Future Generation Computer Systems 138 (2023) 313–327

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Reinforcement learning based monotonic policy for online resource allocation

Pankaj Mishra [a,b,*], Ahmed Moustafa [a,c]

[a] Department of Computer Science, Nagoya Institute of Technology, Japan
[b] School of Computing and Information Technology, University of Wollongong, Australia
[c] Faculty of Informatics, Zagazig University, Egypt

ARTICLE INFO

Article history:
Received 14 April 2021
Received in revised form 9 August 2021
Accepted 17 September 2021
Available online 13 September 2022

Keywords:
Resource allocation
Reinforcement learning
Online mechanism design
Monotonic policy
Critical payment
Resource dominant clustering

ABSTRACT

This research aims to design an optimal and strategyproof mechanism for online resource allocation problems. In such problems, consumers randomly arrive with their resource requests in an arbitrary manner. As a result, there is uncertainty in the future resource demands. In addition, the allocation and payment decisions depend on the providers' past experiences. To address these challenges, we propose a novel reinforcement learning algorithm for optimising the resource allocation policy. The proposed algorithm adopts a novel monotonic reward shaping function that uses a dominant-resource multi-label classification technique. Finally, a critical payment value is calculated in order to maintain the strategyproofness in the online environment. The experimental evaluations show that the proposed mechanism achieves results that are within 96% of the optimal social welfare while outperforming the other mechanisms that use fixed pricing.

1. Introduction

Online mechanism designs are the extension of mechanism design for resource allocation in online resource allocation *ORA* paradigms. In such *ORA* paradigms, the consumer(s) submit their resource request sequentially but in an arbitrary manner to provider with resource constraints [1]. Therefore in such a setting, the allocation rule for resource request of current consumer purely depends on the knowledge of previously arrived consumers. Besides, there is uncertainty about the future resource demand in the *ORA* paradigm. Therefore, designing a mechanism for *ORA* paradigm is fundamentally different from designing a mechanism for offline paradigms. On the other hand, in an offline (classical) setting, the mechanism waits until all the consumers arrive. Then it makes allocation decision using classical auction mechanisms [2] (i.e., second-price auction, first-price auction, etc.). In this research, we focus on building a strategyproof resource allocation mechanism for an *ORA* paradigm, which maximises the social welfare of the provider. There are many real-world applications wherein such online mechanisms are being used, not limited to, selling air-plane tickets, cloud resource allocation [3,4], real-time bidding in display advertisement [5–7], sponsored search [8], dynamic fleet management [9], fog computing [10], real-time ride-sharing [11]. However, in this research, we specifically focus on modelling an online mechanism in cloud-based markets.

Designing an online mechanism for *ORA* is a complex problem that has various challenges which need to be addressed. Firstly, since consumers are arriving sequentially, so details of the future resource demands are not known. Therefore, a mechanism should trade-off the immediate profits of the providers with future potential profits. In specific, an online mechanism should not exhaust (allocate) all the resources to early arriving (might be) less paying consumers, so that all the future potential high paying consumers are rejected. Secondly, since the consumer arrives arbitrarily, so the mechanism is not aware of the available feasible decision set in the near future. Therefore, an online mechanism should learn and adapt itself based on previously arrived consumers. Finally, for any competitive market with time-sensitive resource request, self-interested consumers behave strategically and misrepresent their type (i.e., true resource valuation, the volume of resources, etc.,) to increase their chances of winning. Therefore, the online mechanism should also be strategy-proof [12], such that consumers are incentivised to report their resource requirements truthfully.

In the last decade, many research related to online mechanism [13] have been proposed to solve this classical secretary problem [14]. In this context, different online variant of Vickrey–Clarke–Groves (*VCG*) mechanisms were proposed by [15–17], which focus on Bayesian-Nash incentive compatibility. However, these approaches assumed that the distribution of arrival of future resource demand is known in advance by the mechanism.

* Corresponding author.
E-mail addresses: panksmish@gmail.com (P. Mishra), ahmed@nitech.ac.jp (A. Moustafa).

https://doi.org/10.1016/j.future.2021.09.023
0167-739X/

Similarly, many posted-price based online mechanism [11,18], wherein reserve prices are computed for each allocation. Since these mechanisms are purely based on a static mathematical model, so they do not adapt to the dynamics of the environment. Further, an online mechanism based on exhibiting the monotonic properties has been proposed [19,20]. However, in this research, the authors assumed that the providers have unlimited resources with partial knowledge of future consumers. Similarly, various types of online mechanism have been proposed focusing on different resource allocation characteristics in different domains. For instance, [21,22] focused on building a bid-density based pricing function in cloud computing. Then, [23] focused on scaling problem in cloud computing. They proposed an algorithm to take allocation decision based on load distribution. However, all the above online mechanisms are based on underlying static distribution and also did not consider the adaptation of the mechanism over time. Later, to tackle this problem of adaptability, [24] proposed an online mechanism based on dynamic programming. However, this mechanism has ignored the strategic behaviour of the arriving consumers and assumed that consumers would not misreport their type.

Recently, research combining machine learning and mechanism design is being proposed. For instance, [25–27] proposed machine learning-based strategyproof auction. However, these mechanisms, do not consider the complex dynamics of the online allocation, i.e., fail to consider the trade-off between the current and future rewards. In such resource allocation environment, reinforcement learning (RL) [28] has been successfully applied [29–32]. However, these resource allocation algorithms only focus on adapting the dynamics of the environment and fail to address strategyproofness. Further, [33] proposed a *RL* based strategyproof and adaptive online mechanism. However, since the *RL* algorithm implemented linear function approximation, so it was difficult for the mechanism to adapt to complex state dependencies. Briefly, all the above-mentioned online mechanisms are either based on static models or assume that the mechanism is aware of the future arrival of consumers. Also, none of the existing online mechanisms handled the strategyproof and dynamics of the environment completely. Therefore, to address the above-mentioned challenges and limitations, we propose an online mechanism based on Proximal Policy Optimisation (*PPO*) algorithm [34], which guarantees the monotonic [35] behaviour of the allocation policy based on the novel reward-shaping algorithm. In addition to that, we build a novel adaptive critical payment based pricing technique using the *PPO* algorithm. In this regard, the contributions of this research are as follows:

- First, a novel *PPO*-based allocation rule, which adapts to the online environment and maintains the strategyproofness in the environment.
- Second, we introduce a novel reward shaping technique based on multi-class clustering on the previously arrived consumers.
- Third, we introduce critical-payment based pricing rule for online mechanism using the multi-class clustered data of previously arrived consumers

The rest of this paper is organised as follows. The preliminaries for modelling the resource allocation problem are introduced in Section 2. Section 3 presents the proposed learning-based real-time pricing algorithm. In Section 4, we discuss the properties of the novel *MP-ORA* mechanism. Further, in Section 5, the experimental results are presented for evaluating the proposed approach. Then, in Section 6, we discuss existing studies on resource allocation mechanisms. Finally, the paper is concluded in Section 7.

2. Preliminaries

In this section, we present the *ORA* paradigm and its challenges associated with online mechanism design. In particular, we discuss the modelling of the *ORA* paradigm and then we describe how allocation and pricing rules are modelled in order to address the challenges in designing an online mechanism.

2.1. System model

In *ORA* paradigm, there are two types of participants, namely, the provider, who is the decision maker (accept or reject the resource request) and the others are the consumers, which are sequentially requesting resources. In this work, we consider a dynamic market setting with a single provider who has a limited volume of resources (e.g., storage capacity, bandwidth, computing speed, etc.). On the other hand, set of consumers which arrive sequentially but at random and submit their resource request in each time step. Besides, consumers arrive in an episodic manner and each episode is represented as finite and discrete time horizon T of length t_{max}, s.t. $T = \{1, 2, \ldots, t_{max}\}$. In this setting, provider is denoted as P, whereas, set of all the consumers arriving until time step t_{max} is denoted as C, s.t., $|C| = t_{max}$. The provider P has m different types of resources (for example: *CPU*, *Bandwidth*, *Memory*, etc.,) that are being requested by the consumers sequentially at each time-step, we denote these resources as $R = \{r_1, r_2, \ldots, r_m\}$. Also, for each resource type $r \in R$, provider P has limited units of resources, s.t., $\boldsymbol{a}^t \in N$, s.t., $a^t_{r_1} = 100$ denotes 100 units of resource type $r_1 \in R$ at time-step t. Then, consumer $i \in C$ arrives at time step $t \in T$ and submits their type ϑ_i to provider P. The type ϑ_i for consumer $i \in C$ represents the resource request information denoted as $\vartheta_i = (v_i, q_i, d_i)$, where $v_i \in \mathbb{R}^+$ is the consumer's valuation of vector for requested resources $q_i = [q_{i,r_1}, q_{i,r_2}, \ldots, q_{i,r_m}] \in \mathbb{N}^{r_m}$ of duration size $d_i \in \mathbb{N}$ time-steps. Throughout this paper, we assume that if resource request from consumer $j \in C$ is allocated at time-step $t \in T$, then it is completed at $t + d_j$. Since, single consumer arrives at each time-step, therefore we represent consumer with their corresponding time-step, i.e., for consumer arriving at $t \in T$ is denoted as t and its corresponding type is denoted as ϑ_t. In this research we consider an online setting, wherein arriving consumers are impatient, such that allocation decision of every arriving consumer has to made immediately without any delay. In this context, o Also, let ϑ_i denote the type space of consumer i and $\vartheta = (\vartheta_1, \vartheta_2, \ldots, \vartheta_{t_{max}})$ denote the type profile space for all the consumers. Also, recall that, in *ORA* setting, provider P at time-step $t \in T$ is aware of current consumer with type ϑ_t and all the consumers which have arrived before that denoted as $\vartheta_{<t}$; whereas set of all the consumers arrived by t_{max} is denoted as T.

2.2. Online mechanism design

Online mechanism for every consumer $t \in T$ is denoted as $M = alloc(\vartheta_t, \vartheta_{<t}), pay(\vartheta_t)$, where $alloc(\vartheta_t, \vartheta_{<t}) \in \{0, 1\}$ represents the allocation rule and $pay(\vartheta_i) \in \mathbb{N}$ denotes the pricing rule. Specifically, $alloc(\vartheta_t, \vartheta_{<t}) = 0$ when mechanism M rejects the consumer i, and $alloc(\vartheta_t, \vartheta_{<t}) = 1$ when consumer's resource request is allocated. Then, finally based on the pricing rule $pay(\vartheta_i)$, payment for the consumer is computed, such that, if $pay(\vartheta_i) > 0$, then consumer i pays to the provider and if $pay(\vartheta_i) < 0$ then consumer receives same amount of payment from the provider through the mechanism. It should be noted mechanism M makes feasible allocations only, i.e., sum of all the units of allocated resources would be less than or equal to total available resources R. Also, for consumer $t \in T$, it would be allocated only if $d_i \leq t_{max} - t$. Further, we represent the online

mechanism constraints as represented in [33] using Eqs. (1) and (2).

$$\sum_{t=1}^{t_{max}} alloc(\vartheta_t, \vartheta_{<t})\mathbf{1}(t + d_t > t_{max}).q_{t,r} \leq a_r \forall t, r, \vartheta, \tag{1}$$

$$t + alloc(\vartheta_t, \vartheta_{<t}).d_t \leq t_{max} \forall t, \vartheta \tag{2}$$

where $\mathbf{1}()$ denotes the indicator function keeping the check on the time-step, such that if $d_t \leq t_{max} - t$, then $\mathbf{1}() = 1$. Further, based on the payment rule $pay(\vartheta_i)$, consumer's payment value is computed. Finally, at the end of time-step t_{max}, social-welfare of the provider is computed as sum of expected valuation from all the winning consumers as denoted in Eq. (3).

$$sw = \sum_{t=1}^{t_{max}} alloc(\vartheta_t, \vartheta_{<t}).v_t \tag{3}$$

One of the objectives of designing an online mechanism is to model an allocation rule such that, it maximises the social-welfare. Specifically, our objective is to design an optimal allocation policy ($alloc^*()$) such that it maximises the expected sw of the provider, as shown in Eq. (5).

$$alloc^* = \arg\max_{alloc} \mathbb{E}[\sum_{t=1}^{t_{max}} alloc(\vartheta_t, \vartheta_{<t}).v_t] \tag{4}$$

2.3. Strategyproof for online mechanism

In this subsection, we would present the challenges associated with handling the strategic behaviour of the participating consumers. Generally, in competitive resource allocation problems with limited resources, greedy consumers behave strategically to purchase these resources at least reported valuation. For instance, a consumer would report a lower valuation or higher volume of requested resources to get the requested resources at the minimum possible price. In this context, with such a strategic but realistic setting, it is challenging to develop a mechanism with accurate allocation and pricing rules. Therefore, an online mechanism should be capable of handling the strategic behaviours of the consumers without affecting the individual utility of the consumers. Before introducing the properties that are to be exhibited while building an online mechanism, we introduce some definitions.

