_2}$ for $k_1 \leq k_2$, $v_i^0 = 0$ and $v_i^K \leq v_i$. The function $\Delta_i^F(v_i^k)$ represents the jump of $F_i(\hat{\boldsymbol{v}})$ at the breakpoint $(v_i^k, \hat{\boldsymbol{v}}_{-i})$[5], *i.e.*,

$$\Delta_i^F(v_i^k) = F_i(v_i^k, \hat{\boldsymbol{v}}_{-i}) - F_i(v_i^{k-1}, \hat{\boldsymbol{v}}_{-i}).$$

The intuition behind the payment rule in (5) is that, with the increase of v_i, the agent is allocated at a "better" (*i.e.*, earlier) time slot, so the auctioneer charges the agent for this incremental part. The breakpoint value v_i^k in (5) means the critical price of being allocated at the better time slot, and $\Delta_i^F(v_i^k)$ measures "how better the new time slot is", *i.e.*, the (normalized) value difference between the two allocations for agent i.

[4]Another case of $x_i(\hat{v}_i, \hat{\boldsymbol{v}}_{-i}) = x_i(\hat{v}'_i, \hat{\boldsymbol{v}}_{-i}) = 0$ is trivial to analyze, and we omit it here.

[5]We omit the situation with ties for notation simplicity, *i.e.*, we consider $F_i(v_i^k, \hat{\boldsymbol{v}}_{-i}) = F_i(v_i^k + \epsilon, \hat{\boldsymbol{v}}_{-i})$, for a small positive constant ϵ.

With the payment rule in (5), we can express the utility $u_i(\hat{\boldsymbol{v}})$ of agent $i \in \mathbb{N}$ as:

$$\begin{aligned} u_i(\hat{\boldsymbol{v}}) &= v_i \times f_i(t_i(\hat{\boldsymbol{v}})) \times x(\hat{\boldsymbol{v}}) - \sum_{k=1}^{K} v_i^k \times \Delta_i^F(v_i^k) \\ &= v_i \times F_i(\hat{\boldsymbol{v}}) - \sum_{k=1}^{K} v_i^k \times \Delta_i^F(v_i^k) \\ &= \left(v_i^K + v_i - v_i^K\right) F_i(v_i^K, \hat{\boldsymbol{v}}_{-i}) \\ &\quad - \sum_{k=1}^{K} v^k \times \left(F_i\left(v_i^k, \hat{\boldsymbol{v}}_{-i}\right) - F_i(v_i^{k-1}, \hat{\boldsymbol{v}}_{-i})\right) \\ &= \left(v_i - v_i^K\right) F_i(v_i^K, \hat{\boldsymbol{v}}_{-i}) \\ &\quad + \sum_{k=1}^{K} \left(v_i^k - v_i^{k-1}\right) F_i(v_i^{k-1}, \hat{\boldsymbol{v}}_{-i}), \end{aligned} \tag{6}$$

where the third equation is because $F_i(v_i, \hat{\boldsymbol{v}}_{-i}) = F_i(v_i^K, \hat{\boldsymbol{v}}_{-i})$ as v_i^K is the highest breakpoint for resource allocation for agent i. According to the definition of the value sequence, we have $v_i^K \le v_i$ and $v_i^{k-1} \le v_i^k$ for all $1 \le k \le K$. Therefore, the utility $u_i(\hat{\boldsymbol{v}})$ of agent i can not be negative, and the property of *Individual Rationality* is satisfied.

We now show that the monotone function $F_i(\hat{\boldsymbol{v}})$ in combination with the payment rule $p_i(\hat{\boldsymbol{v}})$ in (5) guarantees the property of *Incentive Compatibility*. We prove this by contradiction. If the auction mechanism is not incentive compatible, there exists an agent i, a true type v_i, and a non-truthful reported type $\hat{v}_i$ with $\hat{v}_i \neq v_i$, such that $\hat{u}_i(\hat{v}_i, \hat{\boldsymbol{v}}_{-i}) > u_i(v_i, \hat{\boldsymbol{v}}_{-i})$. That is, the utility of agent i reporting $\hat{v}_i$ is strictly greater than the utility $u_i(v_i, \hat{\boldsymbol{v}}_{-i})$ that she can achieve from being truthful. By (6), we have

$$\begin{aligned} &\left(v_i - v_i^{\widehat{K}}\right) F_i(v_i^{\widehat{K}}, \hat{\boldsymbol{v}}_{-i}) + \sum_{k=1}^{\widehat{K}} \left(v_i^k - v_i^{k-1}\right) F_i(v_i^{k-1}, \hat{\boldsymbol{v}}_{-i}) \\ &> \left(v_i - v_i^K\right) F_i(v_i^K, \hat{\boldsymbol{v}}_{-i}) + \sum_{k=1}^{K} \left(v_i^k - v_i^{k-1}\right) F_i(v_i^{k-1}, \hat{\boldsymbol{v}}_{-i}), \end{aligned} \tag{7}$$

where $\widehat{K}$ is the corresponding maximum index of breakpoints for the misreported type $\hat{v}_i$. It is worth to note that the misreported type $\hat{v}_i$ only impacts the numbers, rather than the values, of breakpoints compared with the true type v_i, because the values of breakpoints are independent on the declared type of agent i.

We complete the analysis by distinguishing two cases:

► If $\hat{v}_i < v_i$, we then have $\widehat{K} \le K$, and thus $v_i^{\widehat{K}} \le v_i^K$. Since the function $F_i(\hat{\boldsymbol{v}})$ is monotone, we can get:

$$\begin{aligned} \textbf{RHS of } (7) &\ge \left(v_i - v_i^K\right) F_i(v_i^{\widehat{K}}, \hat{\boldsymbol{v}}_{-i}) \\ &\quad + \sum_{k=\widehat{K}+1}^{K} \left(v_i^k - v_i^{k-1}\right) F_i(v_i^{k-1}, \hat{\boldsymbol{v}}_{-i}) \\ &\quad + \sum_{k=1}^{\widehat{K}} \left(v_i^k - v_i^{k-1}\right) F_i(v_i^{k-1}, \hat{\boldsymbol{v}}_{-i}) \\ &\ge \left(v_i - v_i^{\widehat{K}}\right) F_i(v_i^{\widehat{K}}, \hat{\boldsymbol{v}}_{-i}) \\ &\quad + \sum_{k=1}^{\widehat{K}} \left(v_i^k - v_i^{k-1}\right) F_i(v_i^{k-1}, \hat{\boldsymbol{v}}_{-i}) \\ &= \textbf{LHS of } (7), \end{aligned}$$

where we reduce $F_i(v_i^{k-1}, \hat{\boldsymbol{v}}_{-i})$ for $\widehat{K}+2 \le k \le K$ to $F_i(v_i^{\widehat{K}}, \hat{\boldsymbol{v}}_{-i})$ in the second item. Thus, we get a contradiction in this case.

