An Optimal Virtual Valuation-Based Combinatorial Auction Mechanism for Time-Varying Resource Allocation in Heterogeneous Cloud Services

Jixian Zhang[1,2], Xuelin Yang[1], Weidong Li[3*], Wenzhong Li[4]

***Abstract*—The resource allocation problem that is posed by cloud services has long been a popular research topic. The existing auction mechanisms focus primarily on maximizing social welfare, but they often result in lower revenue for cloud service providers. The virtual valuation-based combinatorial auction (VVCA) mechanism can increase the revenue that is obtained by service providers while satisfying dominant strategy incentive compatibility (DSIC). In this study, we innovatively apply the VVCA mechanism to address a time-varying resource allocation problem that involves heterogeneous servers (HTs) in cloud services and effectively increase the revenue that is received by cloud service providers. We begin by transforming the HT problem into an integer programming model with time-varying and resource constraint features. Afterward, we provide the theoretical basis for using the VVCA mechanism to solve the aforementioned problem and provide the DSIC proof. On this basis, we design three progressively more effective mechanisms using the VVCA mechanism. (1) We develop a random mechanism $\mathrm{HT\_VVCA^m}$ and prove that it has a logarithmic approximation ratio, thus offering a better lower bound guarantee than the existing approach does. (2) We propose a gradient-based optimization mechanism $\mathrm{HT\_VVCA^*}$ to approximate the optimal revenue. (3) We design an optimal revenue algorithm called HT_VVCANET on the basis of the transformer architecture that is used in deep learning; this algorithm achieves a good balance between execution efficiency and effectiveness. In the experiments, we implement these mechanisms, which significantly increase the revenue that is received by cloud service providers over that yielded by other benchmark mechanisms.**

***Index Terms*—Cloud Service, Time Varying, Heterogeneous Resources, VVCA, Mechanism Design, Deep Learning.**

I. Introduction

CLOUD services, which are supporting technologies for many applications, have received increasing attention; their scope includes areas such as the Internet of Vehicles [1], mobile edge computing [2], federated learning [3], block chains [4] and crowd sensing [5]. Resource allocation is among the key issues in this domain. The goal of resource allocation is to achieve the preset objectives of service providers, such as revenue maximization [6], [7], social welfare maximization [8], [9], and energy consumption minimization [10], [11]. Goals such as revenue maximization and social welfare maximization stem from economic considerations. Because user participation is involved, mechanism design theory [12] is traditionally introduced to address resource allocation problems. The basic principle of this strategy is to determine a winning user allocation solution for a given system and determine the payment that is received by each user. A primary objective of the mechanism design process is to enable resource providers to obtain greater revenue. However, because users are self-interested, they may submit untruthful information (e.g., untruthful bids or requirements) that may harm the revenue of the service providers. Thus, the mechanism must satisfy economic characteristics such as truthfulness and individual rationality. Truthfulness involves encouraging users to submit their true intentions and then maximizing the revenue, social welfare, or utility that is received by resource providers; on this basis, individual rationality ensures the enthusiasm of participating users.

A. Motivation and Challenges

Resource allocation problems exhibit different characteristics depending on their application domains, such as static, dynamic, real-time, and multiresource allocation; virtual machine allocation; and batch allocation. After tracking the resource management and scheduling habits of data centers over a long period, we identify an interesting class of resource allocation problems: time-varying resource allocation in heterogeneous cloud services. This class of problems was initially derived from an analysis of Alibaba Cloud service logs [13]. The service logs revealed that users rent different virtual machine resources at different times to reduce their usage costs. For example, a live-streaming service provider has very low requirements for virtual machines early in the morning, and a fixed rental scheme for its virtual machine resources would result in significant waste. The existing solutions implement allocation mostly on the basis of fixed resources without considering the time-varying requirements of users. Moreover, the virtual machines that are offered by cloud service providers are likely to be deployed on heterogeneous physical machines, which makes the reasonable deployment of these virtual machines important.

1. School of Information Science and Engineering, Yunnan University, Kunming, Yunnan 650504, China
2. Yunnan Key Laboratory of Intelligent Systems and Computing, Yunnan University, Kunming, Yunnan 650504, China
3. School of Mathematics and Statistics, Yunnan University, Kunming, Yunnan 650504, China
4. State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu 210093, China
Email:zhangjixian@ynu.edu.cn, yangxuelin@stu.ynu.edu.cn, weidong@ynu.edu.cn, lwz@nju.edu.cn
Weidong Li is the corresponding author.

0000–0000/00$00.00 © 2021 IEEE

The use of mechanism design theory to solve resource allocation problems has become an effective strategy. An efficient mechanism design can be used to quickly determine the changes that are exhibited by resource supplies, reasonably address requirements, and set prices for resources. Many application scenarios have been combined with auction mechanisms in pricing virtual resources, such as spectrum resources [14], channel resources [15], virtual machine resources [16], fog computing [17], vehicular edge computing [18] and blockchain proofs [19]. In these scenarios, the most pressing concern for resource providers is obtaining greater revenue; this has led to an important mechanism design area: optimal revenue mechanism design. The concept of designing optimal mechanisms for individual items [20] was proposed by Myerson in 1981, but to date, the design of optimal revenue mechanisms for simple multi-item and multiuser scenarios has not been effectively resolved. The affine maximal auction (AMA) mechanism [21] and its subclass, the virtual valuation combinatorial auction (VVCA) mechanism [22], explore the target parameter space by constructing quasilinear user valuation functions, thereby enabling service providers to obtain greater revenue. However, this type of mechanism consumes a significant amount of time in the parameter search process, so research on the AMA mechanism has mostly remained theoretical, with few practical applications. In 2023, Curry et al. [23] first applied deep learning to the AMA mechanism design process, and Duan et al. [24] implemented a contextual AMA mechanism through the transformer framework, thus providing a new theoretical foundation and directions for increasing the practicality of the AMA mechanism.

Applying the VVCA mechanism to solve time-varying resource allocation problems that involve heterogeneous servers (HTs) for cloud services is a very intriguing strategy that offers a new perspective for the field of cloud service resource allocation. However, this approach also faces many challenges. The first issue involves modeling the time-varying resource allocation problem for HTs. Solving this problem requires the resource heterogeneity that is exhibited by the different physical machines of service providers and the time-varying nature of user task resource requirements to be considered. Time variation is reflected in the inconsistent resource requirements of the users in each time slot, and users also have different entry and departure times, which increases the difficulty of resource allocation. The second challenge lies in the design of the mechanism used. A general mechanism for the revenue maximization problem that satisfies the dominant strategy incentive compatibility (DSIC) property has not yet been proposed in existing research. To address this challenge, we leverage machine learning to approximate near-optimal solutions. Previous studies [24] have been based mostly on simple item allocation settings. However, extending item allocation to complex cloud service scenarios is challenging. We must handle various constraints, including multidimensional resources, time windows, time-varying requirements, and discrete allocations. Machine learning alone struggles to capture such complex constraints. Effectively integrating the mechanism design process, machine learning, and these constraints is a significant challenge. Building upon the concept of the AMA mechanism and its deep learning framework, we explore the extension of this approach from combinatorial auctions to more general and complex problem domains. By effectively exploiting the deep learning framework, we expand it from generating continuous allocation solutions to producing discrete solutions that satisfy multiple constraints, thereby achieving stable and superior performance. Various combinations of cloud service and incentive mechanisms are shown in Fig. 1.

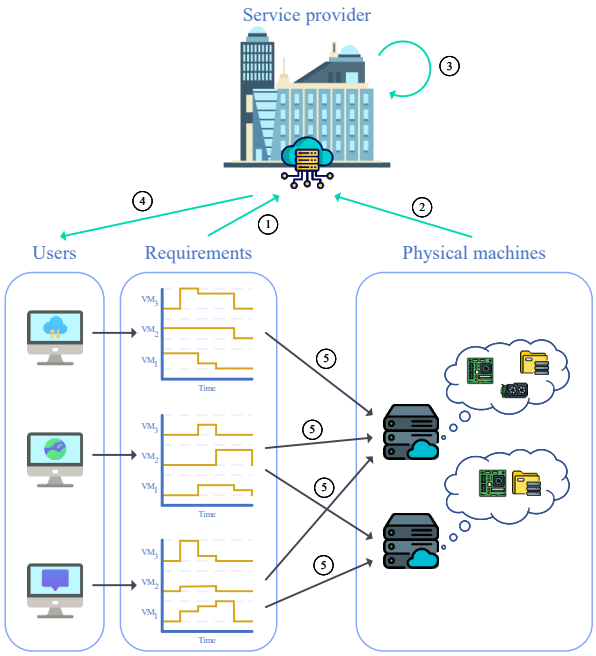

Fig. 1. Combining Heterogeneous Cloud Services with Incentive Mechanisms. The specific steps are as follows. ① Users submit their requirements to the cloud service provider. ② The cloud service provider acquires information about the currently available resources. ③ The cloud service provider calculates the allocation solution and payments based on the mechanism. ④ The cloud service provider notifies the winning users that they must pay. ⑤ Resources are allocated to the winning users.

B. Main Contributions

Based on the above analysis, we model the time-varying resource allocation problem that involves heterogeneous cloud services. The model considers the mapping relationship between heterogeneous physical machines and virtual machines, as well as the real-time and time-varying resource requirements of users. We then convert this problem into an integer programming model with resource and time window constraints. This model is highly compatible with existing heterogeneous resource allocation models [25] and time-varying resource allocation models [26]. It also reflects the typical issues that are encountered by large cloud service providers such as Amazon and Alibaba Cloud in their operations. To solve this problem, we introduce a novel approach that is based on the VVCA mechanism, which is supported by theoretical

analysis and proofs. Our work includes three progressive solutions: HT_VVCA$^{\mathrm{m}}$, HT_VVCA*, and HT_VVCANET. Overall, we summarize the contributions of this study as follows.

- **Novel Model and Theoretical Framework**: We formulate the time-varying resource allocation problem for heterogeneous cloud services as an integer programming model with resource and time window constraints. To maximize the revenue of the resource provider, we adopt the theories of the VVCA to design corresponding mechanisms. The HT_VVCA$^{\mathrm{m}}$ mechanism ensures a logarithmic approximation ratio for revenue optimization, and HT_VVCA* explores optimal revenue through gradient optimization.
- **Efficient Deep Learning-Based Solution**: We propose HT_VVCANET, which is a deep learning-based algorithm that achieves near-optimal revenue with a significantly reduced computation time. HT_VVCANET outperforms traditional mechanisms such as VCG and greedy methods in terms of revenue generation and practicality, especially in handling integer programming problems, which makes it well suited for real-world cloud service operations.

To our knowledge, this is the first study to apply the VVCA mechanism to cloud service resource allocation problems. It not only expands the application scenarios of the VVCA mechanism but also provides principles that can be applied to other resource allocation domains. The remainder of this paper is organized as follows. Related work is presented and discussed in Section II. In Section III, we describe the time-varying resource allocation problem that involves heterogeneous cloud services (the HT problem) and the theoretical basis of the mechanism design. In Section IV, we define the VVCA mechanism and prove that it satisfies the dominant strategy incentive compatibility (DSIC) property. We also design three mechanisms: HT_VVCA$^{\mathrm{m}}$, HT_VVCA*, and HT_VVCANET. In Section V, we evaluate the developed mechanisms through extensive experiments. In Section VI, we summarize our results and present possible directions for future research.