Definition 2.1 (*Reported Type*)**.** In this research, the true type of a consumer, $\vartheta_i = (v_i, q_i, r_i)$ is its private information, i.e., it is not known by the mechanism. The reported type, $\vartheta'_i = (v'_i, q'_i, r'_i)$ is the actual type revealed by the consumer to the mechanism on arrival. Therefore, when $\vartheta_i = \vartheta'_i$, then we say that the consumer is truthful.

Definition 2.2 (*Limited Misreport*)**.** In *ORA* paradigm, for consumer $i \in C$, let $L(\vartheta_i) \subseteq \vartheta_i$, denote the set of type from its type space available for manipulation.

In an *ORA* setting, wherein consumers arrive sequentially, so there is a set of limited misreport associated with each consumer. In specific, consumer $i \in C$ can misreport its valuation v_i and quantity of resources q_i. However, in an online setting, a consumer is not aware of its type until its arrival and needs to be serviced immediately, so it cannot misreport its arrival time. Also, the consumer has no benefit in waiting after the completion of their resource request. Therefore it is practical to assume that the consumer has limited late-departure misreport, i.e., $d'_i \leq d_i$. Further, in this research, we assume that consumer is allowed only one participation. In this setting, the individual utility of the consumer is defined as follows:

Definition 2.3 (*Consumer Utility*)**.** The utility of an allocated consumer t in mechanism M is given by $u_t(\vartheta'_t, \vartheta_t, \vartheta'_{<t}) = alloc(\vartheta'_t, \vartheta'_{<t})v_t - pay(\vartheta'_t, \vartheta'_{<t})$.

In this research, we aim to incentivise the consumers to report true type, so that the online mechanism could handle the strategic behaviour of the consumers. Generally, for a resource allocation mechanism $M = (alloc(), pay())$ to be able to handle strategic behaviour, it must exhibit two key properties, i.e., *strategyproofness* and *individual rationality*.

Definition 2.4 (*Strategyproof*)**.** An online mechanism M is strategyproof if consumer's utility is maximised for reporting its true type. Formally: $u_i(\vartheta_i, \vartheta_t, \vartheta'_{<t}) \geq u_t(\vartheta'_t, \vartheta_t, \vartheta'_{<t}), \forall \vartheta_t, \vartheta'_t, \vartheta'_{<t}$. This is also called as dominant strategy incentive compatible (*DSIC*).

Definition 2.5 (*Individual Rationality*)**.** An online mechanism M is individual rational if the utility of the consumer is non-negative until and unless consumer is reporting its true value. Formally: $u_t(\vartheta_i, \vartheta'_{<t}) \geq 0$. This property also makes sure that any truthful consumer is not made to pay more than their valuation, i.e. $pay(\vartheta_t, \vartheta_{<t}) \leq v_t$.

Intuitively, *Strategyproof* property ensures that consumers would be given incentives for reporting their true type, whereas *Individual Rationality* ensures that no consumer is forced to purchase resources or consumers would be having positive utility as long as they report their true type. Further, similar to [33] setting, we also consider a single-minded resource allocation setting, i.e., each consumer will not accept the allocation offer from the provider if offered resources are less than q_i.

In the offline setting, *Strategyproof* property is usually achieved by implementing *VCG* [15] mechanism, wherein consumers payment value depends on the other participating consumers as follows,

$$pay(\vartheta_i) = \sum_{j \in \vartheta_{-i}} v_j - \sum_{j \in \vartheta, j \neq i} v_j \tag{5}$$

However, the classical form of *VCG* mechanism cannot be directly adopted in an online resource allocation setting. Since online mechanisms are not aware of all the arriving consumers at time-step t. In this regard, based on the Lemma 16.12 and Lemma 16.13 in [1], online mechanisms are said to be strategyproof if the online mechanism has monotonic allocation policy (see Definition 2.6) and critical payment policy (see Definition 2.7).

Definition 2.6 (*Monotonic Allocation*)**.** An allocation rule $alloc(.)$ in online mechanism is monotonic, if any consumer allocated remains allocated for any other reported type which is at least as good as previous reported type. Specifically, if consumer i with type $\vartheta_i = (v_i, q_i, d_i)$ is allocated, then consumer j with type $\vartheta_j = (v_j, q_j, d_j)$ will also be allocated, if $\vartheta_j \succeq \vartheta_i$, i.e., if $v_j \geq v_i \wedge q_j \leq q_i \wedge d_j \leq d_i$. Formally: $alloc(\vartheta_i, \vartheta_{<i}) = 1 \rightarrow alloc_j(\vartheta_j, \vartheta_{<j}) = 1, \forall \vartheta_j \succeq \vartheta_i$.

Definition 2.7 (*Critical Payment*)**.** A payment rule $pay(.)$ in online mechanism is critical if the consumer pays the minimum which was enough for that particular consumer to remain allocated. Formally critical payment rule is shown in Eq. (6).

$$pay(\vartheta_i, \vartheta_{<i}) = \begin{cases} min(v'_i) & \text{if } alloc_i(\vartheta'_i, \vartheta_{<i}) = 1 \\ \infty & \text{if no such } v'_i \text{ exists,} \end{cases} . \tag{6}$$

In this regard, the monotonic allocation policy ensures that the consumers would always win as long as the consumer reports equal or more than their winning reported type. On the

other hand, since, the critical payment does not depend on the consumer's valuation, so the consumer has no incentive to manipulate its type. In this regard, based on the above two techniques, a strategyproof online mechanism is designed.

In the next section, we would introduce the implementation of the above techniques using reinforcement learning (RL) algorithm, to design a self-learning online mechanism.

3. Monotonic policy based *ORA*

In this section, we would present a novel *MP-ORA* algorithm for resource allocation problem in *ORA* setting. In particular, we modified the Proximal Policy Optimisation algorithm (*PPO*) an RL-algorithm using the dominant resource technique to observe truthfulness in the proposed resource allocation mechanism. In this context, a novel *PPO* based mechanism $M \equiv (alloc(), pay())$ is implemented at the provider's end, and the provider is the acting agent which optimises its allocation rule $alloc()$ and pricing rule $pay()$ for sequentially arriving consumers. Therefore the proposed Monotonic Policy-based *ORA* (MP-ORA) algorithm has two major steps, first computing the resource dominant parameters using the proposed Resource Dominant Clustering technique, then using this cluster data to build a *PPO* based learning mechanism as discussed in the following subsections.

3.1. Resource dominant clustering

In this research, we adopt a novel clustering-based dominant resource strategy [36] to categorise the consumers into two types of cluster category. Generally, in an offline setting, the resource allocation mechanism utilises the dominant resource parameters to sort [37] the bidding consumers and allocated the bidders based on their dominant resource parameter in descending order. However, in *ORA* setting, the resource allocation mechanism has to make allocation and pricing decision without any delay purely based on past experiences, so at any instant of time, there is a single consumer. In *ORA* setting, in every episode of length t_{max}, consumers arrive sequentially one in each time-step $\forall t \in T$. In this context, auction begins as soon a consumer at time-step t reports its type ϑ'_t. Since the resource allocation mechanism has to make allocation and pricing decision without any delay purely based on past experiences. In this context with knowledge of a single consumer and uncertainty of the future consumers, designing the allocation decision becomes challenging. To compute the dominance of a single resource request, we utilise the previously arrived consumers and a novel cluster technique. In specific, we categorise consumer-type into two types of consumer clusters at each time-step $t \in T$, namely, $\boldsymbol{K}^t_d$ consumer clusters based on the set of reported sizes, i.e. $d'_{<t}$ and the other is $\boldsymbol{K}^t_q$ consumer clusters on the quantity of reported resource requests, i.e., $q'_{<t}$. The idea is to utilise this cluster information to categorise the dynamically arriving consumers into one of the cluster, then utilise this information to shape the rewarding function and the pricing function in the proposed mechanism. In particular, after consumer $t \in T$ with reported type $\vartheta'_t = (v'_t, q'_t, d'_t)$ is rejected or accepted by the mechanism, then the training model is rewarded based on the taken decisions. In this regard, the reported type of all the previously arrived consumers $\vartheta_{<t}$ stored in *Transaction Database* is utilised to scale the impact of the decision at time-step t. In order to do that, two classes of clusters of consumers arrived until time-step $t-1$, namely, reported size clusters denoted as $\boldsymbol{K}^t_d$ and other reported requested quantity clusters denoted as $\boldsymbol{K}^t_q$. This cluster information is stored in a *Transaction Database* and it is updated after every allocations. Further based on these clustered information, we compute two types of dominant parameters for the consumer at time-step $t \in t_{max}$. These two dominant parameters compute the dominance of reported type at time-step t computed based on the similar reported type similarly arrived consumers in past. Further, these dominant parameters are used as a basis to compute the rewards and the further the allocation price of the allocating consumers.