► If $\hat{v}_i > v_i$, we then have $\widehat{K} \ge K$, and thus $v_i^{\widehat{K}} \ge v_i^K$. By the monotonicity of the function $F_i(\hat{\boldsymbol{v}})$, we obtain

$$\begin{aligned} &\textbf{LHS of } (7) \\ &\le \left(v_i - v_i^{K+1} + \sum_{k=K+2}^{\widehat{K}} (v_i^{k-1} - v_i^k)\right) F_i(v_i^{\widehat{K}}, \hat{\boldsymbol{v}}_{-i}) \\ &\quad + \sum_{k=1}^{K} \left(v_i^k - v_i^{k-1}\right) F_i\left(v_i^{k-1}, \hat{\boldsymbol{v}}_{-i}\right) \\ &\quad + \left(v_i - v_i^K + v_i^{K+1} - v_i\right) F_i\left(v_i^K, \hat{\boldsymbol{v}}_{-i}\right) \\ &\quad + \sum_{k=K+2}^{\widehat{K}} \left(v_i^k - v_i^{k-1}\right) F_i\left(v_i^{k-1}, \hat{\boldsymbol{v}}_{-i}\right), \end{aligned} \tag{8}$$

Furthermore, since we have $v_i \le v_i^{K+1}$ and $v_i^k \ge v_i^{k-1}$, we can obtain

$$\begin{aligned} (8) &\le \left(v_i - v_i^K\right) F_i\left(v_i^K, \hat{\boldsymbol{v}}_{-i}\right) \\ &\quad + \sum_{k=1}^{K} \left(v_i^k - v_i^{k-1}\right) F_i\left(v_i^{k-1}, \hat{\boldsymbol{v}}_{-i}\right) \\ &= \textbf{RHS of } (7). \end{aligned} \tag{9}$$

Thus, we also get a contradiction in this cases. We completed the proof of the "if" part. □

Conversely, we consider the "only if" part.

Lemma 2. *If the online auction mechanism $(\boldsymbol{x}(\hat{\boldsymbol{v}}), \boldsymbol{p}(\hat{\boldsymbol{v}}))$ is strategy-proof, then we have $F_i(\hat{\boldsymbol{v}}) = f_i(t_i(\hat{\boldsymbol{v}})) \times x_i(\hat{\boldsymbol{v}})$ is monotone for each agent.*

Proof. Consider an agent $i \in \mathbb{N}$ and two type profiles $\boldsymbol{v}$, $\hat{\boldsymbol{v}}$ with $\boldsymbol{v}_{-i} = \hat{\boldsymbol{v}}_{-i}$ and $v_i > \hat{v}_i$. We first consider a scenario where the true type of the agent i is v_i. The strategy-proof mechanism ensures that the utility of agent i when reporting her type truthfully is not less than that when she misreports her type, *i.e.*,

$$f_i(t_i(\boldsymbol{v})) v_i x_i(\boldsymbol{v}) - p_i(\boldsymbol{v}) \ge f_i(t_i(\hat{\boldsymbol{v}})) v_i x_i(\hat{\boldsymbol{v}}) - p_i(\hat{\boldsymbol{v}}). \tag{10}$$

We then consider another scenario where the true type of the agent i is $\hat{v}_i$ and she may cheat by misreporting v_i. Similarly, we have

$$f_i(t_i(\hat{\boldsymbol{v}})) \hat{v}_i x_i(\hat{\boldsymbol{v}}) - p_i(\hat{\boldsymbol{v}}) \ge f_i(t_i(\boldsymbol{v})) \hat{v} x_i(\boldsymbol{v}) - p_i(\boldsymbol{v}). \tag{11}$$

Combining (10) and (11), we can get

$$\begin{aligned} & f_i(t_i(\boldsymbol{v})) v_i x_i(\boldsymbol{v}) - f_i(t_i(\hat{\boldsymbol{v}})) v_i x_i(\hat{\boldsymbol{v}}) \\ \ge\ & p_i(\boldsymbol{v}) - p_i(\hat{\boldsymbol{v}}) \\ \ge\ & f_i(t_i(\boldsymbol{v})) \hat{v}_i x_i(\boldsymbol{v}) - f_i(t_i(\hat{\boldsymbol{v}})) \hat{v}_i x_i(\hat{\boldsymbol{v}}) \\ \Rightarrow\ & f_i(t_i(\boldsymbol{v})) x_i(\boldsymbol{v})(v_i - \hat{v}_i) \ge f_i(t_i(\hat{\boldsymbol{v}})) x(\hat{\boldsymbol{v}})(v_i - \hat{v}_i) \\ \Rightarrow\ & F_i(\boldsymbol{v})(v_i - \hat{v}_i) \ge F_i(\hat{\boldsymbol{v}})(v_i - \hat{v}_i). \end{aligned}$$

Algorithm 1: Resource Allocation Algorithm

```
Input: A vector of declared types $\hat{v}$; arrival time $a_i$,
       latest starting time $d'_i$ and resource demand $m_i$ of
       each task $i$.
Output: A set of winners $\mathbb{W}$.
1  $\mathbb{N}_a \leftarrow \emptyset, \mathbb{W} \leftarrow \emptyset, \mathbb{W}_t \leftarrow \emptyset, \forall t \in \mathbb{T}$;
2  foreach $t \in \mathbb{T}$ do
3      foreach $i \in \mathbb{N}$ do
4          if $a_i \le t \le d'_i$ and $i \notin \mathbb{W}$ then
5              $\mathbb{N}_a \leftarrow \mathbb{N}_a \cup \{i\}$;
6              $\hat{v}_i(t) \leftarrow \hat{v}_i \times f_i(t)$;
7      $\Gamma \leftarrow \{< \hat{v}_i(t), m_i >, i \in \mathbb{N}_a\}$;
8      $\mathbb{W}_t \leftarrow DynamicProgramming(W, \Gamma)$;
9      foreach $i \in \mathbb{W}_t$ do
10         $\mathbb{W} \leftarrow \mathbb{W} \cup \{i\}$;
11 return $\mathbb{W}$.
```

Since $v_i > \hat{v}_i$, we have $F_i(\boldsymbol{v}) \ge F_i(\hat{\boldsymbol{v}})$. Thus, we can conclude that $F_i(\boldsymbol{v})$ is monotone. □

IV. PreDisc for Cases with Unit Edge Execution Time

We now present the detailed design for our proposed mechanism PreDisc, and analyze its economic properties and competitive ratio. We first present the mechanism for the cases of unit edge execution time, *i.e.*, $T_e = 1$, in which we do not need to consider the knotty problem of preemption among tasks. We then extend the mechanism for general cases in the next section.