II. Related Work

Many researchers have adopted mechanism design methods for use in cloud service scenarios. To address the problem of dynamic virtual machine provisioning and allocation in a cloud environment, Mashayekhy et al. [27] considered the resource heterogeneity among the different physical machines that are employed by service providers. They designed both optimal and approximate greedy mechanisms to address this challenge. Liu et al. [25] extended this problem to a case with multiple mappings, where a user's virtual machine requirement can be mapped to multiple physical machines. In other research, temporal factors have been considered, either directly or indirectly. Zhang et al. [26] developed an incentive-compatible online cloud auction mechanism that ensures truthfulness through monotonic payment rules and allocation rules that maximize social welfare. Gao et al. [28] simplified the timeline to include multiple auction rounds. A new round starts only after the winning users from the previous round finish their tasks. Wang et al. [29] proposed an online profit maximization-based multiround auction mechanism. This mechanism ensures both the satisfaction of the interests of resource providers and the quality of experience for mobile users. Peng et al. [30] proposed a novel sequential computation offloading mechanism. They used deep reinforcement learning to maximize the received revenue. However, these works did not simultaneously consider the resource heterogeneity of the different physical machines or the time-varying nature of the task resource requirements of users.

The AMA mechanism [21] is a weighted variant of the VCG [31]–[33] mechanism. The AMA mechanism achieves higher revenue than the VCG mechanism does. It ensures that the DSIC property holds by assigning weights to each bidder and assigning boosts to each feasible allocation. Lavi et al. [34] noted that any allocation rule that satisfies certain natural conditions must be "almost" an AMA in a certain technical sense. However, in a combinatorial auction (CA) with n bidders and m items, the traditional AMA mechanism has numerous deterministic allocation candidates, i.e., $(n+1)^m$. As the number of bidders increases, the number of parameters in the AMA increases exponentially. To address this, Likhodedov et al. [22] restricted the AMA to a subclass and proposed a VVCA. In contrast to an AMA, a VVCA has $(n+1)\cdot 2^m$ parameters. The VVCA mechanism can construct a more flexible parameter space for each user, which can be used to improve the user's virtual valuation or to set reservation prices for service providers. Duan et al. [35] used dynamic programming algorithms to calculate the winning allocation solution of the VVCA and combined zeroth-order and first-order techniques to optimize the VVCA parameters. However, the process of searching for the VVCA parameters still has exponential complexity.

Deep learning is a novel and effective method for solving the optimal auction problem that has gradually received increasing attention from researchers. In 2019, Dütting et al. [36] introduced RegretNet. RegretNet is a multilayer neural network that uses "regret" as an indicator to quantify the degree of DSIC violations. RegretNet addresses the problem whereby machine learning (ML) mechanisms struggle to satisfy the properties of truthfulness and individual rationality. In 2022, Ivanov et al. [37] improved the RegretNet framework via multilayer attention mechanisms. Duan et al. [38] proposed the concept of context-integrated auctions and used the transformer architecture to implement auction mechanisms. Because the "regret" cannot reach 0, the aforementioned methods, similar to RegretNet, can ensure only approximate DSIC. In 2023, Curry et al. [23] first applied deep learning to the AMA mechanism design task. Duan et al. [24] implemented the AMA mechanism through a transformer framework. In contrast to previous methods [22], [35] that require the computation of all deterministic allocations, the methods of [23], [24] can reduce the computational time complexity by using a limited menu while preserving the characteristics of the AMA. However, because the menu is generated by neural networks, the optimal allocation solution obtained in [23], [24] is continuous. The

aforementioned approaches were designed based on combinatorial auctions, and we can consider applying the VVCA mechanism to multidimensional resource allocation problems. Furthermore, our approach efficiently obtains a deterministic solution in polynomial time. Appendix E compares our approach with the main closely related methods from five perspectives.

III. HT Resource Allocation Problem and Mechanism Design Preliminaries

A. Parameters and Problem Model

In auction theory, three main roles are considered: sellers, buyers, and the auctioneer. In this study, the cloud service providers are regarded as both sellers and the auctioneer, and the users are considered buyers. We assume that a cloud service provider offer K types of virtual machine (VM) instances, which are represented by the set $\mathcal{K} = \{1, 2, \ldots, K\}$. Each type of virtual machine is composed of R types of resources $\mathcal{R} = \{1, 2, \ldots, R\}$, which can be CPU, memory, or storage resources. The resources that are used by a type of virtual machine $k \in \mathcal{K}$ are defined as $\boldsymbol{q}_k = (q_{k1}, q_{k2}, ...q_{kR})^{\mathrm{T}}$. We assume that the cloud service provider has P physical machines (PMs), which are represented by the set $\mathcal{PM} = \{1, 2, \ldots, P\}$, each with its own capacity limits. $\boldsymbol{c}_j = (c_{j1}, c_{k2}, ..., c_{jR})^{\mathrm{T}}$ is the amount of resources that can be allocated to PM j. Users submit requirements for VM instances to the cloud service provider, which then allocates VM instances to the users, and these VM instances are allocated on the PMs. We divide a period into T discrete time slots. The entire system operates according to time slots $\mathcal{T} = \{1, 2, \ldots, T\}$, where T represents a given constant.

We assume that $\mathcal{N} = \{1, 2, \ldots, N\}$ is the set of system users. The actual requirement of each user $i \in \mathcal{N}$ is represented by $\theta_i = (a_i, d_i, \mathbf{S}_i, e_i, b_i)$, where a_i represents the requirement submission time, d_i represents the requirement execution deadline, and e_i represents the execution time. Clearly, $d_i - a_i + 1 \geq e_i$. During the time slots when a user is executing tasks, the numbers and types of VM requirements for various time slots differ, as indicated by $\mathbf{S}_i$, where

$$\mathbf{S}_i = \begin{bmatrix} s_{i1}^1 \cdots s_{i1}^{e_i} \\ \cdots \cdots \cdots \\ s_{iK}^1 \cdots s_{iK}^{e_i} \end{bmatrix}.$$

s_{ik}^t represents the number of VMs of type k that are requested by user i in the t-th time slot after the task starts executing. t is counted from the beginning of the task execution process. b_i represents the valuation (true bid) of user i for the system to satisfy her/his requirement. We assume that the valuation function of user i is drawn independently from a distribution F_i over the possible valuation functions V_i, and $b_i \in V_i$. The auctioneer knows the distributions $F = (F_1, \ldots, F_n)$ but does not know the realized valuation b_i of bidder i. We assume that the use of users' virtual machines is nonpreemptive, which means that once a virtual machine is allocated, it is not preempted during its execution time. Moreover, d_i and e_i determine the latest start time for the allocation process; if this time is exceeded, the user's requirement will not be satisfied. The users are assumed to be single-minded; this means that user i desires only $\mathbf{S}_i$ and derives a value of b_i if she obtains $\mathbf{S}_i$ or any superset of it at the specified time before its deadline (and zero otherwise).

In accordance with the design principle of the social welfare maximization mechanism, we formulate the HT problem as an integer programming model (IP-HT). In integer programming, we need to calculate two sets of decision variables. $x_{it} \in \{0, 1\}$ and $y_{ijkt} \in \mathbb{Z}^+ \cup \{0\}$. $x_{it} = 1$ indicates that user i starts executing the task in time slot t; otherwise, $x_{it} = 0$. We let $y_{ijkt} \in \mathbb{Z}^+ \cup \{0\}$ represent the fraction of the requirement of user i for virtual machine type k that is satisfied by physical machine j at time slot t. The model is as follows:

$$\underset{\mathbf{X}}{\text{Maximize}} \quad SW(\mathbf{X}) = \sum_{i \in \mathcal{N}} \sum_{t=a_i}^{d_i - e_i + 1} b_i x_{it} \tag{1}$$

$$\sum_{\omega = t - e_i + 1}^{t} s_{ik}^{t+1-\omega} \cdot x_{i\omega} = \sum_{j \in \mathcal{PM}} y_{ijkt}, \quad \forall i \in \mathcal{N}, \forall k \in \mathcal{K}, \forall t \in \mathcal{T} \tag{1a}$$

$$\sum_{i \in \mathcal{N}} \sum_{k \in \mathcal{K}} y_{ijkt} \cdot q_{kr} \leq c_{jr}, \forall r \in \mathcal{R}, \forall j \in \mathcal{PM}, \forall t \in \mathcal{T} \tag{1b}$$

$$\sum_{t=a_i}^{d_i - e_i + 1} x_{it} \leq 1, \forall i \in \mathcal{N} \tag{1c}$$

$$x_{it} \in \{0, 1\}, \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \tag{1d}$$

$$y_{ijkt} \in \mathbb{Z}^+ \cup \{0\}, \forall i \in \mathcal{N}, \forall k \in \mathcal{K}, \forall j \in \mathcal{PM}, \forall t \in \mathcal{T} \tag{1e}$$

where $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iT})^{\mathrm{T}}$ is the set of allocation candidates for each user i; the set of allocation candidates for all users is denoted as $\mathbf{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$. Constraint (1a) ensures that the VM requests by user i at any time slot are satisfied. Constraint (1b) guarantees that the resources that are allocated at any time slot do not exceed the available resources of the PM. Constraint (1c) ensures that resources are allocated at most once during the window period. Because the task execution time for user i is e_i, we must consider only the time slot in which the task begins. Clearly, equation (1) and its constraints can be easily reduced to the multidimensional knapsack problem (MKP), which is strongly NP-hard and cannot be solved in polynomial time unless P = NP. Additionally, our model allows for a single user requirement to be executed across multiple machines.

B. Mechanism Design Concepts and Optimal VCG Mechanism Design

In this section, we first introduce the fundamental concepts of the mechanism design process and several important strategies for identifying properties and then propose a VCG-based mechanism for solving the HT problem (VCG-HT). The VCG mechanism is a generalization of Vickrey's second-price auction [33] that was proposed by Clark [31] and Groves [32]. Usually, a deterministic mechanism is defined as a tuple (A, P), where A represents the allocation function that is used to determine the winning user and P represents the payment function that is used to determine the payment of each user. In our model, b_i of each user in their actual requirement θ_i

is private information. For the user's own benefit, the bid $\hat{b}_i$ that is submitted by the user may not be truthful. We use $\hat{\theta}_i = (a_i, d_i, \mathbf{S}_i, e_i, \hat{b}_i)$ to represent the requirement that is submitted by user i. Except for their bid $\hat{b}_i$, we consider all the information that is submitted by each user to be truthful [28]. Users do not misreport their demand to request fewer VM instances, as doing so would prevent their true needs from being satisfied. Similarly, users do not overstate their demand to request more VM instances because they would risk facing insufficient resources to execute their tasks. For convenience, $\hat{\boldsymbol{\theta}} = (\hat{\theta}_1, \ldots, \hat{\theta}_N)$ denotes the requirements that are submitted by all the users. $\hat{\boldsymbol{\theta}}_{-i} = (\hat{\theta}_1, ., \hat{\theta}_{i-1}, \hat{\theta}_{i+1}, ., \hat{\theta}_N)$ represents all the submitted requirements except for those of user i. $\boldsymbol{\theta} = (\theta_i, \hat{\boldsymbol{\theta}}_{-i})$ denotes the situation in which user i submits a truthful requirement.