Firstly, in order to compute dominant size density parameter (η^d_t), we select a cluster from reported size clusters $\boldsymbol{K}^t_d$ using cluster elicitation function $K(\delta^d_t)$, s.t. $max(v'_j/\delta^d_j) \leq (v'_t/\delta^d_t)$, where δ^d_t is the size density reported by the consumer t, $j \in K(\delta^d_t)$, and $K(\delta^d_t) \in \boldsymbol{K}^t_d$. Then, we compute the dominant size density parameter η^d_t for consumer t using Eq. (7).

$$\eta^d_t = \frac{\sum_{j \in K(\delta^d_t)} v_j/\delta^d_j}{|K(\delta^d_t)|} \tag{7}$$

where, $|K(\delta^d_t)|$ denotes the size of the cluster; whereas, size density δ^d_c for a consumer $c \in T$ is computed using Eq. (8). Intuitively, Eq. (8) computes the part of the total available time-span at time-step c occupied by the allocated consumer. Also, let $vd^n_{\mathbf{K}(\delta^d_t)}$ denote the nth highest valuation size density in the cluster $\mathbf{K}(\delta^d_t)$, s.t., $vd^1_{\mathbf{K}(\delta^d_t)} = max(v_j/\delta^d_j)$, $\forall j \in \mathbf{K}(\delta^d_t)$.

$$\delta^d_c = \frac{d'_c}{t_{max} - c} \tag{8}$$

Similarly, in order to compute dominant request density parameter (η^q_t) from the reported requested quantity clusters $\boldsymbol{K}^t_q$, again we select a cluster $K(\delta^q_t)$, s.t. $max(v'_j/\delta^q_j) \leq (v'_t/\delta^q_t)$, where δ^q_t is the normalised request density for consumer t, $j \in K(\delta^q_t)$, and $K(\delta^d_t) \in \boldsymbol{K}^t_d$. Then, we compute the dominant request density η^q_t using Eq. (9) as follows.

$$\eta^q_t = \frac{\sum_{j \in K(\delta^q_t)} v_j/\delta^q_j}{|K(\delta^q_t)|} \tag{9}$$

where, $K(\delta^q_t)$ represents the clustering function, whereas normalised request density δ^q_c for consumer c arrived at time-step $c \in T$ is computed using Eq. (10).

$$\delta^q_c = \frac{\sum_{r \in R} normalise(dr_{(c,r)})}{|R|} \tag{10}$$

where, R is the set of all the resources and $dr_{(c,r)}$ is the resource request density computed using Eq. (11), whereas, $normalise(.)$ is the normalising function for different types of resource using Eq. (12).

$$dr_{(c,r)} = \frac{q'_{c,r}}{a^c_r} \tag{11}$$

$$normalise(x) = \begin{cases} \frac{x^{max} - x}{x^{max} - x^{min}}, & \text{if } x^{max} - x^{min} \neq 0. \\ 1, & \text{if } x^{max} - x^{min} = 0. \end{cases} \tag{12}$$

Also, let $vq^n_{K(\delta^q_t)}$ denote the nth highest valuation request density in the cluster $K(\delta^q_t)$, s.t., $vq^1_{K(\delta^q_t)} = max(v_j/\delta^q_j)$, $\forall j \in K(\delta^q_t)$. Intuitively, dominant size density parameter η^d_t denotes the dominance of the valuation per size density of consumer t among the similar cluster of consumers arrived in the past. Similarly, request density parameter η^q_t denotes the dominance of valuation per requested resource density of consumer t amongst the other consumers belonging to similar cluster of consumers arrived in the past. Finally, these cluster information are updated in the *Transaction Database* at the end of every allocation, which is also used for representing the state of the environment and shaping an monotonic reward function as discussed further.

Table 1
Consumer reported type sample at time-step 10.

# ID	v'_t	q'_t	d'_t	$dr_{t,r} = q'/a^t$	δ^d	δ^q	v'_t/δ^d_t	v'_t/δ^q_t
c_1	1.9	(3,20,60)	3	(0.06,0.1,0.1)	0.015	0.3	126	6.3
c_2	6.9	(12,32,25)	8	(0.25,0.17,0.04)	0.044	0.32	156	21.56
c_3	3.0	(6,24,30)	15	(0.14,0.15,0.05)	0.078	0.73	39	5.3
c_4	4.1	(10,32,25)	2	(0.032,0.25,0.05)	0.0113	0.8	362	5.1
c_5	2.3	(2,13,32)	3	(0.07,0.011,0.07)	0.016	0.77	143	2.98
c_6	4.1	(6,26,51)	4	(0.12,0.34,0.12)	0.0103	0.17	398	24
c_7	2.5	(1,14,21)	14	(0.04,0.19,0.055)	0.072	0.34	34	7.3
c_8	3.5	(3,6,62)	4	(0.17,0.09,0.19)	0.021	0.35	166	20
c_9	1.2	(3,8,20)	3	(0.2,0.14,0.06)	0.015	0.39	80	4.8
c_{10}	3.0	(2,12,28)	18	(0.17,0.24,0.1)	0.096	0.83	31	3.6

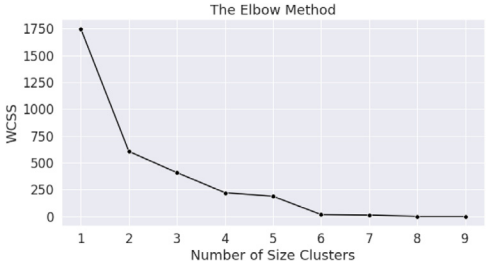

Fig. 1. Value of K in size density class.

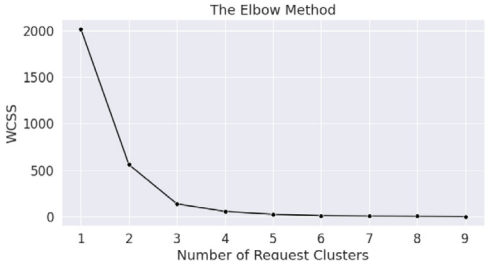

Fig. 2. Value of K in request density class.

Table 2
Dominant cluster values.

# Cluster	$\boldsymbol{K}^{11}_d$	$\boldsymbol{K}^{11}_q$
$\#cluster_1$	(c_4, c_6, c_8)	$(c_3, c_4, c_6, c_{10}, c_{10})$
$\#cluster_2$	(c_2)	(c_2)
$\#cluster_3$	(c_1, c_5, c_7, c_9)	(c_1, c_5, c_7, c_9)
$\#cluster_4$	(c_3, c_8)	–

3.1.1. Dominant clustering example

In this subsection, we briefly demonstrate the novel resource dominant clustering method. In this example, we demonstrate the clustering technique for a consumer c_{11} at time-step $t = 11$. Also, let $|R| = 3$ and $\boldsymbol{a^1} \equiv (50, 200, 600)$ and $t_{max} = 200$. In this context, Table 1 enlists all the reported type of ten consumers $c_1, \ldots, c_{10}$ arrived before c_{11}, along with their corresponding $dr_{t,r}$, δ^d, δ^q_t, v'_t/δ^d_t and v'_t/δ^q_t.

Further, at $t = 11$, c_{11} report their type $\vartheta'_{11} \equiv (8.2, (3, 9, 51), 9)$ and let $a^{11} \equiv (20, 120, 350)$. In this setting, firstly, consumers $c_1, \ldots, c_{10}$ are clustered into two different classes, $\boldsymbol{K}^{11}_d$ and $\boldsymbol{K}^{11}_q$ based on $\delta^d_{<11}$ and $\delta^q_{<11}$, respectively. In this context, number of clusters, i.e., value of K in each class is determined based on the *within cluster sum of squares* (WCSS) [38] technique using elbow method. For instance, from Figs. 1 and 2 it can be seen that, $K = 4$ and $K = 3$ for $\boldsymbol{K}^{11}_d$ and $\boldsymbol{K}^{11}_q$, respectively.

Moving further, consumers are clustered into two classes as shown in Figs. 3 and 4, using the identified values of K. The details of these two classes of clusters are listed in Table 2.

Fig. 3. Size density clusters $\boldsymbol{K}^{11}_d$.

Fig. 4. Request density clusters $\boldsymbol{K}^{11}_q$.

Then using Eq. (8), we compute δ^d_{11} as $\delta^d_{11} = 9/(200 - 11) = 0.047$. Similarly, using Eq. (10) we compute δ^q_{11}. In this regard, to compute δ^q_{11} we normalise reported resource request using Eq. (12). In doing so, from Table 1, *min* and *max* values for all resource request densities are noted, such as, $(dr^{min}_1; dr^{max}_1) \equiv (0.06, 0.17)$; $(dr^{min}_2; dr^{max}_2) \equiv (0.011, 0.34)$ and $(dr^{min}_3; dr^{max}_3) \equiv (0.04, 0.19)$. Further, we compute resource request densities for $c_1 1$, as $dr_{11,1} = 3/20 = 0.15$; $dr_{11,2} = 9/120 = 0.075$; and $dr_{11,3} = 51/350 = 0.14$. Then, based on these values, using Eq. (12), $normalise(dr_{11,1}) = dr^{max}_1 - dr_{11,1}/dr^{max}_1 - dr^{min}_1 = (0.17 - 0.15)/(0.17 - 0.06) = 0.18$; $normalise(dr_{11,2}) = (0.34 - 0.075)/(0.34 - 0.011) = 0.8$; and $normalise(dr_{11,3}) = (0.19 - 0.14)/(0.19 - 0.04) = 0.33$. Then, using these computed values, $\delta^q_{11} = (0.18 + 0.8 + 0.33)/3 = 0.43$; $v'_t/\delta^d_t = 8.2/0.047 = 174$; and $v'_t/\delta^q_t = 8.2/0.43 = 19$.

Now, we elect a single cluster from $\boldsymbol{K}^{11}_d$, in which maximum bid to size ratio is at max v'_{11}/δ^d_{11}. Therefore, from Tables 1 and 2, with random tie-breaking we elicit cluster $cluster_4 \subseteq \boldsymbol{K}^{11}_d$, i.e., $K(\delta^d_{11}) \equiv cluster_4 \equiv \{c_3, c_8\}$. Similarly, we elect a single cluster from $\boldsymbol{K}^{11}_q$, in which maximum bid to request ratio is at max v'_{11}/δ^q_{11}. Therefore, from Tables 1 and 2, with random tie-breaking we elicit a cluster $cluster_3 \subseteq \boldsymbol{K}^{11}_q$ is chosen, i.e., $K(\delta^q_{11}) \equiv cluster_3 \equiv \{c_1, c_5, c_7, c_9\}$.

Finally, using Eqs. (7) and (9), we compute $\eta^d_{11} = (39 + 166/2) = 102.5$ and $\eta^q_{11} = (6.3 + 2.98 + 7.3 + 4.8/4) = 5.34$. Note that, in this example, $vq^1_{K(\delta^d_t)} = v'_8 = 3.5$, whereas $vq^1_{K(\delta^q_t)} = v'_{c_7} = 2.5$. In this regard, resource dominant parameters are computed.

3.2. PPO learning algorithm

In this subsection, we would model the *ORA* problem into the Markov decision process (MDP), and then present the modified *PPO algorithm*. Firstly, we model the provider P as an agent interacting in an *ORA* market environment. In this regard, the state-space of the environment for every episode of length $T = \{1, 2, \ldots, t_{max}\}$ is denoted as $S = \{s_1, s_2, \ldots s_{t_{max}}\}$. Also, recall that, single consumer arrives at each time-step $t \in T$, there s_t denotes state change on arrival of consumer of reported type $\vartheta'_t = (v'_t, q'_t, d'_t)$. Also, in *ORA* setting at any time-step $t \in T$, provider is

also aware about the set of previously arrived consumers, i.e., $\theta'_{<t}$. On the other hand, in an offline setting, the provider is aware of all the consumers, so the state is represented using a type of all the current bidding consumers. In either case, the state must be represented using all the available information about the consumers. So in our setting ideally should be represented using all the past consumers along with current consumer i.e., $s_t \sim \{\vartheta'_{<t}, \vartheta'_t\}$. However, as the number of consumer increases, such representation would be computationally expensive. Therefore, we represent the past consumer in the state using the cluster information and consider only the past consumers from the *Transaction Database* which is in the same cluster as the ϑ'_t.