A. Allocation Rule

We present the procedure of resource allocation rule of PreDisc in Algorithm 1. At each time slot, the active tasks are collected in the set N_a, and their current values and resource demands are collected in the set Γ (Lines 3-7). The allocation problem at each time slot can be formulated as a 0-1 knapsack problem, where the capacity of the knapsack is the resource capacity W and the profit and weight of each item correspond to the current value and the resource demand of the task, respectively. Our goal at each time slot is to select the most cost-efficient active tasks under the resource capacity constraint. Thus, we adopt dynamic programming technique to solve the resource allocation problem at each time slot t to obtain the winner set W_t, and to update the ultimate winner set W (Lines 8-10).

We next show that such a simple allocation rule at each time slot without the knowledge of future tasks, can obtain a constant competitive ratio. The proof is omitted due to space constraint and can be found in [27].

Theorem 2. *The competitive ratio of the resource allocation rule in PreDisc is* 2 *for the cases with unit edge execution time.*

B. Payment Rule

In classical online auction mechanisms [22], [23], to guarantee the strategy-proofness, the payment rule is to set a predefined price for each time slot. However, we have constructed a simple example in the previous section to demonstrate that with such a payment rule, the property of strategy-proofness no longer holds when the value discounts over time. To tackle this obstacle, we calculate the critical price for each single slot, and derive our payment rule based on the extended Myerson Theorem in Section III.

We conduct the following steps to calculate the payment for each winner i in the allocation rule. First, we run the resource allocation algorithm *i.e.*, Algorithm 1 again to compute a new solution without the agent i. During this new allocation process, at each time slot, we can leverage the optimal substructure of dynamic programming, and obtain the minimum bid $\hat{v}_i^{min}(t)$ as the difference between the solutions for total W units of resources and for $W - m_i$ units of resources. The value $\hat{v}_i^{min}(t)$ represents the minimum bid at time slot t that the agent i can win at this time slot. Then, according to the definition of time-varying value function (2), we can get the corresponding critical intrinsic value

$$v_{i,t}^{min} = \frac{\hat{v}_i^{min}(t)}{f_i(t)}$$

which the agent i needs to declare to win at the time slot t. With this critical value for agent i at each time slot, we can greedily select a non-increasing subsequence of critical values over time, which are the breakpoint values as stated in Theorem 1. Intuitively, suppose one breakpoint is $v_{i,t}^{min}$, it means that when the agent i reports an intrinsic value no less than $v_{i,t}^{min}$ at arrives time a_i, she would be selected as a winner no later than the time slot t. We give a procedure in Algorithm 2 to determine the breakpoints from the critical intrinsic values and the corresponding payment for the winning agent i. Following the time slots from the arrival time a_i to the latest starting time d'_i, we set the first breakpoint as the first critical intrinsic price less than the bid of agent. After that, we select the critical intrinsic price as a breakpoint only when it is less than the previously selected breakpoint (Lines 2 to 7). For example, suppose the declared type is 5, and the sequence of critical intrinsic prices is $\{6, 4, 2, 3\}$, we select 4 and then 2 as the breakpoints. We can verify that such a selected set of intrinsic values satisfies the definition of breakpoints. We then sort the selected breakpoints with a non-decreasing order, and calculate the payment using (5) in Theorem 1 (Lines 8-11).

Consider a walkthrough example in Fig. 2, where the total amount of resources W is 5, and the value of each user $i \in \mathbb{N}$ decreases linearly with time through a time-discounting function $f_i(t) = 1 - \frac{1}{3}(t - a_i)$. In Fig. 2, we use solid line to denote the present time interval of each agent. The resource demand and value are also shown beside each agent. In the allocation determination phase, at the first time slot, agents A and C are selected as winners because their total value 9 is larger than that of B. At the second time slot, the value of B becomes $\frac{10}{3}$, and D is chosen due to a higher value 6. At the third time slot, agents B with an updated

Algorithm 2: Payment Calculation Algorithm

```
Input: The declared type $\hat{v}_i$ of agent $i$, a set of critical
       intrinsic values of agent $i$ at each time slot
       $\{v_{i,a_i}^{min}, v_{i,a_i+1}^{min}, \ldots, v_{i,d'_i}^{min}\}$.
Output: The payment $p_i$ of agent $i$.
1  $K \leftarrow 0, p_i \leftarrow 0, V \leftarrow \emptyset$;
2  $CurrentPrice \leftarrow \hat{v}_i$;
3  foreach $t \in \{a_i, a_i+1, \ldots, d'_i\}$ do
4      if $v_{i,t}^{min} \le CurrentPrice$ then
5          $V \leftarrow V \cup \{v_{i,t}^{min}\}$;
6          $CurrentPrice \leftarrow v_{i,t}^{min}$;
7          $K \leftarrow K + 1$;
8  Sort breakpoints in $V$ with a non-decreasing order, and
     re-label them as $v_i^k$ for $k \in \{1, 2, \ldots, K\}$, $v_i^0 \leftarrow 0$;
9  foreach $k \in \{1, 2, \ldots, K\}$ do
10     $\Delta_i^F(v_i^k) \leftarrow F_i(v_i^k, \boldsymbol{v}_{-i}) - F_i(v_i^{k-1}, \boldsymbol{v}_{-i})$;
11     $p_i(\boldsymbol{v}) \leftarrow p_i + v_i^k \times \Delta_i^F(v_i^k)$;
12 return $p_i$.
```

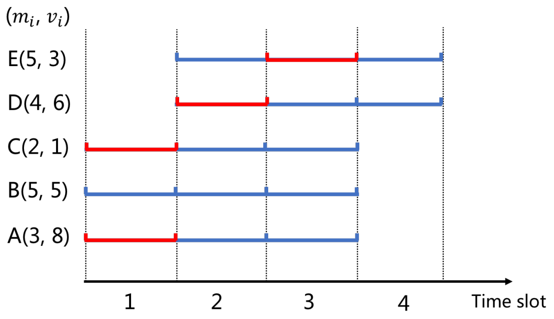


Fig. 2. A walkthrough example for the cases with unit edge execution time.

value $\frac{5}{3}$ and E with an updated value 2 are active, and E is selected as a winner. We denote the winners at each time slot as red in Fig. 2. In the payment calculation phase, for agent A, we remove her and re-run the resource allocation procedure, obtaining the critical intrinsic values for each time slot $v_{A,1}^{min} = 4, v_{A,2}^{min} = 8, v_{A,3}^{min} = 5$. We can greedily get the decreasing subsequence with only a breakpoint $v_A^1 = 4$. Using (5), we can calculate the payment for agent A as $p_A = 4$. Similarly, we have the breakpoint sequence for agent C as $v_C^1 = 0$, for agent D as $v_{D,2}^{min} = \frac{10}{3}, v_{D,3}^{min} = 3, v_{D,4}^{min} = 0$, and for agent E as $v_E^3 = \frac{5}{2}, v_E^4 = 0$. Finally, we can calculate the payment for agents: $p_C = 0, p_D = \frac{19}{9}, p_E = \frac{5}{6}$.