The utility for user i when submitting their true requirement is defined as follows:

$$u_i(\boldsymbol{\theta}) = \begin{cases} b_i - p_i, & i \text{ is the winner} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where p_i represents the payment of user i.

The utility for user i when they submit requirement $\hat{\theta}_i$ is defined as follows:

$$u_i(\hat{\boldsymbol{\theta}}) = \begin{cases} b_i - \hat{p}_i, & i \text{ is the winner} \\ 0, & otherwise \end{cases} \tag{3}$$

where $\hat{p}_i$ represents the payment of user i. The difference between (2) and (3) is that when users provide different bids (b_i, $\hat{b}_i$), the payments that are generated by the mechanism are different (p_i, $\hat{p}_i$); thus, the final utility is also different.

The utility (revenue) that is received by the service provider is defined as follows:

$$u_0(\boldsymbol{\theta}) = \sum_{i=1}^{N} p_i \tag{4}$$

A mechanism must satisfy the DSIC property, that is, truthfulness and individual rationality.

Definition 1 (Individual Rationality). *A mechanism that ensures individual rationality must satisfy the following condition: When a user submits their true requirement θ_i, their utility will be greater than or equal to zero; i.e., $u_i(\boldsymbol{\theta}) \geq 0$. In other words, as long as the user participates in the auction and reports truthful bids, the user will never incur a loss.*

Definition 2 (Truthfulness). *If a user submits their true requirement θ_i, they will obtain the maximum utility; i.e., $u_i(\boldsymbol{\theta}) \geq u_i(\hat{\boldsymbol{\theta}})$. This means that submitting a truthful bid is the dominant strategy for each user [39].*

The VCG mechanism is a classic DSIC mechanism [31]–[33]. In this mechanism, every user has an incentive to report their true requirement. The VCG mechanism determines the optimal allocation solution $\mathbf{X}^*$ by maximizing Formula (1), and the payment rule is defined as follows:

$$p_i = SW(\mathbf{X}^*_{-i}) - SW(\mathbf{X}^*) + b_i \tag{5}$$

where $\mathbf{X}^*$ represents the optimal allocation solution for all users and $\mathbf{X}^*_{-i}$ represents the optimal allocation solution for the case in which user i is not participating. Specifically, $\mathbf{X}^*_{-i} = (\boldsymbol{x}^*_1, ., \boldsymbol{x}^*_{i-1}, \boldsymbol{x}^*_{i+1}, ., \boldsymbol{x}^*_N)$.

Algorithm 1: Framework of VCG-HT.

input : $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_N)$, $\mathbf{Q} = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_K)$, $\mathbf{C} = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_P)$
output: $\mathbf{X}^*$, $p_1, \ldots, p_N$

1 $(SW(\mathbf{X}^*), \mathbf{X}^*)$ = Solution of IP-HT($\boldsymbol{\theta}$);
2 **for** $i = 1$ *to* N **do**
3 $\quad \left(SW(\mathbf{X}^*_{-i}), \mathbf{X}^*_{-i}\right)$ = Solution of IP-HT($\boldsymbol{\theta}_{-i}$);
4 $\quad p_i = SW(\mathbf{X}^*_{-i}) - SW(\mathbf{X}^*) + b_i$;
5 **return** $\mathbf{X}^*$, $p_1, \ldots, p_N$;

With the definition of the VCG mechanism, we design the VCG-HT algorithm to solve the HT problem, as shown in Algorithm 1. The VCG-HT mechanism takes the resource capacity vector $\mathbf{C} = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_P)$, the resource vector $\mathbf{Q} = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_K)$ that is required by the VM, and all the user requirements $\boldsymbol{\theta}$ as its inputs. VCG-HT determines the optimal resource allocation solution for users by solving the IP-HT problem given in Formula (1) (Line 1). Afterward, the mechanism determines the payment for each user (Lines 3–4). In doing so, VCG-HT finds the allocation and social welfare obtained without each user's participation (Line 3). The mechanism then charges each user based on the social welfare that is generated with and without the user's participation (Line 4). The mechanism returns two parameters: the optimal allocation solution $\mathbf{X}^*$ and the payment $p_1, \ldots, p_N$.

Although the VCG-HT mechanism can maximize the social welfare of the system and satisfy the DSIC property, it may not generate significant revenue for the cloud service provider. We can regard social welfare as the sum of user utility and service provider utility. The user utility is defined in Formula (2), and the utility that is received by the service provider is defined in Formula (4). We consider a simple example. We suppose that there are only two users and that the available resources are sufficient. User 1 bids 5, and user 2 bids 4. According to VCG-HT, both users 1 and 2 win. User 1 pays 4-9+5=0, and user 2 pays 5-9+4=0. The social welfare of the system is 5+4=9; the utility of the cloud service provider is the sum of user payments, which is 0; and the utility for all users is 5+4=9, which indicates that all the utility in the system is obtained by the users. However, in cloud service auctions, the cloud service provider is more concerned with its own utility and aims to obtain the greatest possible revenue. When the VCG-HT mechanism is used, Formula (5) should be employed to calculate the payment for each user. However, this approach may result in the cloud service provider having almost no profit. Inspired by the original VVCA mechanism [22], we define a new VVCA mechanism for solving the HT problem; this method maximizes the utility of the cloud service provider by exploring the parameter space. Table I summarizes the symbols that are used in this paper.

TABLE I
SYMBOLS.

Symbol	Description
$\mathcal{K}$	Set of virtual machine types $\{1, 2, \ldots, K\}$
$\mathcal{R}$	Set of resource types $\{1, 2, \ldots, R\}$
$\mathcal{PM}$	Set of physical machines $\{1, 2, \ldots, P\}$
$\mathcal{N}$	Set of users $\{1, 2, \ldots, N\}$
$\mathcal{T}$	Set of time slots $\{1, 2, \ldots, T\}$
q_{kr}	Amount of resources of type $r \in \mathcal{R}$ used by a VM instance of type $k \in \mathcal{K}$
c_{jr}	Capacity of PM $j \in \mathcal{PM}$ for resources of type $r \in \mathcal{R}$
a_i	Submission time for user $i \in \mathcal{N}$
d_i	Deadline for user $i \in \mathcal{N}$
e_i	Execution time for user $i \in \mathcal{N}$
s_{ik}^t	Demand quantity of a VM instance of type $k \in \mathcal{K}$ for user $i \in \mathcal{N}$ in time slot $t \in \mathcal{T}$ after the task begins
b_i, $\hat{b}_i$	Value and submitted bid for user $i \in \mathcal{N}$
θ_i, $\hat{\theta}_i$	Actual requests $\theta_i = (a_i, d_i, \mathbf{S}_i, e_i, b_i)$ and submitted requests $\hat{\theta}_i = (a_i, d_i, \mathbf{S_i}, e_i, \hat{b}_i)$ for user $i \in \mathcal{N}$
x_{it}	Whether the request of user $i \in \mathcal{N}$ is satisfied in time slot $t \in \mathcal{T}$
y_{ijkt}	Number of VM instances of type $k \in \mathcal{K}$ that are satisfied on the PM for user $i \in \mathcal{N}$ in time slot $t \in \mathcal{T}$
$\boldsymbol{x}_i$, $\boldsymbol{x}_i^*$	Allocation solution and optimal allocation solution for user $i \in \mathcal{N}$
$\mathbf{X}$, $\mathbf{X}^*$	Allocation solution and optimal allocation solution for all the users
$\mathbf{X}_{-i}^*$	Optimal allocation solution when user $i \in \mathcal{N}$ does not participate

IV. VVCA MECHANISM DESIGN FOR HT RESOURCE ALLOCATION

In this section, we introduce the fundamental concepts of the VVCA mechanism design. Afterward, we propose mechanisms HT_VVCA$^{\mathrm{m}}$ with approximation ratio guarantees and two automated mechanisms: the HT_VVCA* mechanism, which is based on gradient descent, and the HT_VVCANET mechanism, which is built on an attention transformer. Fig. 2 illustrates the relationships among the above mechanisms. In short, the VVCA mechanism is a subclass of the AMA mechanism, and the VCG mechanism is a special case of the VVCA mechanism. HT-based mechanisms are derived by learning from the VVCA family. Specifically, HT_VVCA* refers to HT_VVCANET under the assumption that the VVCA parameters that correspond to identical allocations can be aggregated. Furthermore, HT_VVCA$^{\mathrm{m}}$ is a special case of HT_VVCA* that incorporates reserve prices, and it corresponds to a reserve-price-based VCG mechanism. Therefore, HT_VVCANET can learn any mechanism that HT_VVCA* can learn, and HT_VVCA* can in turn learn any mechanism that HT_VVCA$^{\mathrm{m}}$ can represent. Consequently, the performance of the HT-based mechanisms is at least as good as that of the VCG mechanism in the worst case.

A. VVCA Mechanism

In this section, we define the VVCA mechanism and prove that the VVCA mechanism satisfies the DSIC property. The DSIC proof for the VVCA mechanism is based on its definition [22].