In particular, state is represented using ϑ_i, requested request density δ^d_t on the available resources, size density δ^d_t on the total time-step to t_{max} and the occupancy $o_{r,j}$ of the each resource type, where $r \in R$ and $j \in \{1, 2, \ldots, a_r\}$. Here occupancy $o_{r,j}$ represents the time-step at which single unit of resource type r will be available. For instance, if $o_r = [0, 1, 1, 2, 5]$, then single unit of resource will be available at time-step 0, then at time-step 1, 2 more units of resource r_1 will be available and directly at time-step 5, one more unit will be available. Besides, we include the corresponding consumer clusters $\mathbf{K}(\delta^d_t)$ and $\mathbf{K}(\delta^q_t)$ (see Section 3.1). In this regard, the final state is denoted using Eq. (13).

$$\begin{aligned} s_t = [&\vartheta_t, \rho_{t,r_1}, \ldots, \delta^q_t, \delta^d_t, \{o_{1,1}, \ldots, o_{1,a_1}\}, \ldots, \\ &\{o_{r_{max},1}\}, \ldots, \{o_{r_{max},a_{r_{max}}}\}, \\ &\mathbf{K}(\delta^d_t), \mathbf{K}(\delta^q_t)] \end{aligned} \tag{13}$$

In this context, $s_{t_{max}}$ is the terminal state, therefore, after t_{max}, i.e., at $t_{max}+1$, the state will be reset and reward will be zero. Further, at each time-step $t \in T$ provider P has continuous action $A_t \in [0, 1]$, wherein 0 usually means reject and 1 usually means accept. Further, at each time-step $t \in T$ on arrival of consumer θ'_t, provider P mainly have two actions, either to accept i.e., $act_t = 1$ or to reject, i.e., $act_t = 0$. However, in this research, apart from the above two extreme actions, we also consider intermediate actions, we consider a continuous action-space $A_t \in [0, 1]$ at each time step. This intermediate actions would aid the mechanism to understand the extent at which *MDP* model is willing to accept of reject the consumer request. Then after performing action act_t, state of the environment is transferred to next state s_{t+1} based on the transition function τ, s.t. $\tau : s_t \times A_t \rightarrow s_{t+1}$. Finally, agent P receives reward $reward_t$ based on the state s_{t+1} and action act_t. Usually, in such auction based resource allocation setting, reward is function of the bidding values [4], s.t. reward $reward_t$ for taking action act_t in-state s_t is denoted as $\Re(s_t, act_t) = v'_t$. However, in order to observe truthfulness in the proposed mechanism, we implemented a novel monotonic reward function based on resource dominant parameters as depicted in Algorithm 1.

Algorithm 1: Monotonic Reward Function

Input: act_t, η^d_t, η^q_t
Result: $reward_i$
/*initialisation*/;
$reward^d_t$, $reward^q_t$;
if *cannotSchedule*(ϑ'_t) **then**
  $reward^d_t = vd^1_{\mathbf{K}(\delta^d_t)}$; // positive reward: rejection is good decision
  $reward^q_t = vq^1_{\mathbf{K}(\delta^q_t)}$; // positive reward: rejection is good decision
else
  /* reward based on reported size */ ;
  if $\eta^d_t * d'_t \leq v'_t$ **then**
    $reward^d_t = -1 * vd^1_{\mathbf{K}(\delta^d_t)}$; // negative reward: rejection is not good decision
  else
    $reward^d_t = vd^1_{\mathbf{K}(\delta^d_t)} * d'_t$; // positive reward: rejection is better decision
  end
  /* reward based on reported resource request */ ;
  if $\eta^q_t * q^{avg}_t \leq v'_t$ **then**
    $reward^q_t = -1 * vq^1_{\mathbf{K}(\delta^q_t)}$; // negative reward: rejection is not good decision
  else
    $reward^q_t = vq^1_{\mathbf{K}(\delta^q_t)} * q^{avg}_t$; // positive reward: rejection is better decision
  end
end
$reward_t = (reward^d_t + reward^q_t)/2$;
return $reward_t$;

This algorithm takes three input, i.e., action act_t and two resource dominant parameters η^d_t and η^q_t. Using this algorithm we compute two types of reward, namely, reward based on reported size d'_t and reward based on reported resource request denoted as $reward^d_t$ and $reward^q_t$ respectively. Then finally, the algorithm gives an average of both the reward as an output. In this setting, the judgement of the reward values are based on the respective resource dominant parameters. Then, these rewards are classified as *negative reward*, *positive but good reward* and *positive but better reward*, wherein *positive but better reward* being the optimal solution. The intuition behind the classification is that if a particular reported d'_t or q'_t is greater than the respective resource dominant parameter and it is accepted, then it is an optimal allocation and positive reward is computed, or else is it is not a good allocation and negative reward is computed.

Further, after modelling the *ORA* problem as *MDP* problem, then we implement RL-algorithm to train the agent to optimise the policies to choose an optimal action in each time-step. There exist different RL-algorithms, but we choose *PPO* algorithm for our problem mainly because of two reasons, namely, (1) Clipping technique: clipping the objective function not only reduces the complexity of the algorithm compared to trust region policy optimisation (*TPRO*) but also helps in achieving a monotonic policy update; and (2) *PPO* algorithm supports continuous action-space, which is appropriate for our setting. Besides, *PPO* algorithm is scalable and performance is better [39] as compared to other on-policy algorithms (A2C and ACKTR). Apart from that, the clipping feature of the algorithm helps in achieving a monotonic policy update. Therefore, we implemented *PPO* algorithm [34] which employs the resource dominant parameters to optimise the stochastic policy π_{θ_t} to choose an optimal action $act_t \in A_t$, $\forall t \in T$, s.t. $\pi_{\theta_t} : A_t \rightarrow [0, 1]$, where θ represents the RL-parameters. Also, in this research we use the *PPO-Clip* variant of the *PPO* algorithm, wherein policy parameter is updated using Eq. (14).

$$\theta_{t+1} = \arg\max_{\theta} \mathbb{E}_{s_t, act_t \ \pi_{\theta_t}}[L(s_t, act_t, \theta_k, \theta)] \tag{14}$$

wherein, L denotes the surrogate advantage function [40], which measures relative performance of new policy π_θ with respect to current policy π_{θ_t}, computed using Eq. (15).

$$L(s, act, \theta_t, \theta) = min(\frac{\pi_\theta(act|s)}{\pi_{\theta_t}(act|s)} Adv^{\pi_{\theta_t}}(s, act),$$

$$g(\epsilon_{t-1}, Adv^{\pi_{\theta_t}}(s, act))) \tag{15}$$

where, $Adv^{\pi_{\theta_t}}$ denotes the advantage function for the current policy π_{θ_t} computed using advantage estimation [41], and value of $g(.)$ is computed as follows:

$$g(\epsilon, Adv) = \begin{cases} (1+\epsilon)Adv, & \text{if } Adv \geq 0. \\ (1-\epsilon)Adv, & \text{if } Adv < 0. \end{cases} \tag{16}$$

where, ϵ is the clipping function computed using Eq. (17)

$$\epsilon_t = \begin{cases} min(\varepsilon, act_t), & \text{if } canSchedule(\vartheta'_t). \\ \varepsilon \end{cases} \tag{17}$$

where, ε is the clipping variable, a hyperparameter which resembles how far new policy is allowed to drift away from the old policy, whereas *canSchedule*() return true when resources are available with the provider. Algorithm 2 depicts the modified *PPO*, which takes policy parameters as input and returns an optimise action value act_t along with resource dominant parameters for each time-step.

Algorithm 2: Modified PPO-Clip

Input: θ_0: initial policy parameter ; ϕ_0: initial value function parameters
Input: set clip-ratio $\varepsilon \in [0.1, 0.3]$ & $act_0 \in [0.1, 0.3]$
$t = 1$;
while $t \neq t_{max}$ **do**
  Run policy $\pi_{\theta_{old}}$ for *T*' timesteps ;
  Compute advantage estimate $Adv_1, \ldots, Adv_K$;
  Observe state s_t from the environment ;
  Obtain act_{t-1} from previous time-step ;
  Obtain ϵ_{t-1} using Equation (17) ;
  Obtain $act_t \xleftarrow{\text{policy}} \pi(act_t|s_t, \vartheta'_t)$; using Equation (14) ;
  Obtain $reward_t$ using Algorithm 1 ;
  Compute the η^d_t and η^q_t using Equation (7) and (9) ;
  Asynchronous Update *PPO* model using gradient decent method ;
  return act_t ;
  $t++$;
end

Further, after computing the action value act_t for the consumer ϑ'_t, cost of the requested resources is computed using Eq. (18).

$$cost(\vartheta'_i) = \begin{cases} \frac{v^2_{\mathbf{K}(\delta^d_t)}.d'_i + v^2_{\mathbf{K}(\delta^q_t)}.q^{avg}_t}{2}.(1-act_i), & \text{if } \boldsymbol{K}(\delta^d_t), \boldsymbol{K}(\delta^q_t) \neq \emptyset. \\ v'_i(1-act_i) \end{cases} \tag{18}$$

From Eq. (18), the cost function computes the cost based on the reported valuation of the consumers in the selected cluster. Further, if the no similar consumers have arrived in the past (i.e. cluster $\boldsymbol{K}(\delta^d_t)$ or $\boldsymbol{K}(\delta^q_t)$ is empty), then cost is computed based on its reported valuation. On the other hand, the allocation decision based on Eq. (19), s.t., 1 means request is allocated or else it is rejected.

$$alloc(\vartheta_i, \vartheta_{<i}) = \begin{cases} 1 & \text{if } cost(\vartheta'_i) \leq v'_i \ . \\ 0 \end{cases} \tag{19}$$

Finally, using the computed cost and the allocation value, critical payment is computed for each consumer using Eq. (20)

$$pay(\vartheta'_i) = cost(\vartheta'_i) \times alloc(\vartheta'_i, \vartheta_{<i}) \tag{20}$$

In this regard, allocation and payment decisions for the arriving consumer are based on dominant resource strategy and the modified *PPO* algorithm. Also, it can seen from Eq. (18), since the payment is computed from other similar consumers but not the reported valuation, this satisfies the critical payment theory. However, it is a special case when $\boldsymbol{K}(\delta^d_t) = \emptyset$ or $\boldsymbol{K}(\delta^q_t) = \emptyset$, this can be seen as a market with single consumers, so the payment policy is computed based on reserve price mechanism [12].

3.3. Execution of MP-ORA

In this subsection we would present the working the novel *MP-ORA* mechanism. Continuing from the example discussed in Section 3.1.1, at time-stem $t = 11$, decision to allocate consumer c_{11} is to be made. In this context, let us assume $act_{11} = 0.1$ computed using Algorithm 2. Also from discussion in Section 3.1.1, $\boldsymbol{K}(\boldsymbol{\delta^d_{11}}) \equiv (c_3, c_8)$, so $v^2_{K(\delta^d_{11})} = v'_3 = 3.0$. Similarly, $\boldsymbol{K}(\boldsymbol{\delta^q_{11}}) \equiv (c_1, c_5, c_7, c_9)$, so $v^2_{K(\delta^q_{11})} = v'_5 = 3.5$. In this regard, from Eq. (18), $cost(\vartheta'_{11}) = 3 + 3.5/2(1-0.1) = 4.77$. Now since $4.77 \leq 8.2$, therefore c_{11} would be allocated (from Eq. (19)), s.t. $alloc(\vartheta_{11}, \vartheta_{<11}) = 1$. Finally, payment is computed as $pay(\vartheta'_{11}) = 4.77 * 1 = 4.77$.

In this transaction, provider *P* earns 4.77, whereas the utility of the consumer c_{11} is computed as $8.2 - 4.77 = 3.43$ (From Definition 2.3). Further, through this example, we would also show that how allocation decision in Eq. (19) ensures strategyproofness. To begin with, if the consumer reports greater than 8.2, then there could be two, there will be no change in the allocation and payment will still be 4.77. So, consumers do not have an incentive in reporting higher value. Further, if the consumer reports the valuation less than the computed critical value, i.e., 4.77, then based on Eq. (19), the consumer will not be allocated, so its utility will be zero. So we showed that consumers do not have any incentive in misreporting their valuations. Therefore, the novel *MP-ORA* mechanism ensures strategyproofness by computing critical payment value for all the arriving consumers. The detailed proof of this strategyproof property is discussed further.

4. Properties of MP-ORA

In this section, we investigate the properties of *MP-ORA*. We first show that the *MP-ORA* mechanism is individually rational (i.e., truthful users will never incur a loss).