We now show the strategy-proofness of PreDisc based on Theorem 1. The proof is omitted due to space constraint and can be found in [27].

Theorem 3. *The online auction mechanism PreDisc with the above allocation and payment rules is strategy-proof for the cases with unit edge execution time.*

V. PreDisc for General Cases

In this section, we extend PreDisc to the general cases, where the edge execution time T_e can be larger than 1, leading to the situation that tasks may be preempted by other tasks in the execution process.

A. Virtual Bid Generation

When a newly arrived agent has a task value higher than that of some ongoing agents, the auctioneer can choose to preempt the ongoing tasks to make up the task value difference, or to reject the new agents to guarantee the continuity of edge service. Once a task is preempted, it would wait until being selected next time to execute the task from the beginning, and hence the preemption may degrade the resource utilization if the newly arrived agents does not offer a substantially higher bid. With this consideration, the auctioneer raises the bids of ongoing agents, which is denoted by $\mathbb{N}_o$, to give them higher priorities of being allocated resources continuously. At time slot t, each ongoing agent $i \in \mathbb{N}_o$ has been allocated m_i units of resources without an interruption from the time slot $t_i(\hat{\boldsymbol{v}})$. We denote the virtual bid of agent i at time slot t as $b_i(t)$, which can be calculated as

$$b_i(t) = \hat{v}_i(t_i(\hat{\boldsymbol{v}})) \times \alpha^{\varphi_i}, \text{where } \varphi_i = (t - t_i(\hat{\boldsymbol{v}}))/T_e$$

which denotes the percentage of task i's completeness at time t, and $\alpha \ge 1$ is the parameter that the auctioneer can adjust to control the preemption frequency: the setting of $\alpha = 1$ represents the preemption model which interrupts the ongoing tasks once there is a newly arrived task with a higher bid. The auctioneer can give more protection to the ongoing tasks by increasing α. When $\alpha \to \infty$, the auctioneer does not allow preemption, and the tasks can execute for continuous T_e time slots once they are allocated resources. For the active agents that have not been allocated resources, *i.e.*, agents in $\mathbb{N}_a \backslash \mathbb{N}_o$, the auctioneer updates their bids *i.e.*, $b_i(t) = \hat{v}_i(t)$. The auctioneer can generate the virtual bid $b_i(t)$ of the agent $i \in \mathbb{N}$ at time slot $t \in \mathbb{T}$ by distinguishing the following two cases.

$$b_i(t) = \begin{cases} \hat{v}_i(t_i(\hat{\boldsymbol{v}})) \times \alpha^{\varphi_i}, & i \in \mathbb{N}_o, \\ \hat{v}_i(t), & i \in \mathbb{N}_a \backslash \mathbb{N}_o. \end{cases} \tag{12}$$

B. Allocation Rule

The algorithm of resource allocation in the general cases is shown in Algorithm 3. For simplicity, we only present the algorithm for one time slot. Similar to the allocation rule in the simple case, the key idea is to use dynamic programming technique with the virtual bids of agents at each time slot. We first update the current values of tasks as the virtual bids to give the ongoing tasks higher priorities of being allocated (Lines 1-7). After that, we consider the problem of resource allocation as the knapsack problem, and adopt the dynamic programming technique to solve it (Lines 8-9). We update the allocation states for two kinds of agents. For newly winning agents, we update their winning time $t_i(\hat{\boldsymbol{v}})$ as t (Lines 10-11). For preempted agents, we set their winning time to *Null* back (Lines 12-13), then they would wait for the next allocation process. We add the agents, who have executed for T_e consecutive time slots before the departure time, into the ultimate winner set (Lines 15-16). We discard the agents

Algorithm 3: Resource Allocation Algorithm for General Cases (for One Time Slot)

```
Input: A time slot $t \in \mathbb{T}$, a set of active agents $\mathbb{N}_a$, a set of ongoing agents $\mathbb{N}_o$, a vector of reported types $\hat{v}$, a preemption factor $\alpha$, resource demand $m_i$ for each task, and a set of temporary winners $\mathbb{W}_{t-1}$ at time slot $t-1$.
Output: A winner set $\mathbb{W}$ and a temporary winner set $\mathbb{W}_t$ for time slot $t$.
1  foreach $i \in \mathbb{N}_a$ do
2      $\hat{v}_i(t) \leftarrow \hat{v}_i \times f_i(t)$;
3      if $i \in \mathbb{N}_o$ then
4          $\varphi_i \leftarrow (t - t_i(\hat{\boldsymbol{v}}))/T_e$;
5          $b_i(t) \leftarrow \hat{v}_i(t_i(\hat{\boldsymbol{v}})) \times \alpha^{\varphi_i}$;
6      else if $i \in \mathbb{N}_a \backslash \mathbb{N}_o$ then
7          $b_i(t) \leftarrow \hat{v}_i(t)$;
8  $\Gamma \leftarrow \{< b_i(t), m_i >, i \in \mathbb{N}_a\}$;
9  $\mathbb{W}_t \leftarrow DynamicProgramming(W, \Gamma)$;
10 foreach $i \in \mathbb{W}_t \backslash \mathbb{W}_{t-1}$ do
11     $t_i(\hat{\boldsymbol{v}}) \leftarrow t$, $\mathbb{N}_o \leftarrow \mathbb{N}_o \cup \{i\}$;
12 foreach $i \in \mathbb{W}_{t-1} \backslash \mathbb{W}_t$ do
13     $t_i(\hat{\boldsymbol{v}}) \leftarrow Null$, $\mathbb{N}_o \leftarrow \mathbb{N}_o \backslash \{i\}$;
14 foreach $i \in \mathbb{N}_a$ do
15     if $i \in \mathbb{W}_t$ and $t - t_i(\hat{\boldsymbol{v}}) + 1 \geq T_e$ then
16         $\mathbb{W} \leftarrow \mathbb{W} \cup \{i\}$, $\mathbb{N}_a \leftarrow \mathbb{N}_a \backslash \{i\}$, $\mathbb{N}_o \leftarrow \mathbb{N}_o \backslash \{i\}$;
17     else if $i \notin \mathbb{W}_t$ and $t \geq d_i'$ then
18         $\mathbb{N}_a \leftarrow \mathbb{N}_a \backslash \{i\}$;
19 return $\mathbb{W}$, $\mathbb{W}_t$.
```

whose tasks cannot be completed in the remaining time (Lines 17-18).