Definition 3 (VVCA). *For each user i with a true bidding function b_i, the allocation function* A *determines the optimal allocation solution $\mathbf{X}^*$ by maximizing Formula (6) with the constraint that is imposed by Formula (1).*

$$\underset{\mathbf{X}}{\text{Maximize}} \quad ASW^{\boldsymbol{\omega},\boldsymbol{\lambda}}(\mathbf{X}) = \sum_{i \in \mathcal{N}} \left(\omega_i \cdot b_i(\mathbf{X}) + \lambda_i(\mathbf{X})\right) \quad (6)$$

In this study, $b_i(\mathbf{X}) = \sum_{t=a_i}^{d_i - e_i + 1} b_i x_{it}$. Moreover, the payment rule p_i for user i is defined as follows:

$$\begin{aligned} p_i^{\boldsymbol{\omega},\boldsymbol{\lambda}}(\mathbf{X}^*, \mathbf{X}_{-i}^*, b_i) = & \frac{1}{\omega_i} \left[ASW^{\boldsymbol{\omega},\boldsymbol{\lambda}}(\mathbf{X}_{-i}^*)\right. \\ & \left. - ASW^{\boldsymbol{\omega},\boldsymbol{\lambda}}(\mathbf{X}^*) + \omega_i \cdot b_i(\mathbf{X}^*)\right] \end{aligned} \quad (7)$$

The VVCA is a subclass of the AMA, and it inherits the DSIC property of the AMA. We prove below that the VVCA satisfies DSIC in our problem model. According to Formula (2) and the VVCA mechanism, given the allocation solution $\mathbf{X}^*$, the utility of user i can be rewritten as follows:

$$u_i^{\boldsymbol{\omega},\boldsymbol{\lambda}}(\mathbf{X}^*, \mathbf{X}_{-i}^*, b_i) = b_i(\mathbf{X}^*) - p_i^{\boldsymbol{\omega},\boldsymbol{\lambda}}(\mathbf{X}^*, \mathbf{X}_{-i}^*, b_i) \quad (8)$$

Theorem 1. ***The VVCA mechanism satisfies individual rationality.***

Proof. Given that the allocation solution is $\mathbf{X}^*$, if user i bids truthfully, his/her payment is

$$\begin{aligned} p_i^{\boldsymbol{\omega},\boldsymbol{\lambda}}(\mathbf{X}^*, \mathbf{X}_{-i}^*, b_i) = & \frac{1}{\omega_i} \left[ASW^{\boldsymbol{\omega},\boldsymbol{\lambda}}(\mathbf{X}_{-i}^*)\right. \\ & \left. - ASW^{\boldsymbol{\omega},\boldsymbol{\lambda}}(\mathbf{X}^*) + \omega_i \cdot b_i(\mathbf{X}^*)\right] \\ \leq & \sum_{t=a_i}^{d_i - e_i + 1} b_i x_{it}^* = b_i(\mathbf{X}^*) \end{aligned} \quad (9)$$

According to the definition of $ASW(\cdot)$, we have $ASW^{\boldsymbol{\omega},\boldsymbol{\lambda}}(\mathbf{X}_{-i}^*) - ASW^{\boldsymbol{\omega},\boldsymbol{\lambda}}(\mathbf{X}^*) \leq 0$. Combined with $\omega_i \geq 0$, Formula (9) holds. Therefore, the VVCA mechanism satisfies individual rationality. □

Theorem 2. ***The VVCA mechanism satisfies truthfulness.***

We prove this theorem in Appendix A.

Although the AMA mechanism has good theoretical properties, it has a long execution time. This is because traversing all the allocation solutions to find a suitable λ function is necessary. The AMA mechanism generates $2^{N \cdot T}$ different task allocation solutions. In contrast, the VVCA mechanism, which is a subclass of the AMA mechanism, assigns a function λ_i to

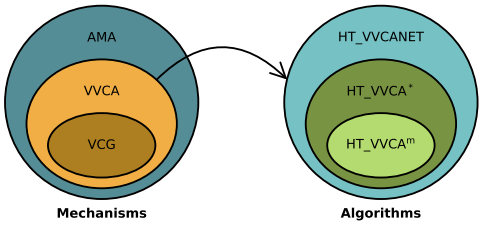

Fig. 2. Relationships among the mechanisms and algorithms.

each user i, thereby resulting in a total of $N \cdot 2^T$ lambda values that must be considered. Therefore, the VVCA mechanism can compute more quickly.

B. VVCA Mechanism Design

In this section, we first propose a randomized mechanism HT_VVCA$^{\mathrm{m}}$ and prove that it has a logarithmic approximation ratio. Afterward, we design an automated mechanism HT_VVCA*. Although HT_VVCA$^{\mathrm{m}}$ requires certain special conditions, we theoretically prove that the revenue that is obtained by the service provider has a good approximation ratio with the OPT, which provides a solid lower bound guarantee for the developed algorithm. The HT_VVCA* mechanism employs gradient optimization to approach the optimal revenue.

HT_VVCA^m: We design a randomized mechanism HT_VVCA$^{\mathrm{m}}$, as shown in Algorithm 2. The design of

Algorithm 2: Framework of HT_VVCA$^{\mathrm{m}}$.

input : $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_N)$
output: $\mathbf{X}^*$, $p_1, \ldots, p_N$
for $i = 1$ *to* N **do**
  if $b_i \geq \sum_{k=1}^{K} \sum_{d=1}^{e_i} s_{ik}^d \cdot l \cdot (1+\varepsilon)^m$ **then**
    Add user i to $\mathbf{X}^*$;
    $p_i = \sum_{k=1}^{K} \sum_{d=1}^{e_i} s_{ik}^d \cdot l \cdot (1+\varepsilon)^m$;
return $\mathbf{X}^*$, $p_1, \ldots, p_N$;

HT_VVCA$^{\mathrm{m}}$ requires the following assumptions.

1) The user valuation functions b_i are all additive.
2) l represents the minimum bid of any user for any virtual machine per time slot.
3) h represents the maximum bid of any user for any virtual machine per time slot.
4) The service provider knows the values of l and h.
5) The system has sufficient resources to satisfy the requirements of the users. (In practice, we can remove users with low cost-effectiveness so that the remaining users have sufficient resources to satisfy this condition.)

Definition 4 (HT_VVCA$^{\mathrm{m}}$)**.** HT_VVCA$^{\mathrm{m}}$ *satisfies the following requirements.*

1) *A virtual user $i = 0$ is constructed, and their bid is defined as $l \cdot (1+\varepsilon)^m$ for a single virtual machine in any time slot.*
2) *For any allocation result that is obtained for virtual user $i = 0$, their bid is defined as the number of allocated virtual machines multiplied by $l \cdot (1+\varepsilon)^m$.*
3) *$\omega_i = 1, \forall i = 0, 1, 2, ..., N$, and $\lambda_i(\mathbf{X}) = 0, \forall i = 1, 2, ..., N$.*
4) *m is randomly selected from $\{0, 1, 2, ..., \lfloor \log_{1+\varepsilon}(h/l) \rfloor\}$.*

User $i = 0$ represents a service provider with a specified reserve price. Therefore, HT_VVCA$^{\mathrm{m}}$ is equivalent to a VCG auction with a reserve price, which means that the cloud service provider has a minimum valuation $l \cdot (1+\varepsilon)^m$ for the usage cost of each virtual machine per time slot. The HT_VVCA$^{\mathrm{m}}$ mechanism can be used to guide the setting of a reasonable reserve price and ensure the expected revenue of the service provider.

Lemma 1. ***When the requirement of user i is met, the user's payment satisfies the following equation:***

$$p_i = \sum_{k=1}^{K} \sum_{d=1}^{e_i} s_{ik}^d \cdot l \cdot (1+\varepsilon)^m. \tag{10}$$

Proof. According to the definition of VVCAs, we have the following:

$$p_i = ASW(\mathbf{X}_{-i}) - ASW(\mathbf{X}) + b_i$$

, where $ASW(\mathbf{X}_{-i})$ represents the optimal social welfare level when user i does not participate. In this case, the requirement that was originally allocated to user i is reallocated to the cloud service provider, who can bid $\sum_{k=1}^{K} \sum_{d=1}^{e_i} s_{ik}^d \cdot l \cdot (1+\varepsilon)^m$ for these virtual machines. Moreover, given that sufficient virtual machines are available, the allocation solutions for the other users remain unchanged. Thus, $ASW(\mathbf{X}_{-i}) - ASW(\mathbf{X}) = \sum_{k=1}^{K} \sum_{d=1}^{e_i} s_{ik}^d \cdot l \cdot (1+\varepsilon)^m - b_i$. Substituting this into the expression for p_i yields the following:

$$\begin{aligned} p_i &= ASW(\mathbf{X}_{-i}) - ASW(\mathbf{X}) + b_i \\ &= \sum_{k=1}^{K} \sum_{d=1}^{e_i} s_{ik}^d \cdot l \cdot (1+\varepsilon)^m. \end{aligned}$$

□

This result implies that if a user's valuation b_i for the required virtual machines does not satisfy

$$b_i \geq \sum_{k=1}^{K} \sum_{d=1}^{e_i} s_{ik}^d \cdot l \cdot (1+\varepsilon)^m,$$

the service provider will refuse to allocate the virtual machines to that user.

Theorem 3. ***The expected revenue approximation ratio that is obtained by the service provider through the*** HT_VVCA$^{\mathrm{m}}$ ***mechanism is***

$$E\left[R_0^m\right] \geq \frac{OPT}{e^{\frac{\sqrt{\ln^2(h/l)+4\ln(h/l)}-\ln(h/l)}{2}} \cdot \left(1+\left(\left\lfloor \frac{2\ln(h/l)}{\sqrt{\ln^2(h/l)+4\ln(h/l)}-\ln(h/l)} \right\rfloor\right)\right)}. \tag{11}$$

We prove this theorem in Appendix B. The proof follows the approach that is presented in [22]. Notably, when $\varepsilon = 1$, the approximation ratio is equivalent to $\frac{1}{2+2\lfloor \log(h/l) \rfloor}$ from [22], but our approximation ratio is better than the previously presented ratios. When $h/l \leq e$, we can achieve a 0.27 approximation. A detailed example of HT_VVCA$^{\mathrm{m}}$ is presented in Appendix C.

HT_VVCA*: In our problem, the VVCA mechanism generates at least $N \cdot 2^T$ parameters, with each user i being assigned a function λ_i. To address the scalability issue, we constrain the users to using the same function λ_i in each time slot and propose a subset of the VVCA, HT_VVCA*. As a result, the number of parameters in HT_VVCA* is reduced to N.

Definition 5 (HT_VVCA*)**.** *For each user i with a true bidding function b_i, the allocation function* A *determines the optimal allocation solution $\mathbf{X}^*$ by maximizing Formula* (12) *with the constraint that is imposed by Formula* (1)*.*

$$\begin{aligned}\underset{\mathbf{X}}{\text{Maximize}} \quad ASW^{\boldsymbol{\omega},\boldsymbol{\lambda}}(\mathbf{X}) &= \sum_{i\in\mathcal{N}} (\omega_i \cdot b_i(\mathbf{X}) + \lambda_i(\mathbf{X})) \\ &= \sum_{i\in\mathcal{N}} \sum_{t=a_i}^{d_i-e_i+1} x_{it} \cdot (\omega_i \cdot b_i + \lambda_i)\end{aligned} \tag{12}$$

where $\lambda_i(\mathbf{X}) = \begin{cases} \lambda_i, & \sum_{t=a_i}^{d_i-e_i+1} x_{it} = 1 \\ 0, & \sum_{t=a_i}^{d_i-e_i+1} x_{it} = 0 \end{cases}$*. In this paper,* $b_i(\mathbf{X}) = \sum_{t=a_i}^{d_i-e_i+1} b_i x_{it}$*. Moreover, the payment rule p_i for user i is defined as follows:*

$$\begin{aligned}p_i^{\boldsymbol{\omega},\boldsymbol{\lambda}}(\mathbf{X}^*, \mathbf{X}^*_{-i}, b_i) = &\frac{1}{\omega_i}\left[ASW^{\boldsymbol{\omega},\boldsymbol{\lambda}}(\mathbf{X}^*_{-i})\right. \\ &\left. - ASW^{\boldsymbol{\omega},\boldsymbol{\lambda}}(\mathbf{X}^*) + \omega_i \cdot b_i(\mathbf{X}^*)\right]\end{aligned} \tag{13}$$

Because the VVCA mechanism satisfies the DSIC property, we only need to maximize the expected revenue. In practice, calculating the expected revenue is challenging; thus, we approximate it using the sample revenue. Therefore, we minimize the negative sample revenue, and the loss function is defined as follows:

$$\ell(\boldsymbol{\omega}, \boldsymbol{\lambda}) = -\frac{1}{|\mathcal{S}|} \sum_{s=1}^{|\mathcal{S}|} \sum_{i=1}^{N} p_i^{(s)}(\boldsymbol{\omega}, \boldsymbol{\lambda})$$

where $\mathcal{S}$ represents the training set and $p_i^{(s)}(\boldsymbol{\omega}, \boldsymbol{\lambda})$ represents the payment of user i in sample s.