Theorem 1. *The MP-ORA mechanism is individually rational.*

Proof. Let us consider a consumer c_t with reported type $\vartheta'_t = (v'_t, d'_t, q'_t)$, and $alloc(\vartheta'_t, \vartheta'_{<t}) = 1$ i.e., consumer t is allocated. Then in this proof we prove that, if reported type is true then its utility is non-negative. This can be easily be seen from Eq. (18), cost of the requested resources depends on the resource dominant parameters, η^d_t and η^q_t, which are computed using Eqs. (7) and (9). Since these parameters represent the valuation density in their corresponding clusters $\mathbf{K}(dr^d_t)$ group, such that maximum value is not more than the reported valuation v'_t. In this regard, always the computed payment is $pay(\vartheta'_t) \leq v'_t$. As a result, the utility of the consumer t, $u_t = v'_t - pay(\vartheta'_t) \geq 0$ is non-negative and thus consumer will never face loss. On the other hand, payment for truthful consumer which do not win is 0 from Eq. (20), since $alloc(\vartheta'_i, \vartheta_{<i}) = 0$. Therefore, in the *MP-ORA* mechanism utility of all the consumer is non-negative, thus *MP* = *ORA* mechanism is individually rational. □

We now prove that *MP-ORA* mechanism is incentive compatible. In order to prove that, we need to show that the allocation algorithm is monotone, and the payment rule is based on the critical payment.

Theorem 2. *The MP-ORA mechanism is incentive compatible.*

Proof. We would first show that the allocation rule in the *MP-ORA* mechanism is monotone. That is, if consumer $t \in T$ is allocated by reporting ϑ_t', then it will also be allocated if consumer reports more preferred request i.e., $\vartheta_t'' \subseteq \vartheta_t'$. In this regard, if $v_t'' \geq v_t'$, then clearly from Eq. (19), ϑ_t'' will also be allocated. Because, cluster $\mathbf{K}(\delta_t^d)$ and $\mathbf{K}(\delta_t^q)$ remains the same from Eqs. (7) and (9), respectively. Further, if the consumer is allocated on reporting d_t', then it will also be allocated with $d_t'' \leq d_t'$, as *canAllocate*() function will return true. Similarly, if consumer is allocated reporting q_t', then it will also be allocated with $q_t'' \leq q_t'$.

Further, we prove that the payment rule in *MP-ORA* mechanism is based on the critical payment. In order to show that, we will prove that $pay(\vartheta_t')$ computed in Eq. (20) is the least value that consumer $t \in T$ is expected to report to get the allocation. Since from Eq. (19), if $v_t' < cost(\vartheta_t')$, then $alloc(\vartheta_t') = 0$. Therefore, $cost(\vartheta_t')$ is the critical value for the consumer t which satisfies Definition critical payment property. □

We now show that novel *MP-ORA* mechanism has polynomial time complexity.

Theorem 3. *The time complexity of MP-ORA mechanism is polynomial.*

Proof. Firstly, Algorithm 1 i.e. Monotonic Reward Function comprises of three primary computations requiring steps, clustering of past consumers, sorting of cluster and checking the feasibility of scheduling. Firstly, clustering of past consumers at time-step t has the $O(C^2)$ time complexity. Then the time complexity of sorting the chosen clusters $K(\delta_t^d)$ and $K(\delta_t^q)$ is $O(Clog(C))$. Finally, the time complexity of checking the feasibility of the allocation for every arrival of consumer would be $O(CR)$. As a whole overall complexity of Algorithm 1 is $O(C^2 + Clog(C) + CR)$, i.e. polynomial time.

Further, Algorithm 2, i.e. Modified PPO-Clip is the *PPO* based learning algorithm which has a learning network to generate action values for every allocation request. Therefore, its time complexity depends on the implemented fully connected neural network [42]. In this regard, the time complexity of the fully connected neural network depends on the number of neural nodes performing multiplication operations [43]. In this context, the time complexity is polynomial-time represented as $O(\sum_{f=1}^{F} n_f n_{f-1})$, where n is the number of neural nodes and f represents the fully connected layers. Since the time complexity of both the algorithms is in polynomial time, so the time complexity of the novel *MP-ORA* mechanism is polynomial (see Table 3). □

5. Experimental evaluation

In this section we present the experimental setting and results to investigate the properties of our proposed *MP-ORA* mechanism for online resource allocation domain based on range of experimental settings. The extensive experimental settings were performed on real like workload for three types cloud resources, i.e., C.P.U, memory and storage. Throughout the experiment, we use following hyper-parameters for *MP-ORA*: $\varepsilon = 0.1$ *(clip ratio)* ; $\gamma = 0.9$ *(discount factor)* ; *learning-rate* $= 3e^{-4}$; *epochs* $= 20$, *batch* $-$ *size* $= 64$, as these hyper-parameters gave good results. Towards the end, we compare the proposed *MPORA* mechanism with following benchmarks.

- *Offline Greedy*: An offline setting wherein future consumer type is already known, i.e., unrealistic solution for *ORA* paradigm. This is similar to the classical winner determination problem, and the mechanism elects the consumers in decreasing order of their valuations.

Table 3
List of symbols.

# Symbol	# Definitions
P	The provider
T	Time horizon up to max-time step t_{max}
C	Set of consumers arrived withing time-horizon T
$R = \{r_1, r_2, \ldots, r_m\}$	m types of resources available with the provider
a_{r_i}	Units of resources of type $r_i \in R$
ϑ_i	Reported type of the consumer $i \in C$
v_i & d_i	Reported valuation and size of the consumer of consumer i
$q_i = \{q_{i,r_1}, \ldots, q_{i,r_m}\}$	q_{i,r_j} units of resources requested by consumer i, where $r_j \in R$
$alloc(\vartheta_t, \vartheta_{<t})$	Allocation policy of the mechanism M
$pay(\vartheta_t)$	Pricing policy of the mechanism M
$\vartheta_{<t}$	Set of all the consumers arrived until time-step t
$\mathbf{K_d^t}$	Clusters of consumers based on reported size d arrived before time step t
$\mathbf{K_q^t}$	Clusters of consumers based on reported resource request q arrived before time step t
δ_i^d & δ_i^q	Size density & resource density, respectively of the consumer $i \in C$
η_t^d & η_t^q	Dominant size density & dominant request density, respectively of the consumer i
$K(\delta_i^d)$ & $K(\delta_i^q)$	Cluster elicitation function
$vq^n_{K(\delta_t^q)}$	nth highest valuation request density in the cluster $K(\delta_t^q)$

- *Optimal Pricing*: A posted-pricing based algorithm, wherein allocation decisions are based on the fixed reserve price. Specifically, the consumer's request is allocated only if $v_i' \geq \psi$, where ψ are optimally selected reserve price from a pre-defined specific range of prices, as discussed in [44].
- *First-Come-First-Serve (FCFS)*: This is the simple schedule with no mechanism, and this algorithm would allocate the consumer's request as long as there are still sufficient resources available.
- *Strategyproof-Reinforcement Learning (SP-RL)*: This is a linear approximation based strategyproof reinforcement learning mechanism [33]. In this algorithm, the strategyproof property is used based on the concepts of monotonic allocation rule and critical payment rule.

From the above-mentioned benchmarks algorithms, *Offline Greedy*, *Optimal Pricing*, and *FCFS* are evaluated based on average of *2000* iteration. On the other hand, *SP-RL* and *MP-ORA* algorithms are evaluated based on training the algorithm from scratch after every *100* episodes for *10,000* training episodes, to ensure noise reduction. Finally, all the algorithms are implemented in *Python 3* and the experiments are performed on *Intel Xeon 3.6 GHz 6* core processor with *32 GB RAM*.

5.1. Experimental setup

We use the Google cluster trace [45] to run the simulation. In particular we extract consumers resource request for three types of resources, namely, *CPU*, *Memory* and *Disk Storage* from the *Task Events Tables* in Google cluster traces, i.e., $m = 3$ s.t, $R = \{CPU, Memory, Storage\}$. In the provided dataset, the volume of each requested resources i.e., $q_i = \{q_{(i,CPU)}, q_{(i,Memory)}, q_{(i,Storage)}\}$ are the re-scaled values and not the actual values. Further, based

on these re-scaled values we divide the consumers into two categories, namely, moderate consumers (ϑ'_α) and heavy consumers (ϑ'_β) using a median split technique [46]. Then, we scale these re-scaled values by considering the maximum re-scaled value as one unit for each resource type and then scaled all the requested units by it.

Also, since, *size* for each resource request is not publicly available in the dataset (i.e., *Task Events Tables*). Therefore, similar to [33], we simulate the *size* for both d'_α and d'_β sampling uniformly at random from [1, 30]. Similarly, we simulate the valuation v'_α and v'_α by sampling per unit price (*PUP*) for all three resources *CPU*, *Memory* and *Storage* uniformly at random from $PUP^{CPU} \in [10, 20]$, $PUP^{Memory} \in [5, 10]$ and $PUP^{Storage} \in [1, 10]$, respectively. The intuition behind choosing such distribution is that, cost of *CPU* is expensive as compared to other two, and storage is the least expensive. Further, using these per unit prices, valuation is computed for every consumer is computed $v'_i = d_i \sum^{R \in r} PUP_r * q_{(i,r)}, \forall i \in [1, t_{max}]$.

In this setting, we train the *PPO* algorithm for 10,000 training episodes, which has fixed length t_{max} preset at the beginning of the experiment. We also preset the total available resources $a_r = (CPU = 10, Memory = 50, Storage = 100)$. Finally, all the algorithms are implemented in *Python 3* [47] and are performed on *Intel Xeon 3.6 GHz 6* core processor with *32 GB RAM*.

In this setting, we then compare the performance of *MP-ORA* for different workloads. For each workload, we examine the execution time, the social welfare, the number of consumers allocated and the resource utilisation for each mechanism. In this context, resource utilisation for every resource type is the percentage of allocated resource out of the total capacity of that resource over the entire time. We now present the results obtained by *MP-ORA* and other benchmarks for the selected parameters.

5.2. Impact of type of consumer on the mechanism

This experimental setting evaluates the impact of the arrival of different consumer type (i.e., heavy consumer and moderate consumer) on four different evaluation parameters, namely, provider's social welfare, execution time (allocation delay), number of consumers allocated, and resource utilisation. In this regard, we design six test cases having different probabilities (e) at which heavy consumer arrives as follows: 0.05, 0.1, 0.15, 0.2, 0.25 and 0.3. Besides, we fix the total number of the consumer to $t_{max} = 200$ and the resource capacity multiplier as 40, i.e., $a_r = (CPU = 400, Memory = 2000, Storage = 4000)$. Firstly, from Fig. 5, it is observed that the social welfare of the provider in all the algorithms increases with the increase in the probability of heavy consumers. In particular, social welfare is maximum at $e = 0.3$ and least at $e = 0.05$. Besides, social welfare in the *SP-RL* algorithm is least as compared to *optimal pricing* and *MP-ORA*. Overall, in all the test cases, social welfare in *MP-ORA* is comparatively higher.

Fig. 6 depicts the execution time, i.e., the delay in allocation of the winning consumers. Note that execution time for non-learning based algorithms (*FCFS*, *Optimal Price* and *Offline Greedy*) is negligible. Therefore, we would focus only on learning-based algorithms (*SP-RL* and *MP-ORA*). From Fig. 6 it is observed that execution time in *MP-ORA* is comparatively much lower. Besides, execution time for *SP-RL* algorithm slowly rises with increase in value of e. On the other hand, execution time is either levelled off or drops slightly with an increase in e. This observation is interesting as it shows that, unlike *SP-RL*, the novel *MP-ORA* is adaptable to change in consumer type. In particular, at $e = 0.25$, execution time for *SP-RL* algorithm rises, whereas it drops for *MP-ORA* algorithm.