Theorem 4. *The competitive ratio of our resource allocation rule in PreDisc is* $1 + \frac{\alpha}{1-\alpha^{-\frac{1}{T_e}}}$ *for the general cases compared with the offline optimal solution.*

The proof is omitted due to space constraint and can be found in [27]. Base on the theorem, we can get the optimal preemption factor $\alpha = (1 + \frac{1}{T_e})^{T_e}$ with simple mathematical calculations, while the corresponding competitive ratio is $(T_e + 1)(1 + \frac{1}{T_e}^{T_e})$, which is a small constant related to T_e.

C. Payment Rule

Similar to the payment rule for the simple case, it is necessary to calculate the critical intrinsic price for i to be a winner at different time slot. But the difference is that, in the general cases with $T_e \geq 1$, the critical intrinsic price should guarantee that agent i can win in continuous T_e time slots. Following the procedure in Section IV-B, we can get the minimum virtual bid $b_i^{min}(t)$ of winning at a single time slot t for agent i. Correspondingly, we can calculate $\hat{v}_i^{min}(t)$, the minimum bid at time slot t for agent i to win T_e continuous time slots starting from time t ($t \in [a_i, d_i']$):

$$\hat{v}_i^{min}(t) = \max_{t' \in [t, t+T_e-1]} \frac{b_i^{min}(t')}{\alpha^{\frac{t'-t}{T_e}}}, \tag{13}$$

which is the highest minimum bid (mapping from the minimum virtual bids) for single time slots in the present interval. Then we have the corresponding critical intrinsic value for the agent i at time t:

$$v_{i,t}^{min} = \frac{\hat{v}_i^{min}(t)}{f_i(t)}.$$

Next, we can call Algorithm 2 to calculate the payment of each winning agent.

Finally, we can get the following theoretical guarantee on strategy-proofness, and again omit the proof due to space limit.

Theorem 5. *Our proposed mechanism PreDisc with the above allocation and payment rule is strategy-proof for the general cases.*

VI. Evaluation Results

A. Experimental Settings

We implement our proposed mechanism in C++, and compare it with the existing mechanisms. In the experiments, we set the number of users N as 100, the number of time slots T as 100, the cloud execution time T_c as 10, the edge execution time T_e as 3, the resource capacity W as 10, and the arrival rate γ as 0.1 as default. In particular, we set the intrinsic values of tasks following a uniform distribution over (1, 10), while the numbers of required resources are set as integers following a uniform distribution over [1, 5]. The time discounting function $f_i(t)$ is specified as a linear function $f_i(t) = 1 - \frac{(t-a_i)}{T_c - T_e}$. Each user generates a task at a time slot with probability (arrival rate) γ if she has no active task at the time. We evaluate the changes of both weighted average AoI and revenue with different parameters under different mechanisms. We take the average of 500 runs to get the result.

We compare our mechanism PreDisc with the following benchmark mechanisms:

- **First-Come-First-Served (FCFS)**: In FCFS, at each time slot, the active tasks (including ongoing tasks) are sorted by their arrival time in an increasing order. If their arrival times are the same, the tasks with higher values are selected first. It is worth to note that FCFS is naturally non-preemptive, since tasks with later arrival times are always executed later.
- **Last-Come-First-Served with Preemption (LCFS-p)**: In LCFS-p, at each time slot, the active tasks (including ongoing tasks) are sorted decreasingly by their arrival time. Similarly, if their arrival times are the same, the tasks with higher values are served first. Note that LCFS-p does not protect ongoing tasks from preemption, and hence the tasks are very likely to be preempted by subsequent tasks.
- **Last-Come-First-Served with Non-preemption (LCFS-np)**: LCFS-np is similar to LCFS-p, with the difference that ongoing tasks are protected from

interruption, *i.e.*, once a task is selected to execute, it would be completed without preemption.

- **Offline VCG (VCG-off)**: VCG is a well-known mechanism with optimal social welfare for problems with strategic input. We convert the problem of edge resource allocation into the offline version, and consider VCG mechanism as the ideally optimal baseline. We remark that this mechanism cannot be deployed in real life, as it needs the offline global information.

We conduct experiments on PreDisc with 3 kinds of preemption factors: $\alpha = 1$ (PreDisc-1), $\alpha = 100$ (PreDisc-100) and optimal $\alpha \approx 2.4$ (PreDisc-opt), while PreDisc-opt is also named as PreDisc in some figures as the default setting. To calculate the revenue of FCFS, LCFS-p and LCFS-np, we adopt a simple payment rule which is widely used in practice, *i.e.*, $p_i = \rho \cdot v_i(t_i)$ where $0 < \rho < 1$ is a constant. We set $\rho = 0.5$ in our simulations, meaning that the edge service provider charges half of the values of completed tasks. We remark that such a payment rule is easy to deploy but not truthful, as users can easily cheat at their values to reduce their payments.

B. Numerical Results

The evaluation results on weighted average AoI with different parameters are shown in Fig. 3. We first compare different mechanisms with different arrival rate γ in Fig. 3(a). Overall, we can see that our mechanisms achieve significant reduction on the average AoI than the other mechanisms, and PreDisc-opt obtains the smallest weighted AoI among them. There are two reasons behind the advantage of our mechanisms: First, our mechanisms realize an optimal resource allocation in each time slot, since a dynamic programming rather than a simple greedy algorithm is employed. Second, PreDisc-opt makes a good trade-off between preemption and non-preemption. In addition, FCFS and LCFS-p result in the worst performances, because FCFS tends to select stale tasks with earlier arrival times, while LCFS-p preempts tasks frequently once there are newly arrived tasks. In LCFS-np, fresh tasks with high values are selected and completed without preemption, hence a low AoI is achieved. When γ increases from 0.1 to 0.3, a large amount of tasks are uploaded to the edge, and hence many tasks with high values are not completed. Thus, the weighted AoIs of all mechanisms increase with the arrival rate.

In Fig. 3(b), we compare the above mechanisms with the offline VCG mechanism, the ideally optimal benchmark. The computation complexity of VCG is extremely high, as it needs to enumerate every possible scheduling outcomes. Thus, we reduce the scale of the problem, setting $N = 20$, $T = 10$, $T_c = 5$, $T_e = 3$, $W = 5$, and average the evaluation results over 100 runs. We can observe from Fig. 3(b) that the average AoI of our mechanisms are very close to that of the offline VCG mechanism, which demonstrates the effectiveness of PreDisc. A small difference from Fig. 3(a) is that, the AoIs of some mechanisms decrease with the arrival rate in Fig. 3(b). This is because the resources are relatively sufficient under the scale-reduced setting, and thus the impact of incremental completed tasks is higher than that of incremental uncompleted tasks. We further evaluate the computation complexity (*i.e.*, the program execution time) of FCFS, LCFS-p, LCFS-np, our mechanisms and VCG-off, and show the results in Table I. These results show that our proposed mechanism PreDisc can achieve an approximate optimal weighted average AoI with much lower computation complexity than the optimal solution.