The model parameters $\boldsymbol{\omega}$ and $\boldsymbol{\lambda}$ are updated on the basis of gradient descent. The gradient of $\ell(\boldsymbol{\omega}, \boldsymbol{\lambda})$ with respect to $\boldsymbol{\omega}$ for a fixed $\boldsymbol{\lambda}$ is given by

$$\nabla_{\boldsymbol{\omega}} \ell(\boldsymbol{\omega}, \boldsymbol{\lambda}) = -\frac{1}{|\mathcal{S}|} \sum_{s=1}^{|\mathcal{S}|} \sum_{i=1}^{N} \nabla_{\boldsymbol{\omega}} p_i^{(s)}(\boldsymbol{\omega}, \boldsymbol{\lambda})$$

The gradient of $\ell(\boldsymbol{\omega}, \boldsymbol{\lambda})$ with respect to $\boldsymbol{\lambda}$ for a fixed $\boldsymbol{\omega}$ is given by

$$\nabla_{\boldsymbol{\lambda}} \ell(\boldsymbol{\omega}, \boldsymbol{\lambda}) = -\frac{1}{|\mathcal{S}|} \sum_{s=1}^{|\mathcal{S}|} \sum_{i=1}^{N} \nabla_{\boldsymbol{\lambda}} p_i^{(s)}(\boldsymbol{\omega}, \boldsymbol{\lambda})$$

Notably, during training, we assume that the user value distribution is known but that the specific values of the users are unknown. Therefore, we use this distribution to generate sample data as the training set. The revenue of HT_VVCA* is related only to the parameters $\boldsymbol{\omega}$ and $\boldsymbol{\lambda}$. Thus, we aim to maximize the sample revenue and update the parameters $\boldsymbol{\omega}$ and $\boldsymbol{\lambda}$ via gradient descent. The final experiments show that HT_VVCA* is very effective.

C. HT_VVCANET Mechanism Design

Considering the disadvantages of HT_VVCA$^{\mathrm{m}}$ and HT_VVCA* and referring to [23], we propose the HT_VVCANET mechanism, which divides the VVCA solution into two steps: a menu generation algorithm (MGA) and a parameter training process (HT_VVCANET). The MGA uses evolutionary algorithms to generate several good and feasible allocation solutions, thereby significantly reducing the complexity of training; in the parameter training process, an attention transformer is used to capture the relationships of requirement information, and the optimal revenue is ultimately output. Because HT_VVCANET outputs the same VVCA parameters $\boldsymbol{\omega}$ and $\boldsymbol{\lambda}$ for identical inputs, regardless of the associated bids, its DSIC property can be directly derived from the proof for the VVCA.

Architecture of HT_VVCANET: The first step of the HT_VVCANET mechanism is to design the MGA. This algorithm generates an allocation menu $\mathbf{M} \in \{0,1\}^{L\times N\times T}$ as the input for HT_VVCANET, where L represents the size of the allocation menu, N represents the number of users, and T represents the total number of time slots. M_{lit} represents an element (l, i, t) of the 3-dimensional tensor $\mathbf{M}$, and $\mathbf{M}_{l::}$ represents the 2-dimensional l-th slice of the 3-dimensional tensor $\mathbf{M}$. $M_{lit} = 1$ indicates that under allocation option l, the requirement of user i will start to be executed at time slot t. The algorithm ensures that each $\mathbf{M}_{l::}$ is a feasible solution that satisfies the constraints of Formula (1). The MGA is given in Algorithm 3. The algorithm initializes the parameters on Lines 1–7, including the resource availability tensor $\mathbf{R}$, where $\mathbf{R}_{ltj:}$ indicates the amount of resources that are available for allocation on the j-th PM at time slot t under allocation option l. To record the feasible allocation options, we use two matrices: $\mathbf{M}'$ and $\mathbf{M}$. Whereas $\mathbf{M}'$ ensures the diversity of the feasible solutions, $\mathbf{M}$ is used for model training. Because a user's payment depends only on their selection status, $\mathbf{M}$ ensures unique allocation options when focusing on the user selection status. $\mathbf{M}'$ and $\mathbf{M}$ undergo $Iters$ iterations. On line 9, a menu option $\mathbf{V}$ is randomly selected from $\mathbf{M}'$. On lines 10–19, based on the allocation option $\mathbf{V}$, we attempt to allocate resources to a currently unselected user i at time slot t. On lines 20–22, if the remaining resources $\mathbf{R}'$ are sufficient to allocate to user i, we add this allocation option to $\mathbf{M}'$. On lines 23–25, if the allocation option does not overlap with any options in menu $\mathbf{M}$, we add this option to $\mathbf{M}$. In contrast with HT_VVCA*, which searches the exponential solution space for optimality, HT_VVCANET leverages the discarded feasible solutions, thereby enabling polynomial-time efficiency.

We subsequently use an attention transformer to construct HT_VVCANET. As shown in Fig. 3, HT_VVCANET consists of four main parts: a λ-input layer, an ω-input layer, a transformer interaction layer, and an output layer. The allocation menu $\mathbf{M}$ is used as an input, and HT_VVCANET calculates

Algorithm 3: Menu Generation Algorithm.

input : L, S_i, a_i, d_i, e_i, $\mathbf{Q} = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_K)$, $\mathbf{C} = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_P)$
output: $\mathbf{M}$

$\mathbf{M} \leftarrow [0]^{L\times N\times T}$, $\mathbf{M}' \leftarrow [0]^{L\times N\times T}$;
// Initialize the menus
$\mathbf{R} \leftarrow [0]^{L\times T\times P\times R}$;
// Initialize the remaining resources of the PMs
$\boldsymbol{r} \leftarrow [1, 0, \cdots, 0]^{2^N}$;
// Initialize the additional records of the menu options in $\mathbf{M}$
$p_1 \leftarrow 1$, $p_2 \leftarrow 1$;
// Initialize the pointer to the current menu option
for $l = 1$ *to* L **do**
  for $t = 1$ *to* T **do**
    $\mathbf{R}_{lt::} \leftarrow \mathbf{C}$;
for $iter = 1$ *to* $Iters$ **do**
  Generate a random integer l in the range $[1, L]$, and let $\mathbf{V} = \mathbf{M}'_{l::}$;
  for $i = 1$ *to* N **do**
    if $\sum_{t=1}^{T} V_{it} < 1$ **then**
      for $t = a_i$ *to* $d_i - e_i + 1$ **do**
        $V_{it} \leftarrow 1$, $flag \leftarrow 0$;
        $\mathbf{R}' \leftarrow \mathbf{R}_{l:::}$;
        for $d = t$ *to* $t + e_i - 1$ **do**
          On the basis of $\mathbf{V}$ and $\mathbf{R}'_{d::}$, we attempt to sequentially allocate a PM for each VM required by user i and update the remaining resources of the PMs in the variable $\mathbf{R}'$;
          if *There is a VM that cannot be successfully allocated* **then**
            $flag \leftarrow 1$;
            Break;
        if $flag = 0$ **then**
          Let $\mathbf{M}'_{p_1::} = \mathbf{V}$ and $\mathbf{R}_{p_1:::} = \mathbf{R}'$;
          $p_1 \leftarrow (p_1 + 1)\%L$;
          $d \leftarrow \sum_{i=1}^{N}\sum_{t=1}^{T} V_{it} \times 2^i$;
          if $r_d = 0$ **then**
            $\mathbf{M}_{p_2::} \leftarrow \mathbf{V}$;
            $r_d \leftarrow 1$, $p_2 \leftarrow (p_2 + 1)\%L$;
        $V_{it} \leftarrow 0$;
return $\mathbf{M}$;

the user weights $\boldsymbol{\omega}$ and boosts $\boldsymbol{\lambda}$ for the candidate allocation solution. The allocation menu $\mathbf{M}$ serves as the input to the λ-input layer, whose output is processed through the transformer interaction layer to obtain the VVCA parameter boosts $\boldsymbol{\lambda}$ that correspond to the candidate allocation solution. User information is passed through the output layer to generate the VVCA weights $\boldsymbol{\omega}$. Finally, with the weights $\boldsymbol{\omega}$, the boosts $\boldsymbol{\lambda}$, the bid $\boldsymbol{b} = (b_1, b_2, \ldots, b_N) \in \mathbb{R}^N$, and the allocation menu $\mathbf{M}$, the allocation and payment results are calculated according to the VVCA mechanism. The sum of the negative payments is used as the loss, which is backpropagated to update the network parameters.

λ-Input Layer: In the λ-input layer, we use the allocation menu $\mathbf{M} \in \{0,1\}^{L\times N\times T}$ as the input $I^\lambda \in \{0,1\}^{L\times N}$; i.e., $I^\lambda_{li} = \sum_{t=1}^{T} M_{lit}$, where I^λ_{li} is the element (l, i) of matrix I^λ. Two 1×1 convolutions with rectified linear unit (ReLU) activation are subsequently applied to I^λ to obtain the tensor E:

$$E = \text{Conv}_2 \circ \text{ReLU} \circ \text{Conv}_1 \circ I^\lambda \in \mathbb{R}^{L\times d},$$

where d represents the dimensionality of the new latent representation that is produced for each candidate allocation, Conv_1 and Conv_2 represent convolutions, and $\text{ReLU}(z) = \max(z, 0)$. E is used as the input of the transformer interaction layer.

Transformer Interaction Layer: Because the attention mechanism in transformers can capture high-order feature interactions while preserving the crucial property of permutation equivariance, this layer is designed based on the transformer architecture. Through a transformer, we enable L allocation candidates to interact with each other so that the differences between the allocation candidates can be captured. We apply the transformer to E to obtain J:

$$J = \text{transformer - Interaction}\,(E) \in \mathbb{R}^{L\times d_h}$$

, where d_h represents the number of hidden-layer nodes that are contained in the transformer. Next, we use two 1×1 convolutions with ReLU activation to change the number of channels in J:

$$J^{out} = \text{Conv}_4 \circ \text{ReLu} \circ \text{Conv}_3 \circ J \in \mathbb{R}^{L\times d_{out}}.$$

The above structure represents one transformer interaction layer. In practice, we use only one layer. Setting $d_{out} = 1$, we then obtain $J^{out} \in \mathbb{R}^{L\times 1}$ and take J^{out} as an input for the output layer; i.e., $F^\lambda = J^{out}$.