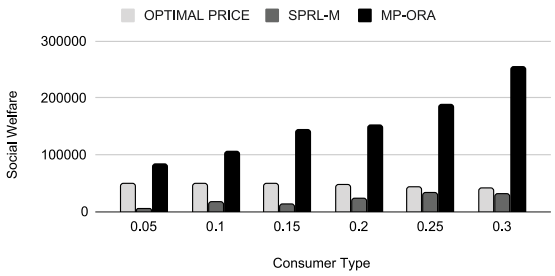

Fig. 5. Social of the provider.

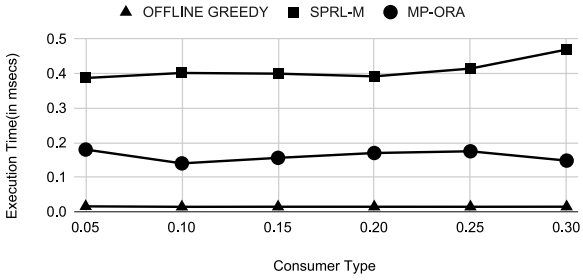

Fig. 6. Execution time.

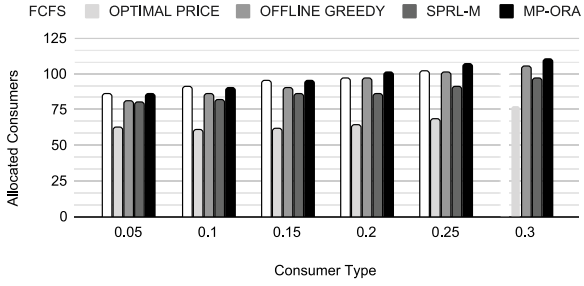

Fig. 7. Number of consumer completed.

Fig. 7 shows that average number of consumers allocated for different algorithms. Generally, a higher number of allocated consumers not necessarily denotes the efficiency of the algorithm. However, higher social welfare and the higher number of allocated consumers showcase the reliability of the system. Overall, the number of allocated consumers is steady with an increase in e and novel *MP-ORA* has the maximum number of consumers in all the test cases.

Further, Figs. 8–10 depict the resource utilisation of all the three types of resources in different test cases. It can be seen in the figure that, with a higher probability of heavy consumers, resource utilisation is also high. For instance, when $e = 0.25$, the resource utilisation is more as compared to when $e = 0.05$, $e = 0.1$ and $e = 0.15$. Thus the presence of heavy consumers has an impact on resource utilisation. Overall, resource utilisation is least for the *MP-ORA* algorithm in all the test cases, which showcases the availability of the resources in all the cases.

5.3. Impact of number of consumers

This experimental setting evaluates the impact of the increase in the number of consumers, i.e., increase in total time-step, wherein the probability of higher consumer, i.e., $e = 0.3$ and capacity multiplier $c = 40$, i.e., $a_r = (CPU = 400, Memory = 2000, Storage = 4000)$. In this setting, we design four test cases having different time-step $t_{max} = 200$, $t_{max} = 400$, $t_{max} = 600$

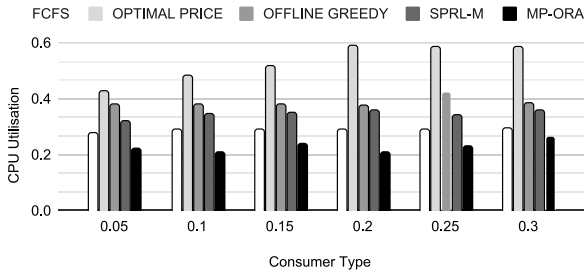

Fig. 8. CPU utilisation.

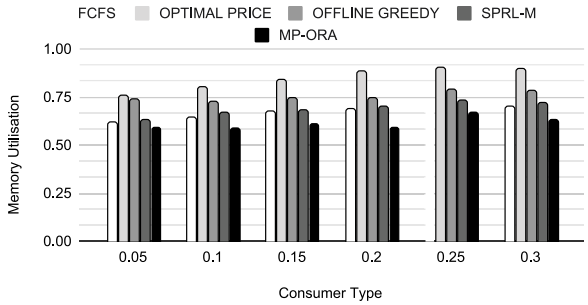

Fig. 9. Memory utilisation.

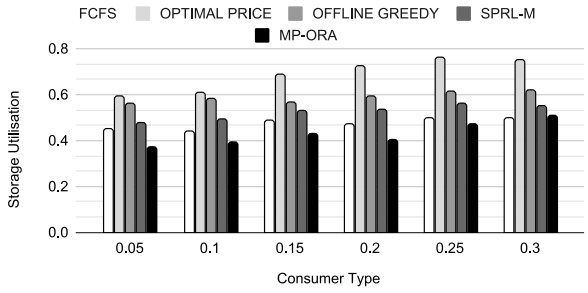

Fig. 10. Storage utilisation.

and $t_{max} = 800$. Similar to previous setting, we evaluate the mechanism based four different evaluation parameters, namely, provider's social welfare, execution time (allocation delay), number of consumers allocated, and resource utilisation.

Fig. 11 depicts that the provider's social welfare is directly proportional to the number of consumers served. Note that for *FCFS* and *Offline Greedy* algorithm, total social welfare is negligible compared to the other three algorithms. On the other hand, social welfare earned in *SP-RL* and *Optimal Price* are the same in all the test cases. On the other hand, social welfare earned in the *MP-ORA* algorithm doubles with the increase in the number of consumers. In particular, with the number of consumers *800*, social welfare is almost double compared to social welfare earned with *600*. Overall, social welfare earned in that *MP-ORA* is more compared to *SP-RL* algorithm in all test cases.

Fig. 12, execution time gradually increases in learning-based algorithms as the number of consumers increases. However, execution time is steady for non-learning based algorithms. Besides, execution time in *MP-ORA* increases in the beginning at reaches its threshold at 600 and then the execution time becomes steady.

Further, in Fig. 13, the average number of consumers allocated for different algorithms is shown. Overall, the number of allocated consumers steadily increases with an increase in the number of consumers. Apart from that, the novel *MP-ORA* has the maximum number of consumers allocated in all test cases.

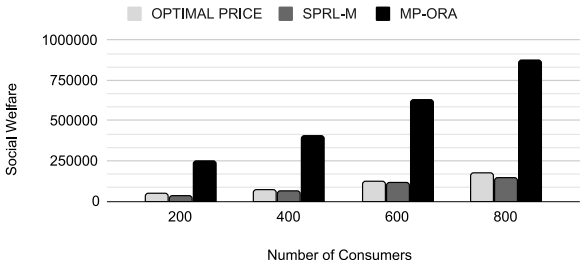

Fig. 11. Social welfare of the provider.

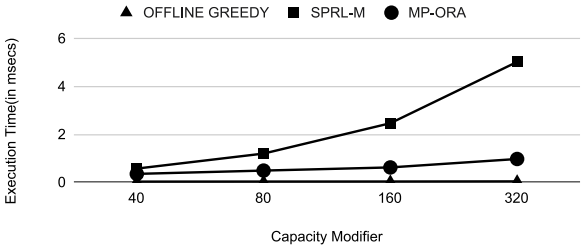

Fig. 12. Execution time.

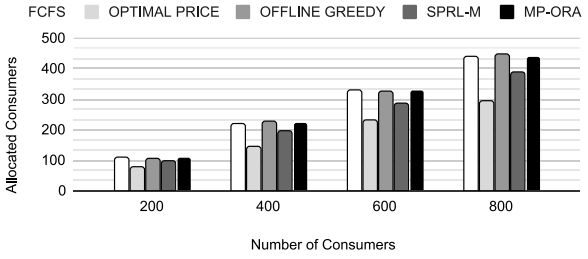

Fig. 13. Number of user completed.

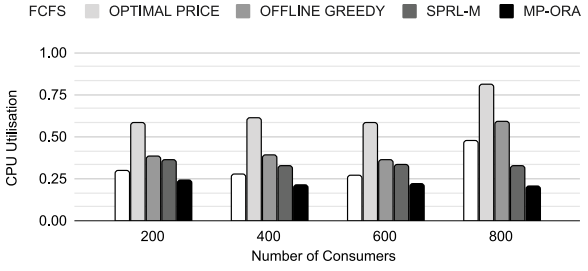

Fig. 14. CPU utilisation.

Further, Figs. 14–16 depict the resource utilisation of all the three types of resources in different test cases. Naturally, in an offline setting, with the increase in the number of consumers, resource utilisation must rise. However, in an online setting, all the resources are not exhausted at the same time. Therefore, the average utilisation of the resources remains lower as compared to the offline setting. Overall, in all test cases, resource utilisation in *Optimal Pricing* is the highest, whereas it is lowest in *MP-ORA* algorithm.

5.4. Impact of available resource capacity with the provider

In this experimental setting, we aim to evaluate the impact of the increase in the availability of resources with the provider, i.e., capacity multiplier (c). In this setting, probability of higher consumer, i.e., $e = 0.3$ and $t_{max} = 400$. We design four test

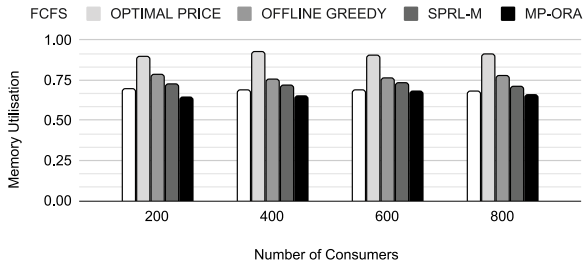

Fig. 15. Memory utilisation.

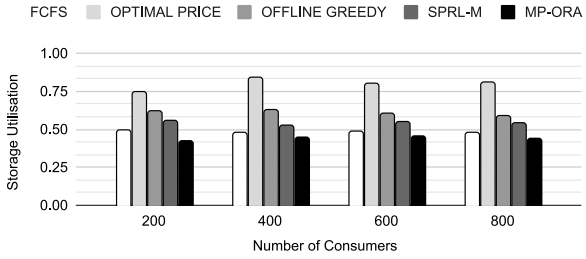

Fig. 16. Storage utilisation.

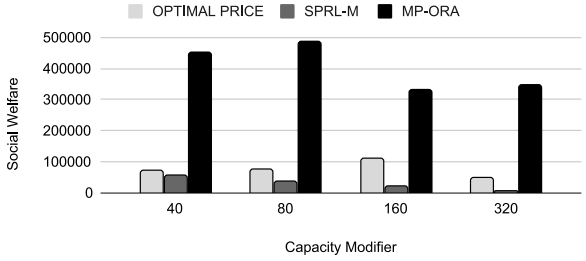

Fig. 17. Social welfare of the provider.

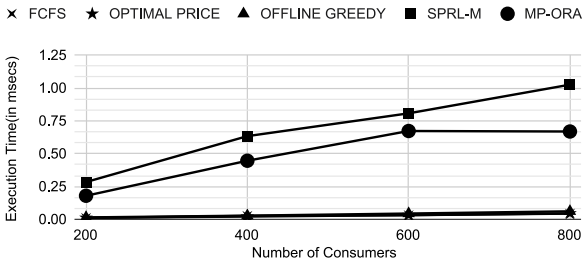

Fig. 18. Execution time.

Fig. 19. Number of user completed.

Fig. 20. CPU utilisation.

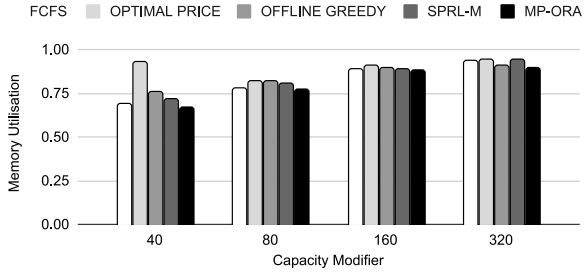

Fig. 21. Memory utilisation.

cases having different capacity multiplier, i.e., $c = 40$, $c = 80$, $c = 160$ and $c = 320$. Again, we evaluate the mechanism based four different evaluation parameters, namely, provider's social welfare, execution time (allocation delay), number of consumers allocated, and resource utilisation.