TABLE I
PROGRAM EXECUTION TIME (ms)

FCFS	LCFS-p	LCFS-np	PreDisc-1
0.007	0.005	0.005	0.078
PreDisc-100	PreDisc-opt	VCG-off	
0.080	0.073	137	

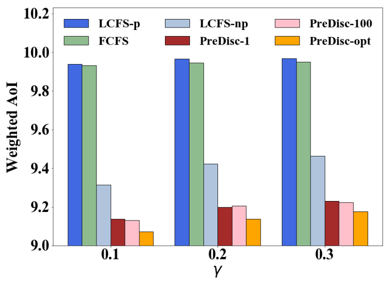
Fig. 3(c) shows the impact of resource capacity W. With a large resource capacity, the edge server can efficiently schedule the tasks to reduce the average AoI, leading to the decrease of AoI from all mechanisms. When $W \geq 60$, nearly all tasks are completed in time in all mechanisms, and thus an identical low AoI is realized. When $W \leq 40$, the resource is limited and PreDisc has a much better resource utilization and then a lower weighted AoI than the other mechanisms.

The impact of preemption factor α on weighted AoI is depicted in Fig. 3(d). We can see that when the preemption factor is close to the optimal α, which is approximately 2.4 under our default settings, the weight AoI indeed realizes a better performance. This performance result demonstrates the optimality of preemption parameter selection in our theoretical analysis of PreDisc.

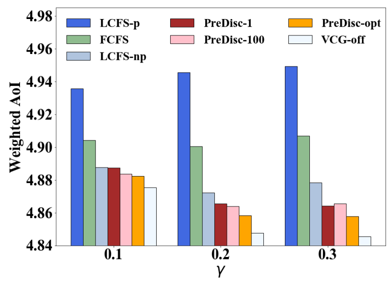
We report the evaluation results under different cloud execution time T_c and edge execution time T_e in Fig. 3(e) and Fig. 3(f), respectively. We remark that T_c is the largest AoI because every task can get a response from the cloud after T_c time slots. A large T_c enables a flexible scheduling for emergency tasks sent to edge, and thus reduces the weighted AoI for these tasks. However, the weighted AoI of the tasks sent to cloud, which is the majority of all tasks, has a significant increase, due to a large T_c. Thus, the overall AoI from tasks sent to edge and cloud increases with T_c. A large T_e implies that tasks would have to wait a longer time to complete. Therefore, the weighted AoI would be higher with the increase of T_e.

We further investigate the average revenue of the edge in different mechanisms in Fig. 4. Fig. 4(a) shows the revenue performance of different mechanisms. We can observe that the revenues of our mechanism outperforms all other mechanisms due to the high utilization of edge resources. In addition, PreDisc-1 achieves the highest revenue in our mechanism, which will be explained later. With the increase of arrival rate γ, the revenues of our mechanisms increase, because more tasks result in a stiffer competition, and hence a higher critical price for winners.

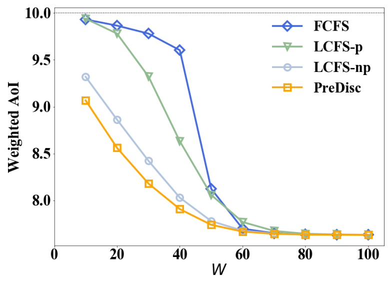
We show the comparison results on average revenue with offline VCG in Fig. 4(b) under the setting of reduced problem scale. Offline VCG achieves the highest revenue, but the gap between our mechanisms and VCG-off is small. Given the extremely large computation complexity and the need of global information of VCG-off mechanism, PreDisc is more practical in deployment with a slight revenue loss. When



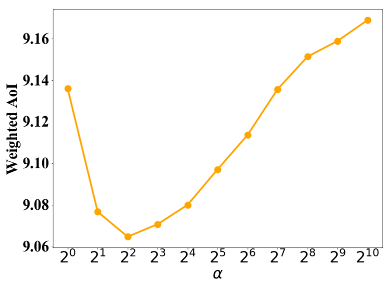
(a) Arrival rate γ



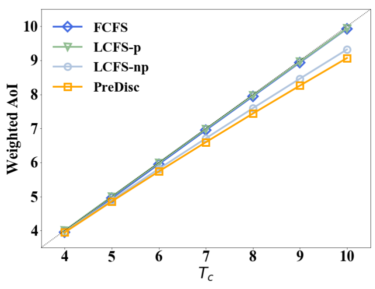
(b) Arrival rate γ with offline VCG



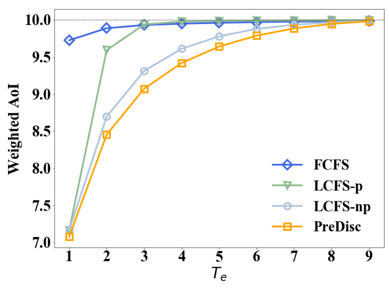
(c) Resource capacity W



(d) Preemption rate α



(e) Cloud execution time T_c



(f) Edge execution time T_e

Fig. 3. The weighted average AoI with different parameters.

$\gamma = 0.1$ or 0.2, the revenue of LCFS-np is slightly higher than our mechanisms, this is because the resources are relatively sufficient under the scale-reduced setting, and thus the critical prices in our mechanisms is low to some extent. We also note that as the payment rule of LCFS-np is not strategy-proof, its present revenue may degrade in real life.

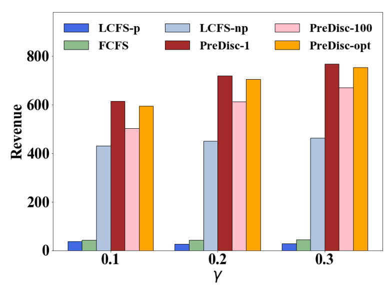
Fig. 4(c) shows the impact of resource supply W on revenue. Naturally, the revenues of FCFS, LCFS-p and LCFS-np increase with a higher W, because the revenues of these mechanisms are proportional to the numbers of completed tasks, which obviously increase with the resource capacity. In contrast with these mechanisms, the revenue of PreDisc would first increase and then decrease into 0 with a large W, because the number of completed tasks increases but the critical prices for resources decrease when the resource supply is more abundant. Thus, we remark that we can improve the revenue of PreDisc by increasing the competition among users.

In Fig. 4(d), we present the impact of preemption factor α on the revenue of PreDisc. We observe that with a higher α, the revenue decreases. This is because a low α leads to frequent preemption, resulting in a high critical price in each time slot. Therefore, we can conclude that both weighted AoI and revenue decrease with the preemption factor α, when it is lower than the optimal value, leading to a trade-off between AoI and revenue when choosing α in this range.