ω-Input Layer: Similar to positional encoding, we use the unique ID of the user as their initial representation $U \in \mathbb{R}^N$, where U_i represents the value of U in dimension i and $U_i = i$. Afterward, we take the user representation U as the input of the ω-input layer, i.e., $I^\omega = \text{Embedding}(U) \in \mathbb{R}^{N\times d_e}$. Next, we use a multilayer perceptron (MLP), which is a fully connected neural network, to process I^ω, which yields the following output:

$$F^\omega = \text{MLP}\,(I^\omega) \in \mathbb{R}^N.$$

Output Layer: We obtain two tensors from the transformer interaction layer and the ω-input layer: $F^\omega \in \mathbb{R}^N$ and $F^\lambda \in \mathbb{R}^{L\times 1}$. Next, we process these two tensors separately. In this work, we impose the constraint that each user weight lies within $(0, 1]$. Given F^ω, we obtain the weight for each user i by

$$\omega_i = \text{Sigmoid}\,(F^\omega_i) \in (0, 1)$$

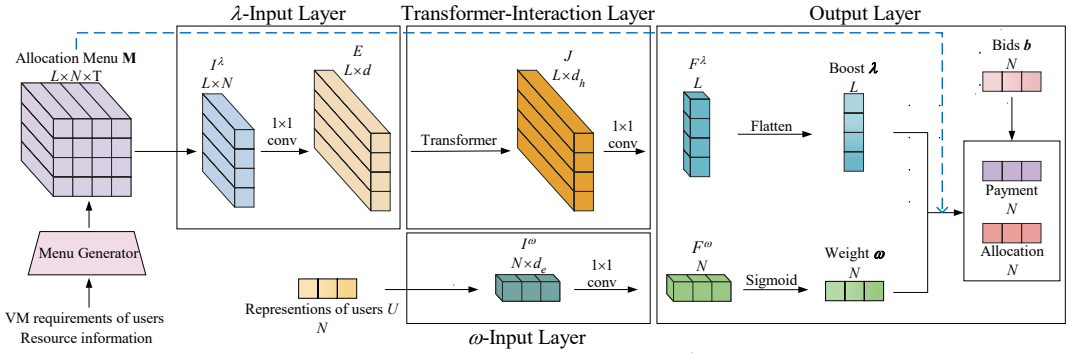

Fig. 3. Network Architecture of HT_VVCANET.

, where F_i^{ω} represents the value of F^{ω} in dimension i and $\text{Sigmoid}(z) = 1/(1+e^{-z}) \in (0,1)$ for all $z \in \mathbb{R}$. With F^{λ}, we can calculate the boost variables $\boldsymbol{\lambda}$. We use the flattening function to unfold F^{λ}, which yields

$$\boldsymbol{\lambda} = \text{Flatten}\left(F^{\lambda}\right) \in \mathbb{R}^{L}.$$

In this way, we obtain $\boldsymbol{\omega}$ and $\boldsymbol{\lambda}$. According to the VVCA mechanism, we can calculate the payment and allocation solutions.

Loss: Our goal is to maximize the expected revenue. As in HT_VVCA*, we approximate the expected revenue with the sample income, and the HT_VVCANET loss is set as follows:

$$\ell(\mathcal{S},\xi) = -\frac{1}{|\mathcal{S}|}\sum_{s=1}^{|\mathcal{S}|}\sum_{i=1}^{N} p_i^{\xi}(\mathbf{M}^{(s)}, \boldsymbol{b}^{(s)}, U)$$

, where $\mathcal{S}$ represents the training set (which contains 4992 samples), ξ contains all the neural network weights in HT_VVCANET, and $p_i^{\xi}(\mathbf{M}^{(s)}, \boldsymbol{b}^{(s)}, U)$ represents the payment of user i.

During training, for any sample s, the affine social welfare level of option l in the generated allocation menu $\mathbf{M}$ is as follows:

$$ASW_l^{(s)}(\mathcal{S},\xi) = \sum_{i\in\mathcal{N}}\left(\omega_i^{\xi} b_i^{(s)} \sum_{t=1}^{T} M_{lit}^{(s)}\right) + \lambda_l^{\xi}, \tag{14}$$

where λ_l^{ξ} represents the boost that corresponds to option l in $\mathbf{M}$ and ω_i^{ξ} represents the weight for user i when the network parameter is ξ.

To ensure that the loss is differentiable during the training process, we use the softmax function in place of the max function [23], and the maximum affine welfare is as follows:

$$\Delta_l^{(s)} = \text{Softmax}\Big(ASW^{(s)}(\mathcal{S},\xi)\Big)_l$$

$$ASW^{*(s)}(\mathcal{S},\xi) = \sum_{l=1}^{L} \Delta_l^{(s)} \cdot ASW_l^{(s)}(\mathcal{S},\xi)$$

where $ASW^{(s)}(\mathcal{S},\xi) = (ASW_1^{(s)}(\mathcal{S},\xi), \ldots, ASW_L^{(s)}(\mathcal{S},\xi)) \in \mathbb{R}^L$ and $\text{Softmax}(\boldsymbol{z})_l = e^{z_l}/(\sum_{m=1}^{L} e^{z_m}) \in (0,1)$ for all $\boldsymbol{z} \in \mathbb{R}^L$. We calculate the payment $p_i^{\xi}(\mathbf{M}^{(s)}, \boldsymbol{b}^{(s)}, U)$ as follows:

$$\begin{aligned} p_i^{\xi}(\mathbf{M}^{(s)}, \boldsymbol{b}^{(s)}, U) =& \frac{1}{\omega_i^{\xi}}\left(ASW_{-i}^{*(s)}(\mathcal{S},\xi) - ASW^{*(s)}(\mathcal{S},\xi)\right) \\ &+ \sum_{l=1}^{L}\left(\Delta_l^{(s)} \cdot b_i^{(s)} \cdot \sum_{t=1}^{T} M_{lit}^{(s)}\right) \end{aligned} \tag{15}$$

where $ASW_{-i,l}^{(s)}(\mathcal{S},\xi) = \sum_{m\neq i, m\in\mathcal{N}}\left(\omega_m^{\xi} b_m^{(s)} \sum_{t=1}^{T} M_{lmt}^{(s)}\right) + \lambda_l^{\xi}$ and $ASW_{-i}^{*(s)}(\mathcal{S},\xi)$ represents the maximum affine welfare for user i, who is not participating in the allocation process, for sample s. $ASW_{-i}^{*(s)}(\mathcal{S},\xi)$ is calculated by the same method as that used for $ASW^{*(s)}$. A discussion of HT_VVCANET is provided in Appendix D.

D. Theoretical Analysis of the VVCA

We present the time complexity, space complexity, and feasibility analyses in Appendix F.

V. Experiments

In this section, we describe the extensive experiments that are conducted to verify the effectiveness of HT_VVCANET and HT_VVCA*. We compare these two methods with three baseline methods: the VCG, greedy and PPO approaches. In all the experiments, the allocation solutions that are obtained by all the algorithms are discrete. We compare and analyze the performance of all the algorithms in terms of service provider revenue, number of winners, social welfare, and execution time. Compared with the baseline methods, the mechanisms that are designed on the basis of affine maximization properties achieve better results.

A. Experimental Setup

1) The data that are used in our experiments are sourced from Alibaba Cloud service logs [40]. We select three tables from the log files: PaiJobTable (job starting information), PaiTaskTable (task starting information), and PaiMachineSpec (PM information). The experimental setup is described in detail in Appendix G. We assume that the valuations are additive. Moreover, we assume that the valuation of user i for any virtual machine

per time slot is $\bar{b}_i$; i.e., $b_i = \bar{b}_i \cdot \sum_{k=1}^{K} \sum_{t=1}^{e_i} s_{ik}^t$. As the valuation distribution is not recorded in the logs, we sample valuation information from a distribution. We consider various types of valuation distributions, including symmetric uniform, asymmetric uniform, and normal distributions. These distributions are described as follows.

(A) Symmetric Uniform Distribution: The valuation $\bar{b}_i$ of each user i is sampled from $U[0, 1]$. This setup has been widely used in previous studies [22], [23], [36].
(B) Asymmetric Uniform Distribution: The valuation $\bar{b}_i$ of each user i is sampled from $U[i-1, i]$. The valuations are asymmetric for users.
(C) Normal Distribution: The valuation $\bar{b}_i$ of each user i follows a normal distribution with a mean of 1 and a variance of 0.1, which is truncated to the range $[0, +\infty]$.

2) On the basis of the classic VCG approach, we impose resource constraints. The implementation of HT_VVCA* follows the design that is outlined in Section IV-B. Both our VCG and HT_VVCA* mechanisms are implemented in CPLEX. We then design a greedy algorithm that computes the optimal allocation solution using a greedy strategy and calculates the payment via binary search [25]. During the allocation procedure, the users are sorted in descending order on the basis of $d_i = \frac{b_i}{\sqrt{\sum_{k=1}^{K} \sum_{t=1}^{e_i} s_{ik}^t}}$. To allocate VMs to PMs, the greedy algorithm is consistent with the allocation method that is described in Algorithm 3. Notably, the greedy algorithm satisfies the DSIC property. In PPO, the action consists of selecting 0 or 1 for each user. The state aggregates all the relevant environmental information. At each step, the environment considers only users whose action is 1, who have not yet been allocated resources, and for whom sufficient resources are available. The payment rule in PPO follows the same scheme as in VCG.
3) Each algorithm is executed five times under the same data distribution but with different data samples when calculating service provider revenue, number of winners, social welfare, and execution time.
4) During the training process of HT_VVCANET, the maximum number of iterations is set to 100, with 4992 samples generated in each iteration. We evaluate 960 samples. The model that produces the highest revenue on the training set is selected, and the revenue that it obtains on the test set is used as the final evaluation result. In the MGA module of HT_VVCANET, the menu size is set to 200, and the number of iterations ($Iters$) is 200. The hidden dimension of the network parameters is set as $d = 32$, where $d_h = 32$ and $d_e = N$. The softmax temperature is set to 500, which is sufficiently large to approximate the max function. The batch size is 64, and the learning rate is set to 1×10^{-3}. Given the induced VVCA parameters, our implementation of the remainder of the AMA mechanism is built upon the implementation of Curry et al. [23].
5) We use an NVIDIA RTX 4090 GPU and an Intel Xeon Platinum 8352V CPU to train and test HT_VVCANET. The training process is described in Appendix H. The analysis of the HT_VVCANET allocation menu is presented in Appendix I. All the experimental code is available at https://github.com/YNU-DMC-yxl597/HT_VVCANET.

B. Experimental Results

1) **Effect of the Valuation Distribution on Revenue:** We evaluate the revenue that is produced by the five mechanisms under various valuation distributions, as shown in Table II. We also perform t tests between HT_VVCANET and the other four algorithms, as shown in Table III. In this experiment, the number of users is fixed at 6 ($N = 6$), the number of PMs is fixed at 3 ($P = 3$), and the experiment is conducted over 4 time slots ($T = 4$). Both HT_VVCA* and HT_VVCANET clearly significantly outperform VCG, thus demonstrating that integrating affine parameters ω_i and λ_i (7) into VCG substantially increases the resulting revenue. The standard deviations of HT_VVCANET are relatively low across all the distributions. This finding indicates that the algorithm yields consistent results across different data samples. Notably, under the asymmetric distribution (B), compared with the other algorithms, HT_VVCANET shows significant differences; this is due to larger variations in ω_i^{ξ} (14) that are caused by differing user valuation distributions. Smaller values of ω_i^{ξ} are more likely to occur. According to the payment formula (15), smaller ω_i^{ξ} lead to larger payments p_i. Overall, the performance of HT_VVCANET is superior.