From Fig. 17, it is observed, that social welfare in *MP-ORA* mechanism is more as compared to other algorithms. Also, in the beginning, social welfare in the *MP-ORA* mechanism increases with an increase in total available resources but then decreases gradually. For instance, when $c = 40$, social welfare is maximum but then it drops when $c = 80$. One of the possible justification could be the greater interval between the two allocations as compared to their size of the requested resources. Thus, there could be many time-steps, wherein no allocation takes place.

Further, from Fig. 18, it is observed that execution time in *MP-ORA* remains lower but steady in all the test cases. On the other hand, the execution time in *SP-RL* algorithm gradually increases with the increase in available resources. Also, a sudden increase in the execution time at $c = 160$ is observed. One logical justification for such behaviour could be in the implementation of a linear function approximator in *SP-RL*. Because with the increase in the number of resources, the complexity of state representation increases.

From Fig. 19, it is observed that the number of consumers allocated, rises only in the beginning. Later the number of consumers allocated becomes steady in all the algorithms. In specific, number of allocated consumers increases when $c = 40$ is changed to $c = 80$, but for $c = 160$ and $c = 320$, average number of allocated consumers are approximately same. One of the interesting observation here is a number of allocated consumers are nearly the same in all the algorithms in every test cases.

Finally, from Figs. 20–22, the resource utilisation of all the three resources in different test cases is observed. The resource utilisation gradually increases with the increase in the availability of the resources. Although, utilisation is observed nearly the same in all the algorithm, in *MP-ORA* algorithm number of allocated consumers are comparatively higher.

5.5. Impact of time-varying resource request

In this experimental setting, we aim to evaluate the impact of the time-varying resource requests for different number of consumers. Also, we set the probability of higher consumer, i.e., $e = 0.3$ and capacity multiplier $c = 40$, i.e., $a_r = (CPU = 400, Memory = 2000, Storage = 4000)$. In this setting, we design

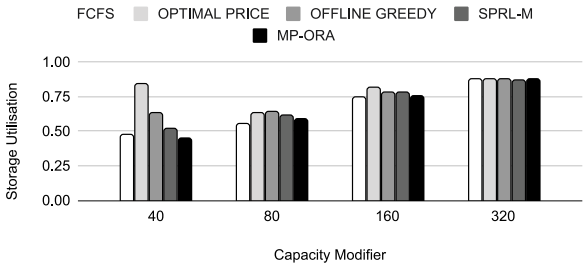

Fig. 22. Storage utilisation.

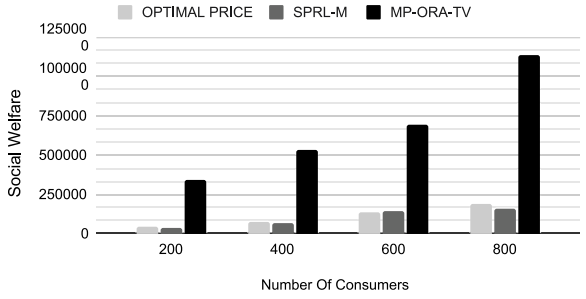

Fig. 23. Social welfare of the provider.

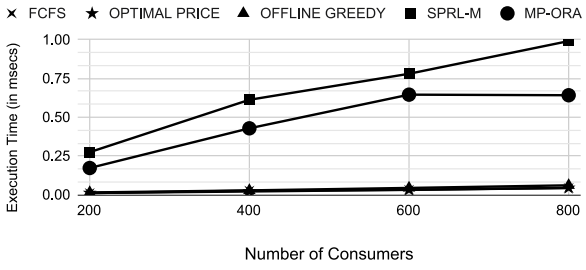

Fig. 24. Execution time.

four test cases having different time-step $t_{max} = 200$, $t_{max} = 400$, $t_{max} = 600$ and $t_{max} = 800$. Also, we evaluate the algorithm based four different evaluation parameters, namely, provider's social welfare, execution time (allocation delay), number of consumers allocated, and resource utilisation.

From Fig. 23, social welfare gradually increases with the increase in the number of arriving consumers. Also, similar to consistent resource requests setting, social welfare in *FCFS* and *Offline Greedy* algorithm are negligible comparatively. So social welfare observed in *SP-RL* and *Optimal Price* is nearly the same, whereas social welfare is rising as high as double compared to *SP-RL* algorithm. Therefore, again the overall, social welfare observed in that *MP-ORA* is more compared to *SP-RL* algorithm in all test cases.

Moving to execution time, from Fig. 24, similar to the consistent resource request, execution time observed in *MP-ORA* remains steady with the increase in the number of consumers. On the other hand, the execution time in *SP-RL* algorithm gradually increases with the increase in available resources. One interesting observation is, until $t_{max} = 600$, execution time rises gradually in *MP-ORA* algorithm, then it is levelled-off.

Further, from Fig. 25, it is observed that the number of allocated consumers in *MP-ORA* algorithm is maximum as compared to other algorithms, except for the *Offline Greedy* algorithm. Also, it is observed that initially, the number of allocations rises until it reaches its threshold, then begins to fall.

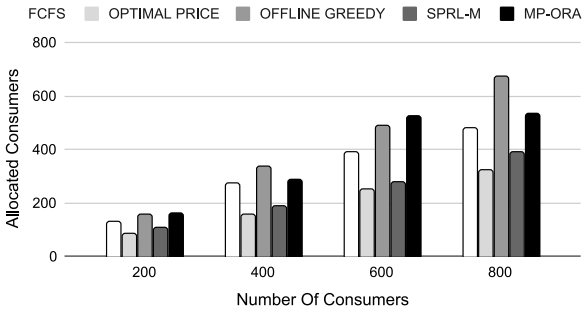

Fig. 25. Number of user completed.

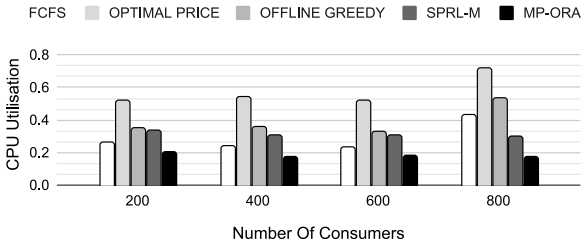

Fig. 26. CPU utilisation.

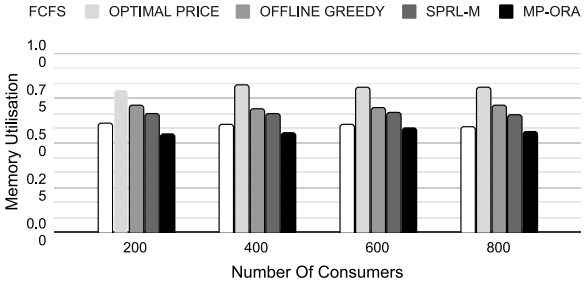

Fig. 27. Memory utilisation.

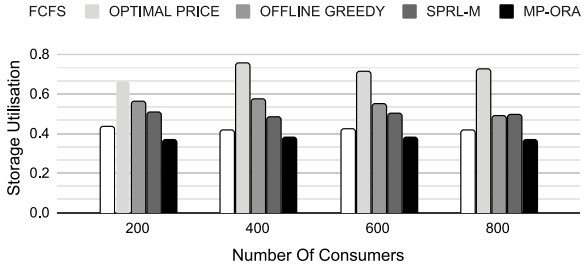

Fig. 28. Storage utilisation.

Furthermore, Figs. 26–28 depict the resource utilisation of all the three types of resources in different test cases, respectively. Since in an online setting, all the resources do not get allocated at once, so its resource utilisation always falls below the offline setting, i.e., *offline greedy* algorithm. Overall, in all test cases, resource utilisation is lowest in *MP-ORA* algorithm as compared to all the benchmarks algorithms.

Besides, we compare the performance of the novel *MP-ORA* algorithm with the consistent consumer requests denoted as *MP-ORA-C* and the time-varying consumer requests denoted as *MP-ORA-TV*. To begin with, from Fig. 29, the higher number of consumers are allocated in *MP-ORA-TV* setting, as compared to in *MP-ORA-C* setting. This rise in the number of allocated consumers

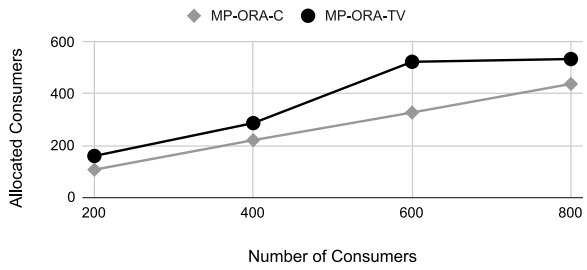

Fig. 29. Number of user completed.

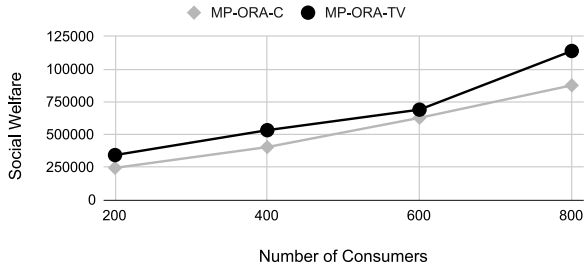

Fig. 30. Social welfare.

could be due to the higher availability of resources. An interesting observation is, in the beginning, the number of allocated consumers increases with the number of arriving consumers, but later it is levelled off. For instance until $t_{max} = 600$, number of allocated consumers reached a threshold value of about 550, but then it remained consistent for $t_{max} = 800$. One reason for this behaviour could be consumption of resources must have reached its peak value, such that no further consumers get allocated. Overall, this provides evidence that novel *MP-ORA* is compatible with the time-varying consumers.

Further, to provide evidence of the enhanced performance of the *MP-ORA* in a time-varying setting, we compare its social welfare with consistent resource requests. From Fig. 30, it is visible that, *MP-ORA-TV* naturally has higher social welfare, since it has a comparatively higher number of allocated consumers. However, from Fig. 30 it is seen that unlike Fig. 29, social welfare continues to increase with increase in number of consumers. This analysis illustrates that social welfare depends on the type of consumers allocated rather than the number of consumers allocated. Therefore, novel *MP-ORA* mechanism-based allocation decisions are optimised such that it maximises the overall social welfare.

Therefore, from the above results and discussion in all the experimental settings, it is clear that novel *MP-ORA* algorithm has higher social-welfare for the providers. Also, novel *MP-ORA* algorithm, ensures the stability of the online market by keeping efficient resource utilisation and truthfulness of the arriving consumers. Finally, from the time-varying experimental setting, we showed that the proposed approach is compatible with the time-varying resource requests.

6. Related work

Classical resource allocation problems are solved using different offline mechanisms [48,49], wherein the resource allocation mechanism is aware of the resource demands. The primary objective of such an offline mechanism is to maximise the social welfare of the providers. For instance, Wu et al. [50] proposed a clique based winner determination approach for combinatorial auctions to find optimal resource allocation matching. Similarly, Zhou et al. [49] proposed a branch-and-bound based optimal resource allocation approach. However, the existing offline mechanism fails to make optimal allocation decisions in practical settings with uncertain future demands. Owing to this heuristic algorithms combined with the offline mechanism were widely adopted for allocation and payment decisions. For instance, Nejad et al. [51] proposed an integer programming based mechanism for truthful allocation of resources in an auction paradigm. In this regard, many other heuristics approaches [52–54] have been proposed for the allocation of multi-unit resource requirements. These mechanisms could achieve truthfulness based on the monotonic allocation rule and critical value-based payment rule. However, the heuristic-based mechanism leads to higher execution and also lowered the social welfare of the provider. Besides, most of them were well suited for offline market settings.