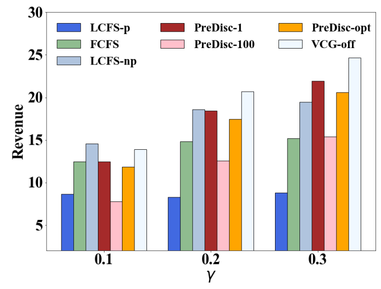
We finally show the revenues of mechanisms with different values of T_c and T_e in Fig. 4(e) and Fig. 4(f), respectively. In Fig. 4(e), the revenue of PreDisc increases with T_c at first and then decreases when T_c is larger than a threshold. This is because PreDisc is able to schedule the tasks flexibly with a large T_c, leading to the number of completed tasks and then the revenue increases. However, if T_c continues to grow, large number of completed tasks implies low critical prices, so the revenues of PreDisc decreases slightly. The revenue of LCFS is always quite small, as the frequent preemption for ongoing tasks causes only a few of tasks to be completed. In LCFS-np, only newly arrived tasks are selected, so the revenue decreases instead because less tasks are produced with a large T_c. In FCFS mechanism, a larger T_c means that tasks with top priorities are more stale, so the revenue decreases with T_c substantially. Fig. 4(f) depicts the impact of edge execution time T_e. When T_e is larger, each task needs resources in more time slots, leading to less tasks to be completed and then lower revenue to obtain. When $T_e = 1$, preemption does not occur, hence LCFS-p and LCFS-np have the same performance. With a higher T_e, the tasks selected by FCFS become fresher, leading to the increase of revenue when $T_e \geq 5$.

VII. Related Work

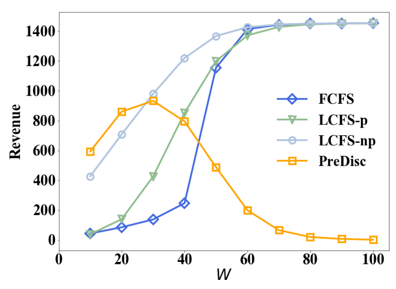
The concept of age of information was first studied in [5], where an optimal updating rate is provided for remote monitor systems to optimize the timeliness. Following this work, much attention has been focused on this metric, typically with the queueing theory technique [5], [28], [29]. This metric was investigated in real-time computing problems in recent years [30]–[32], and different types of update policies and preemption strategies are proposed. However, these studies did not consider the strategic behaviors of user in a timely computing scenario. There are several works that considered the selfish agents in status update systems [33]–[35]. Hao *et al.* [33] investigated the competition of selfish crowdsourcing platforms to reduce their own AoI. They proposed a non-monetary punishment mechnism in a repeated game to enforce their cooperation. The work of [34] introduced the concept of



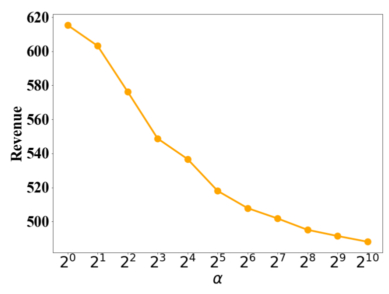
(a) Arrival rate γ



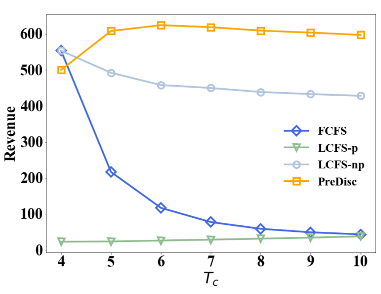
(b) Arrival rate γ with offline VCG



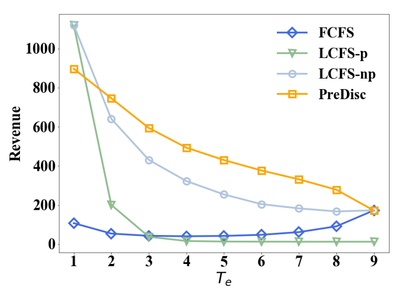
(c) Resource capacity W



(d) Preemption rate α



(e) Cloud execution time T_c



(f) Edge execution time T_e

Fig. 4. The average revenue of the edge with different parameters.

fresh data market. They proposed a new pricing mechanism to maximize the profit of information source and minimize the cost of the destination. These works treat updates as homogeneous ones and only manipulate the update frequency. However, in a real-time edge computing problem, tasks are heterogeneous and users may misreport the information about their tasks. Therefore, the above studies are substantially different from our work.

The topic of online auction was first introduced by Lavi and Nisan [36]. Based on the 2-competitive model of [37] for reusable resource allocation and the proof of competitive ratio, the work in [38] raised the concept of auction with preemption and its application in online spectrum auctions. However, the above classical works only considered constant values during the auction. The authors of [39] considered online auctions with discounting values. However, they imposed constraints on unit resource demand and unit edge execution time, and hence their proposed mechanism does not apply to our more general cases.

From the perspective of edge computing, there are extensive studies that considered the high cost of edge deployment and the resource limitation at the edge server [11], [40], [41]. Some of these works proposed task scheduling algorithms to better utilize the resources [16], [42]–[44]. For example, the authors of [42] proposed an online scalable algorithm, called OnDisc, for the job dispatching and scheduling problem with a constant competitive ratio. In [16], the edge server and the remote cloud server are combined into a heterogeneous cloud. However, all of these studies did not take the pricing mechanism into account, and hence is not practical in the real-life deployment. An online incentive mechanism for the task offloading in mobile edge computing was proposed in [23] based on the primal-dual optimization framework, but they only considered a maximal tolerance delay for each task, rather than the time discounting values of tasks, *i.e.*, the AoI metric. Therefore, their proposed simple threshold-based pricing mechanism cannot be applied in our problem.

VIII. Conclusions

We have proposed a strategy-proof online mechanism PreDisc for the cloud-edge collaborative computing system to reduce the weighted average AoI. A preemption factor is employed to trade off the newly arrived tasks and ongoing tasks. We have proved that PreDisc guarantees a constant competitive ratio compared with the offline optimal solution. Extensive simulations have been conducted and the results demonstrated the effectiveness of PreDisc.

References

[1] W. Zhang, S. Li, L. Liu, Z. Jia, Y. Zhang, and D. Raychaudhuri, "Hetero-edge: Orchestration of real-time vision applications on heterogeneous edge clouds," in *Proceedings of the 38th IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2019, pp. 1270–1278.

[2] R. D. Yates, M. Tavan, Y. Hu, and D. Raychaudhuri, "Timely cloud gaming," in *Proceedings of the 36th IEEE International Conference on Computer Communication (INFOCOM)*. IEEE, 2017, pp. 1–9.

[3] J. Du, Z. Zou, Y. Shi, and D. Zhao, "Zero latency: Real-time synchronization of bim data in virtual reality for collaborative decision-making," *Automation in Construction*, vol. 85, pp. 51–64, 2018.