TABLE II
EFFECTS OF THE VALUATION DISTRIBUTION ON THE MEAN AND STANDARD DEVIATION OF REVENUE. FOR EACH SETTING, THE HIGHEST MEAN REVENUE AMONG ALL METHODS IS SHOWN IN **bold font**.

Method	Symmetric(A)		Asymmetric(B)		Normal(C)	
	mean	std.	mean	std.	mean	std.
VCG	1.45	0.04	8.22	0.30	5.20	0.11
greedy	1.50	0.04	8.78	0.20	4.74	0.06
PPO	1.20	0.04	8.02	1.01	4.48	0.11
HT_VVCA*	2.23	0.06	11.8	2.46	6.08	0.52
HT_AMANET	**2.27**	0.04	**15.4**	0.38	**6.97**	0.08

TABLE III
t AND p OF THE T-TESTS COMPARING HT_VVCANET WITH THE OTHER FOUR ALGORITHMS.

Method	Symmetric(A)		Asymmetric(B)		Normal(C)	
	t	p	t	p	t	p
VCG	60.78	$< 10^{-6}$	107.9	$< 10^{-7}$	20.77	$< 10^{-4}$
greedy	46.02	$< 10^{-5}$	46.56	$< 10^{-5}$	35.99	$< 10^{-5}$
PPO	41.22	$< 10^{-5}$	11.38	$< 10^{-3}$	30.34	$< 10^{-5}$
HT_VVCA*	1.351	0.25	2.645	0.05	3.788	0.02

2) **Effects of the Number of Users:** We compare the performance differences among the four mechanisms under various numbers of users, as shown in Fig. 4. In this experiment, the number of PMs is fixed at 3 ($P = 3$), and the experiment is conducted over 5 time slots ($T = 5$) with the valuation distribution (C). By changing the number of users, where User $\in \{2, 4, 6, 8, 10\}$, we observe the changes that are induced in

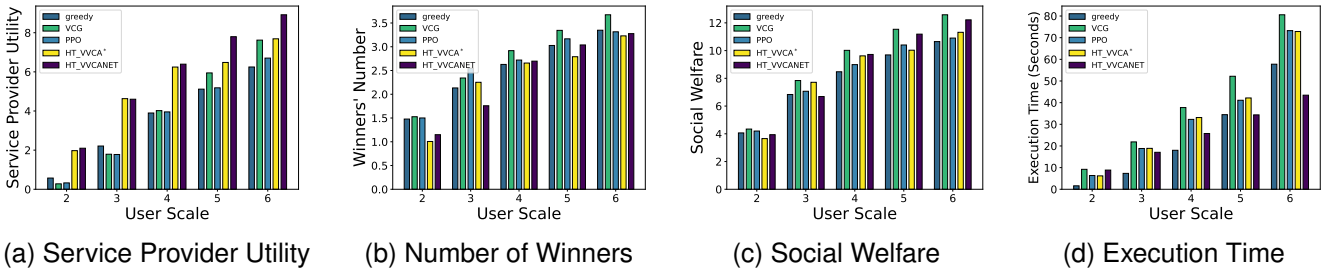

Fig. 4. Effects of the Number of Users.

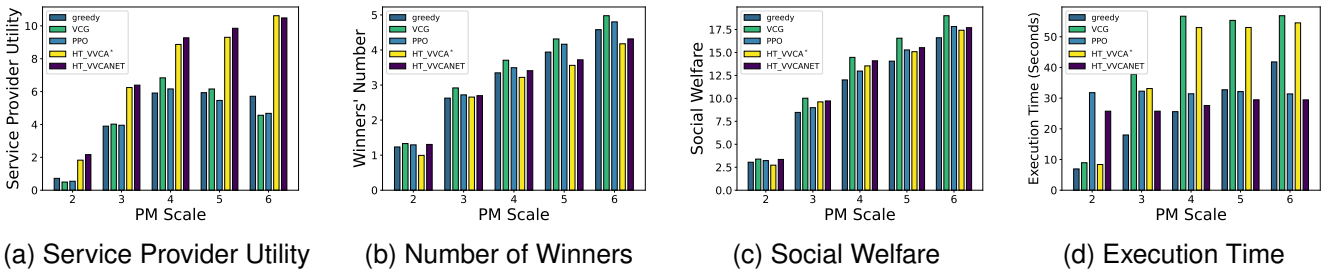

Fig. 5. Effects of the Number of PMs.

terms of service provider revenue, number of winners, social welfare, and execution time across various algorithms. Our objective is to maximize the service provider's revenue; more winners and higher social welfare are not necessarily needed. The service provider revenue changes that are produced by the different algorithms are shown in Fig. 4(a). Compared with the other methods, both HT_VVCANET and HT_VVCA* achieve better results. As the number of users increases, the revenue of HT_VVCA* gradually approaches that of VCG. This is because HT_VVCA* is VCG when $\omega_i = 1$ and $\lambda_i = 0$ (12). Under complex or large-scale data scenarios, HT_VVCA* tends to revert to VCG. In contrast, λ_l^ξ and ω_i^ξ (14) in HT_VVCANET are generated through a transformer network and an MLP network. The transformer network helps us discover differences between the allocation candidates by allowing them to interact in the transformer interaction layer, thereby producing interaction information F^λ. The MLP component performs particularly well for low-dimensional outputs and provides stable and superior results. The changes in the number of winners are shown in Fig. 4(b). As the number of users increases, the number of winners for each algorithm also increases. This is because more high-quality users (with high bids and low resource requirements) participate in the auction. The numbers of winners for HT_VVCANET and HT_VVCA* are significantly lower than those for VCG, which indicates that these two algorithms increase revenue by reducing the number of winners. The social welfare results are shown in Fig. 4(c). VCG achieves the highest social welfare. The social welfare of the other algorithms does not exceed that of VCG, which indicates that the proposed allocation method is effective. HT_VVCANET achieves lower social welfare than VCG does, which suggests that integrating affine parameters ω_i and λ_i (7) into VCG reduces social welfare. Compared with the greedy approach, PPO achieves greater social welfare owing to the strong decision-making capability of reinforcement learning. However, its social welfare is lower than that of HT_VVCANET and HT_VVCA* because PPO operates fully online whereas the latter methods are semi-online. Considering Fig. 4(a), the revenue gap between HT_VVCANET and VCG is larger than the social welfare gap, which makes the tradeoff for revenue worthwhile. Moreover, most social welfare is obtained by the service provider, which aligns with our design goal. The execution times that are required on the test set are shown in Fig. 4(d). The execution time of HT_VVCANET comprises the time that is required for menu generation and the time that is needed to process the test set on the neural network model. CPLEX can rapidly obtain optimal solutions through strategies such as pruning. Consequently, neither HT_VVCA* nor VCG exhibits exponential time growth. Concurrently, HT_VVCANET demonstrates the ability to generate high-quality menus within remarkably short timeframes. Moreover, HT_VVCANET needs to generate the menu only once for repeated training operations, whereas HT_VVCA* requires the CPLEX solver to be called each time the parameters are updated. Therefore, HT_VVCANET has a higher training speed. Overall, the performance of HT_VVCANET is superior.

3) **Effects of the Number of PMs:** We compare the performance differences among the four mechanisms under various numbers of PMs, as shown in Fig. 5. In this experiment, the number of users is fixed at 6 ($N = 6$), and the experiment is conducted over 5 time slots ($T = 5$), with the valuation distribution (C). By changing the number of PMs, where PM $\in \{2, 3, 4, 5, 6\}$, we observe the changes in service provider revenue, number of winners, social welfare, and execution time across the different algorithms. The changes in service provider revenue when different algorithms are used are shown in Fig. 5(a). As the number of PMs increases, the revenue of HT_VVCANET and HT_VVCA* increases. The revenue of the greedy and VCG methods first increases but then decreases. When resources are extremely scarce, no user's requirements can be met, which results in low revenue. When

resources are abundant, the greedy and VCG methods satisfy all the users' requirements, thereby leading to a revenue that is close to 0. HT_VVCANET and HT_VVCA* use the VVCA parameters to intentionally leave some users' needs unmet, thereby achieving high revenue. The changes in the number of winners are shown in Figure 5(b). As the number of PMs increases, the number of winners for each algorithm increases. This is because the more resources that are available, the more users there are whose needs the service provider can meet. The social welfare results are shown in Figure 5(c). VCG achieves the highest social welfare. The execution times on the test set are shown in Figure 5(d). As the number of PMs increases, the execution times stabilize; this could be because the feasible solution space is no longer constrained, and the program can exit the loop early after the PM is allocated. Overall, HT_VVCANET exhibits superior performance.

VI. Conclusions and Future Work

In our view, the most significant innovations of this study are twofold. First, we combined the VVCA mechanism with the time-varying resource allocation problem (HT problem) that involves heterogeneous cloud services. Second, we designed a deep learning network architecture (HT_VVCANET) to obtain deterministic allocation solutions for the HT problem. Through this study, we explored the application domains of integrating the AMA mechanism with deep learning. We then exploited this framework to achieve stable and superior performance in solving complex allocation problems. The experimental results showed that compared with the traditional VCG mechanism, the HT_VVCANET mechanism can significantly increase the revenue of cloud service providers by 56% to 87%; this provides a new approach for applying deep learning to solve constrained problems.

We also discuss various limitations and potential directions for future work. Our current approach is semi-online and requires knowledge of the time-varying heterogeneous demands of all users in advance to plan allocations for a future period. Such a setting still has practical applications. A possible extension is to divide future time into segments and obtain user information in each segment to make appropriate allocations. In future work, we plan to consider fully online scenarios. Another limitation is that sufficient samples are required to train the model. However, this is a common requirement that is shared by most existing models [22]–[24], [35], [36] because current deep learning methods still rely on learning from historical data. We believe that more advanced approaches will emerge in the future to address this limitation.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Nos. 62062065 and 12071417), the Natural Science Foundation of Yunnan Province of China (Nos. 202501AS070076 and 202401AT070471), the Open Project Program of Yunnan Key Laboratory of Intelligent Systems and Computing (No. ISC24Z01) and the Program for Excellent Young Talents, Yunnan, China. We would like to thank the anonymous reviewers for their helpful remarks.