[4] D. Morrison, P. Corke, and J. Leitner, "Learning robust, real-time, reactive robotic grasping," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 183–201, 2020.

[5] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *Proceedings of the 31st IEEE International Conference on Computer Communication (INFOCOM)*. IEEE, 2012, pp. 2731–2735.

[6] R. D. Yates, "Age of information in a network of preemptive servers," in *Proceedings of the 37th IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS).* IEEE, 2018, pp. 118–123.

[7] J. Zhong, W. Zhang, R. D. Yates, A. Garnaev, and Y. Zhang, "Age-aware scheduling for asynchronous arriving jobs in edge applications," in *Proceedings of the 38th IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS).* IEEE, 2019, pp. 674–679.

[8] J. Zhong, "Age of information for real-time network applications," Ph.D. dissertation, Rutgers University-School of Graduate Studies, 2019.

[9] S. Gopal, S. K. Kaul, and R. Chaturvedi, "Coexistence of age and throughput optimizing networks: A game theoretic approach," in *Proceedings of the 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC).* IEEE, 2019, pp. 1–6.

[10] A. Garnaev, W. Zhang, J. Zhong, and R. D. Yates, "Maintaining information freshness under jamming," in *Proceedings of 38th IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS).* IEEE, 2019, pp. 90–95.

[11] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[12] K. Sasaki, N. Suzuki, S. Makido, and A. Nakao, "Vehicle control system coordinated between cloud and mobile edge computing," in *Proceedings of the 55th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE).* IEEE, 2016, pp. 1122–1127.

[13] N. Alliance, "5g white paper," *Next generation mobile networks, white paper*, vol. 1, 2015.

[14] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[15] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[16] T. Zhao, S. Zhou, X. Guo, and Z. Niu, "Tasks scheduling and resource allocation in heterogeneous cloud for delay-bounded mobile edge computing," in *Proceedings of 2017 IEEE international conference on communications (ICC).* IEEE, 2017, pp. 1–7.

[17] 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," *Computer Networks*, vol. 129, pp. 399–409, 2017.

[18] X. Wang and L. Duan, "Dynamic pricing for controlling age of information," in *Proceedings of 2019 IEEE International Symposium on Information Theory (ISIT).* IEEE, 2019, pp. 962–966.

[19] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *The Journal of Finance*, vol. 16, no. 1, pp. 8–37, 1961.

[20] E. H. Clarke, "Multipart pricing of public goods," *Public choice*, pp. 17–33, 1971.

[21] T. Groves, "Incentives in teams," *Econometrica: Journal of the Econometric Society*, pp. 617–631, 1973.

[22] D. Zhao, X.-Y. Li, and H. Ma, "How to crowdsource tasks truthfully without sacrificing utility: Online incentive mechanisms with budget constraint," in *Proceedings of the 33th IEEE Conference on Computer Communications (INFOCOM).* IEEE, 2014, pp. 1213–1221.

[23] G. Li and J. Cai, "An online incentive mechanism for collaborative task offloading in mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 624–636, 2019.

[24] R. B. Myerson, "Optimal auction design," *Mathematics of Operations Research*, vol. 6, no. 1, pp. 58–73, 1981.

[25] D. Fudenberg and J. Tirole, "Game theory," 1991.

[26] A. Mas-Colell, M. D. Whinston, J. R. Green *et al.*, *Microeconomic theory*. Oxford university press New York, 1995, vol. 1.

[27] "Strategy-proof online mechanisms for weighted aoi minimization in edge computing (technical report)," *https://zhengzhenzhe220.github.io*.

[28] R. D. Yates, "The age of information in networks: Moments, distributions, and sampling," *IEEE Transactions on Information Theory*, https://ieeexplore.ieee.org/abstract/document/9103131, 2020.

[29] A. M. Bedewy, Y. Sun, and N. B. Shroff, "Minimizing the age of information through queues," *IEEE Transactions on Information Theory*, vol. 65, no. 8, pp. 5215–5232, 2019.

[30] A. Arafa, R. D. Yates, and H. V. Poor, "Timely cloud computing: Preemption and waiting," in *Proceedings of the 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton).* IEEE, 2019, pp. 528–535.

[31] V. Kavitha, E. Altman, and I. Saha, "Controlling packet drops to improve freshness of information," *arXiv preprint arXiv:1807.09325*, 2018.

[32] B. Wang, S. Feng, and J. Yang, "When to preempt? age of information minimization under link capacity constraint," *Journal of Communications and Networks*, vol. 21, no. 3, pp. 220–232, 2019.

[33] S. Hao and L. Duan, "Regulating competition in age of information under network externalities," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 4, pp. 697–710, 2020.

[34] M. Zhang, A. Arafa, J. Huang, and H. V. Poor, "How to price fresh data," *arXiv preprint arXiv:1904.06899*, 2019.

[35] Y. Xiao and Y. Sun, "A dynamic jamming game for real-time status updates," in *Proceedings of the 37th IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS).* IEEE, 2018, pp. 354–360.

[36] R. Lavi and N. Nisan, "Online ascending auctions for gradually expiring goods," in *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2005.

[37] M. T. Hajiaghayi, "Online auctions with re-usable goods," in *Proceedings of the 6th ACM Conference on Electronic Commerce*, 2005, pp. 165–174.

[38] L. Deek, X. Zhou, K. Almeroth, and H. Zheng, "To preempt or not: Tackling bid and time-based cheating in online spectrum auctions," in *Proceedings of the 30th IEEE International Conference on Computer Communication (INFOCOM).* IEEE, 2011, pp. 2219–2227.

[39] F. Wu, J. Liu, Z. Zheng, and G. Chen, "A strategy-proof online auction with time discounting values," in *Proceedings of 28th AAAI Conference on Artificial Intelligence (AAAI)*, 2014.

[40] L. Peterson, T. Anderson, S. Katti, N. McKeown, G. Parulkar, J. Rexford, M. Satyanarayanan, O. Sunay, and A. Vahdat, "Democratizing the network edge," *ACM SIGCOMM Computer Communication Review*, vol. 49, no. 2, pp. 31–36, 2019.

[41] Y. Li, K.-H. Kim, C. Vlachou, and J. Xie, "Bridging the data charging gap in the cellular edge," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp. 15–28.

[42] H. Tan, Z. Han, X.-Y. Li, and F. C. Lau, "Online job dispatching and scheduling in edge-clouds," in *Proceedings of the 36th IEEE Conference on Computer Communications (INFOCOM).* IEEE, 2017, pp. 1–9.

[43] S. Jošilo and G. Dán, "Computation offloading scheduling for periodic tasks in mobile edge computing," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 667–680, 2020.

[44] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic task offloading and scheduling for low-latency iot services in multi-access edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 668–682, 2019.