References

[1] S. Lu and W. Shi, "Vehicle computing: Vision and challenges," *Journal of Information and Intelligence*, vol. 1, no. 1, pp. 23–35, 2023.
[2] 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, 2016.
[3] Y. Jiao, P. Wang, D. Niyato, B. Lin, and D. I. Kim, "Toward an automated auction framework for wireless federated learning services market," *IEEE Transactions on Mobile Computing*, vol. 20, no. 10, pp. 3034–3048, 2020.
[4] K. Zhang, X. Wang, B. Yi, M. Huang, L. Qiu, E. Lv, and J. Guo, "A reliable distributed-cloud storage based on permissioned blockchain," *IEEE Transactions on Services Computing*, vol. 18, no. 3, pp. 1216–1231, 2025.
[5] J. Zhang, Y. Zhang, H. Wu, and W. Li, "An ordered submodularity-based budget-feasible mechanism for opportunistic mobile crowdsensing task allocation and pricing," *IEEE Transactions on Mobile Computing*, vol. 23, no. 2, pp. 1278–1294, 2024.
[6] A. Mampage, S. Karunasekera, and R. Buyya, "Deep reinforcement learning for scheduling applications in serverless and serverful hybrid computing environments," *IEEE Transactions on Services Computing*, vol. 18, no. 2, pp. 718–728, 2025.
[7] C.-C. Hu and J.-Y. Zeng, "A service-oriented optimization framework for edge caching with revenue maximization and qos guarantees," *IEEE Transactions on Services Computing*, vol. 18, no. 5, pp. 2559–2573, 2025.
[8] J. Zhang, X. Yang, N. Xie, X. Zhang, A. V. Vasilakos, and W. Li, "An online auction mechanism for time-varying multidimensional resource allocation in clouds," *Future Generation Computer Systems*, vol. 111, pp. 27–38, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19331127
[9] H. Zhao, Z. Wang, G. Cheng, W. Qian, P. Chen, J. Yin, S. Dustdar, and S. Deng, "Online workload scheduling for social welfare maximization in the computing continuum," *IEEE Transactions on Services Computing*, vol. 18, no. 4, pp. 2267–2280, 2025.
[10] C. H. Liu, Z. Chen, and Y. Zhan, "Energy-efficient distributed mobile crowd sensing: A deep learning approach," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1262–1276, 2019.
[11] M. Teng, X. Li, X. Zhang, Y. Bu, K. Zhu, A. Mahmood, J. Wu, and Q. Z. Sheng, "Integrated resource allocation for sequential task offloading in edge computing," *IEEE Transactions on Services Computing*, vol. 18, no. 4, pp. 2115–2128, 2025.
[12] T. Roughgarden, "Algorithmic game theory," *Communications of the ACM*, vol. 53, no. 7, pp. 78–86, 2010.
[13] Tianchi, "Alibaba cluster data," 2024. [Online]. Available: https://tianchi.aliyun.com/dataset/dataDetail?dataId=6287
[14] X. Zhou and H. Zheng, "Trust: A general framework for truthful double spectrum auctions," in *IEEE INFOCOM 2009*, 2009, pp. 999–1007.
[15] I. Lotfi, H. Du, D. Niyato, S. Sun, and D. I. Kim, "On the robustness of channel allocation in joint radar and communication systems: An auction approach," *IEEE Transactions on Mobile Computing*, vol. 23, no. 4, pp. 3466–3483, 2024.
[16] Y. Li, Y. He, J. Lin, Z. Xu, and S. Zhang, "A reinforcement learning-based population hyper-heuristic for energy-efficient cloud workflow scheduling problem," *IEEE Transactions on Services Computing*, pp. 1–14, 2025.
[17] R. Besharati, M. H. Rezvani, and M. M. Gilanian Sadeghi, "An auction-based bid prediction mechanism for fog-cloud offloading using q-learning," *Complexity*, vol. 2023, no. 1, p. 5222504, 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1155/2023/5222504
[18] H. Ren, K. Liu, G. Yan, C. Liu, Y. Li, C. Li, and W. Wu, "Truthful auction mechanisms for dependent task offloading in vehicular edge computing," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 14 987–15 002, 2024.
[19] X. Liu, L. Liu, Y. Yuan, Y.-H. Long, S.-X. Li, and F.-Y. Wang, "When blockchain meets auction: A comprehensive survey," *IEEE Transactions on Computational Social Systems*, vol. 11, no. 3, pp. 4242–4254, 2024.
[20] R. B. Myerson, "Optimal auction design," *Mathematics of operations research*, vol. 6, no. 1, pp. 58–73, 1981.
[21] K. Roberts, "The characterization of implementable choice rules," *Aggregation and revelation of preferences*, vol. 12, no. 2, pp. 321–348, 1979.

[22] A. Likhodedov, T. Sandholm *et al.*, "Approximating revenue-maximizing combinatorial auctions," in *AAAI*, vol. 5, 2005, pp. 267–274.
[23] M. Curry, T. Sandholm, and J. Dickerson, "Differentiable economics for randomized affine maximizer auctions," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, E. Elkind, Ed. International Joint Conferences on Artificial Intelligence Organization, 8 2023, pp. 2633–2641, main Track. [Online]. Available: https://doi.org/10.24963/ijcai.2023/293
[24] Z. Duan, H. Sun, Y. Chen, and X. Deng, "A scalable neural network for dsic affine maximizer auction design," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 56 169–56 185. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/af31604708f3e44b4de9fdfa6dcaa9d1-Paper-Conference.pdf
[25] X. Liu, W. Li, and X. Zhang, "Strategy-proof mechanism for provisioning and allocation virtual machines in heterogeneous clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 7, pp. 1650–1663, 2018.
[26] J. Zhang, N. Xie, X. Zhang, and W. Li, "Strategy-proof mechanism for online time-varying resource allocation with restart," *Journal of Grid Computing*, vol. 19, pp. 1–20, 2021.
[27] L. Mashayekhy, M. M. Nejad, and D. Grosu, "Physical machine resource management in clouds: A mechanism design approach," *IEEE Transactions on Cloud Computing*, vol. 3, no. 3, pp. 247–260, 2015.
[28] G. Gao, M. Xiao, J. Wu, H. Huang, S. Wang, and G. Chen, "Auction-based vm allocation for deadline-sensitive tasks in distributed edge cloud," *IEEE Transactions on Services Computing*, vol. 14, no. 6, pp. 1702–1716, 2021.
[29] Q. Wang, S. Guo, J. Liu, C. Pan, and L. Yang, "Profit maximization incentive mechanism for resource providers in mobile edge computing," *IEEE Transactions on Services Computing*, vol. 15, no. 1, pp. 138–149, 2022.
[30] H. Peng, Y. Zhan, D.-H. Zhai, X. Zhang, and Y. Xia, "Egret: Reinforcement mechanism for sequential computation offloading in edge computing," *IEEE Transactions on Services Computing*, vol. 17, no. 6, pp. 3541–3554, 2024.
[31] E. H. Clarke, "Multipart pricing of public goods," *Public choice*, pp. 17–33, 1971.
[32] T. Groves, "Incentives in teams," *Econometrica: Journal of the Econometric Society*, pp. 617–631, 1973.
[33] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *The Journal of finance*, vol. 16, no. 1, pp. 8–37, 1961.
[34] R. Lavi, A. Mu'alem, and N. Nisan, "Towards a characterization of truthful combinatorial auctions," in *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, 2003, pp. 574–583.
[35] Z. Duan, H. Sun, Y. Xia, S. Wang, Z. Zhang, C. Yu, J. Xu, B. Zheng, and X. Deng, "Automated deterministic auction design with objective decomposition," 2024. [Online]. Available: https://arxiv.org/abs/2402.11904
[36] P. Dütting, Z. Feng, H. Narasimhan, D. Parkes, and S. S. Ravindranath, "Optimal auctions through deep learning," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 1706–1715. [Online]. Available: https://proceedings.mlr.press/v97/duetting19a.html
[37] D. Ivanov, I. Safiulin, I. Filippov, and K. Balabaeva, "Optimal-er auctions through attention," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 34 734–34 747. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/e0c07bb70721255482020afca44cabf2-Paper-Conference.pdf
[38] Z. Duan, J. Tang, Y. Yin, Z. Feng, X. Yan, M. Zaheer, and X. Deng, "A context-integrated transformer-based neural network for auction design," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 5609–5626. [Online]. Available: https://proceedings.mlr.press/v162/duan22a.html
[39] A. Mu'alem and N. Nisan, "Truthful approximation mechanisms for restricted combinatorial auctions," *Games and Economic Behavior*, vol. 64, no. 2, pp. 612–631, 2008, special Issue in Honor of Michael B. Maschler. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S089982560800050X
[40] Tianchi, "cluster-trace-gpu-v2020," 2020. [Online]. Available: https://github.com/alibaba/clusterdata/tree/master/cluster-trace-gpu-v2020

Jixian Zhang received an M.S. degree in computer science from the University of Electronic Science and Technology of China in 2006 and a Ph.D. in computer science from the University of Electronic Science and Technology of China in 2010. He is an associate professor at the School of Computer Science and Engineering at Yunnan University. He has published 30+ articles in peer-reviewed journals and conferences, e.g., TMC, TSC, TNSM, TNSE, TCSS, FGCS, JOGC, Cluster Computing, and Computing. His research interests include cloud computing, edge computing and mechanism design.

Xuelin Yang received a B.S. degree in computer science and technology from Xidian University in 2023. She is a master's student at Yunnan University and is expected to obtain a master's degree in 2026. Her main research directions include deep learning, mechanism design, and combinatorial optimization.

Weidong Li received a Ph.D. from the Department of Mathematics, Yunnan University, in 2010. He is currently a professor at the School of Mathematics and Statistics of Yunnan University. He has published 180+ articles in peer-reviewed journals and conferences, e.g., ALGO, JOA, NRL, EJOR, TPDS, TMC, TVT, TNSM, TNSE, TCSS, Sci. Sin. Inform, JPDC, MSCS, TCS, DAM, CN, JOGC, FGCS, IPL, 4OR, OL, JOCO, FCS, APJOR, IJCM, DMAA, ORF, RAIRO-OR, JORSC, JOCC, CEE, TJOS, TOETT, CLUSTER COMPUT and Computing. His main research interests include computational economics and discrete optimization.

Wenzhong Li (M'08) received B.S. and Ph.D. degrees from Nanjing University, China, both in computer science. He was an Alexander von Humboldt Scholar Fellow at the University of Göttingen, Germany. He is now a full professor in the Department of Computer Science, Nanjing University. Dr. Li's research interests include distributed computing, data mining, mobile cloud computing, wireless networks, pervasive computing, and social networks. He has published more than 100 peer-reviewed articles in international conferences and journals, which include INFOCOM, UBICOMP, IJCAI, ACM Multimedia, ICDCS, IEEE Communications Magazine, IEEE/ACM Transactions on Networking (ToN), IEEE Journal on Selected Areas in Communications (JSAC), IEEE Transactions on Parallel and Distributed Systems (TPDS), and IEEE Transactions on Wireless Communications (TWC). He served as Program Cochair of MobiArch 2013 and Registration Chair of ICNP 2013. He was a TPC member of several international conferences and a reviewer for many journals. He is the principal investigator of three projects that are funded by the NSFC and the coprincipal investigator of a China–Europe international research staff exchange program. Dr. Li is a member of the IEEE, ACM, and China Computer Federation (CCF). He also received the Best Paper Award from ICC 2009 and APNet 2018. He was featured in Elsevier's Most Cited Chinese Researchers list in 2022 and 2023.