Further, for decades online mechanisms are being studied [55], which is an extension of offline setting. The classical online problem is appointing an optimal secretary among the sequence of applications [14,56]. In the literature, many different solutions have proposed [10,16], which assumed that consumer arrives randomly from the set of the known distribution of consumers. However, these approaches are based on the static allocation rule, so they failed to address the challenges in an online setting. Recently a heuristic-based online approach does not guarantee a truthful mechanism, for instance, Zhang et al. [57], which is based on integer programming. However, the complexity of such algorithms increases with the number of constraints. Further, to adapt to the dynamically changing constraints in the online setting, machine learning-based online mechanisms are being implemented. For instance, [30,58] introduced an *RL* based online resource allocation mechanism. However, these approaches mainly focused on maximising social welfare and does not guarantee truthfulness. Later Blum et al. [26] proposed a strategyproof online mechanism for online resource allocation. Then, Babaioff et al. [59] proposed an online mechanism, wherein a stream of agents submit their resource request, which needs to be accepted or rejected immediately. In this regard, within the literature, Cai et al. [31], Du et al. [60] proposed *RL* based mechanism to allocate to strategic agents in a cloud computing environment. However, again this work focused on maximising the revenue and did not handle the strategyproofness. Besides, all the existing machine learning approaches considered the complex state space representing the correlation between the future and the current allocation decisions. Later, Stein et al. [33] proposed a similar online mechanism, which is closely related to our work, which partially represented the complex state space. However, state representation becomes complicated with the increase in the number of agents. In summary, existing work has archived good results in an online setting to some extent, however still there exist shortcomings: (1) existing online mechanism design cannot satisfy strategyproofness in such an uncertain setting; (2) representation of the state-space for dynamically arriving distinct consumers; and (3) adapting to the dynamically changing environment to maximise the social welfare of the provider. To address these shortcomings, we investigate the *RL* based online mechanism that would not only truthfully maximise the social welfare of the provider but also efficiently build the state space in such an online setting (see Table 4).

7. Conclusion

This research focuses on building a novel online mechanism to address the challenges associated with *ORA* paradigms. Specifically, we propose a reinforcement learning-based strategyproof an online mechanism that gradually adapts itself to the dynamics

Table 4
Comparison of our approach with related work.

# Mechanism	TV	OL/OF	Type	HT	IR	S
Angelelli et al. [48]	✗	OF	HU	✗	✗	✗
Zhou et al. [49]	✗	OF	AP	✗	✗	✗
Wu et al. [50]	✗	OF	AP	✗	✓	✓
Nejad et al. [51]	✗	OF	HU	✗	✗	✓
Liu et al. [53]	✗	OF	AP	✗	✓	✓
Zhang et al. [54]	✗	OL	HU	✓	✓	✓
Mashayekhy et al. [52]	✗	OF	AP	✗	✗	✓
Parkes et al. [55]	✗	OF	AP	✗	✓	✓
Cheng et al. [30]	✗	OF	LR	✓	✗	✗
Mao et al. [58]	✗	OF	LR	✓	✗	✗
Cai et al. [31]	✗	OF	LR	✓	✗	✗
Du et al. [60]	✗	OF	LR	✗	✗	✗
Jixian et al. [57]	✓	OL	HU	✓	✓	✓
Stein et al. [33]	✓	OL	LR	✓	✓	✓
MP-ORA	✓	OL	LR	✓	✓	✓

TV: time-varying; OL: online; OF: offline; HU: heuristic; AP:approximation; LR: Learning; HT: heterogeneous resource IR: individual rationality; S:strategyproof.

of the environment. In this context, we implement a custom proximal policy optimisation algorithm to build a monotonic allocation policy. Further, design a critical payment based payment policy. In this regard, the novel *MP-ORA* mechanism incentivises each consumer to report truthfully and maximise the social welfare of the provider. Towards the end, the novel *MP-ORA* mechanism outperformed the existing state-of-the-art mechanisms in series of experiments to evaluate social welfare and allocation efficiency. In addition, the proposed mechanism is faster as compared to the learning-based *SPRL* mechanism, having higher resource utilisation. For future work, we aim at designing an online mechanism for group resource allocation which would maximise the availability in presence of multiple independent resource providers of *ORA* paradigms.

CRediT authorship contribution statement

Pankaj Mishra: Conceived the presented idea, Developed the theory, Performed the computations, Verified the analytical methods, Investigation, Supervision, Writing of the final manuscript. **Ahmed Moustafa:** Conceived the presented idea, Verified the analytical methods, Writing of the final manuscript.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] N. Nisan, T. Roughgarden, E. Tardos, V. Vazirani, Algorithmic Game Theory, Cambridge Univ, 2007.
[2] V. Krishna, Auction Theory, Academic Press, 2009.
[3] D. Alsadie, Z. Tari, E.J. Alzahrani, A.Y. Zomaya, Dynamic resource allocation for an energy efficient vm architecture for cloud computing, in: Proceedings of the Australasian Computer Science Week Multiconference, 2018, pp. 1–8.
[4] P. Mishra, A. Maustafa, T. Ito, M. Zhang, Optimal auction based automated negotiation in realistic decentralised market environments, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 13726–13727.
[5] J. Jin, C. Song, H. Li, K. Gai, J. Wang, W. Zhang, Real-time bidding with multi-agent reinforcement learning in display advertising, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, ACM, 2018, pp. 2193–2201.
[6] J. Wang, S. Yuan, Real-time bidding: A new frontier of computational advertising research, in: Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, ACM, 2015, pp. 415–416.
[7] H. Cai, K. Ren, W. Zhang, K. Malialis, J. Wang, Y. Yu, D. Guo, Real-time bidding by reinforcement learning in display advertising, in: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, ACM, 2017, pp. 661–670.
[8] J. Zhao, G. Qiu, Z. Guan, W. Zhao, X. He, Deep reinforcement learning for sponsored search real-time bidding, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1021–1030.
[9] K. Lin, R. Zhao, Z. Xu, J. Zhou, Efficient large-scale fleet management via multi-agent deep reinforcement learning, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1774–1783.
[10] F. Bi, S. Stein, E. Gerding, N. Jennings, T. La Porta, A truthful online mechanism for allocating fog computing resources, 2019.
[11] W. Shen, C.V. Lopes, J.W. Crandall, An online mechanism for ridesharing in autonomous mobility-on-demand systems, 2016, arXiv preprint arXiv:1603.02208.
[12] R.B. Myerson, Optimal auction design, Math. Oper. Res. 6 (1) (1981) 58–73.
[13] E.J. Friedman, D.C. Parkes, Pricing wifi at starbucks: issues in online mechanism design, in: Proceedings of the 4th ACM Conference on Electronic Commerce, 2003, pp. 240–241.
[14] A. Karlin, E. Lei, On a competitive secretary problem, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 29, 2015.
[15] A. Gershkov, B. Moldovanu, Efficient sequential assignment with incomplete information, Games Econom. Behav. 68 (1) (2010) 144–154.
[16] D.C. Parkes, S. Singh, An MDP-based approach to online mechanism design, 2004.
[17] M.T. Hajiaghayi, R. Kleinberg, D.C. Parkes, Adaptive limited-supply online auctions, in: Proceedings of the 5th ACM Conference on Electronic Commerce, 2004, pp. 71–80.
[18] X. Tan, B. Sun, A. Leon-Garcia, Y. Wu, D.H. Tsang, Mechanism design for online resource allocation: A unified approach, Proc. ACM Meas. Anal. Comput. Syst. 4 (2) (2020) 1–46.
[19] R. Porter, Mechanism design for online real-time scheduling, in: Proceedings of the 5th ACM Conference on Electronic Commerce, 2004, pp. 61–70.
[20] M. Babaioff, R. Lavi, E. Pavlov, Mechanism design for single-value domains, in: AAAI, Vol. 5, 2005, pp. 241–247.
[21] L. Mashayekhy, M.M. Nejad, D. Grosu, A.V. Vasilakos, An online mechanism for resource allocation and pricing in clouds, IEEE Trans. Comput. 65 (4) (2015) 1172–1184.
[22] S. Chawla, J.B. Miller, Y. Teng, Pricing for online resource allocation: Intervals and paths, in: Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, 2019, pp. 1962–1981.
[23] Z. Xiao, Q. Chen, H. Luo, Automatic scaling of internet applications for cloud computing services, IEEE Trans. Comput. 63 (5) (2012) 1111–1123.
[24] W. Ma, B. Zheng, T. Qin, P. Tang, T.-Y. Liu, Online mechanism design for cloud computing, 2014, arXiv preprint arXiv:1403.1896.
[25] M.-F. Balcan, A. Blum, J.D. Hartline, Y. Mansour, Mechanism design via machine learning, in: 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05), IEEE, 2005, pp. 605–614.
[26] A. Blum, J.D. Hartline, Near-optimal online auctions, 2005.
[27] J. Dickerson, K. Sankararaman, K. Sarpatwar, A. Srinivasan, K.-L. Wu, P. Xu, Online resource allocation with matching constraints, in: International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2019.
[28] R.S. Sutton, A.G. Barto, et al., Introduction To Reinforcement Learning, Vol. 2, MIT press Cambridge, 1998.
[29] P. Mishra, A. Moustafa, T. Ito, Reinforcement learning based real-time pricing in open cloud markets, in: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Springer, 2020, pp. 419–430.
[30] M. Cheng, J. Li, S. Nazarian, DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers, in: 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), IEEE, 2018, pp. 129–134.
[31] Q. Cai, A. Filos-Ratsikas, P. Tang, Y. Zhang, Reinforcement mechanism design for e-commerce, in: Proceedings of the 2018 World Wide Web Conference, 2018, pp. 1339–1348.
[32] J. Lu, C. Yang, X. Gao, L. Wang, C. Li, G. Chen, Reinforcement learning with sequential information clustering in real-time bidding, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 1633–1641.
[33] S. Stein, M. Ochal, I.-A. Moisoiu, E. Gerding, R. Ganti, T. He, T. La Porta, Strategyproof reinforcement learning for online resource allocation, in: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, 2020, pp. 1296–1304.
[34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, 2017, arXiv preprint arXiv:1707.06347.

[35] Q. Yuan, N. Xiao, A monotonic policy optimization algorithm for high-dimensional continuous control problem in 3D MuJoCo, Multimedia Tools Appl. 78 (20) (2019) 28665–28680.
[36] W. Wang, B. Liang, B. Li, Multi-resource fair allocation in heterogeneous cloud computing systems, IEEE Trans. Parallel Distrib. Syst. 26 (10) (2014) 2822–2835.
[37] P. Samimi, Y. Teimouri, M. Mukhtar, A combinatorial double auction resource allocation model in cloud computing, Inform. Sci. 357 (2016) 201–216.
[38] J.A. Hartigan, M.A. Wong, Algorithm AS 136: A k-means clustering algorithm, J. R. Stat. Soc. Ser. C. Appl. Stat. 28 (1) (1979) 100–108.
[39] S.-s. Lee, S. Lee, Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information, IEEE Internet Things J. 7 (10) (2020) 10450–10464.
[40] J. Achiam, Spinning up in deep RL, 2018, URL https://spinningup.openai.com/en/latest/algorithms/ppo.html#proximal-policy-optimization Accessed: 2021-03-16.
[41] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: International Conference on Machine Learning, PMLR, 2016, pp. 1928–1937.
[42] Y. Zhan, P. Li, Z. Qu, D. Zeng, S. Guo, A learning-based incentive mechanism for federated learning, IEEE Internet Things J. 7 (7) (2020) 6360–6368.
[43] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, Deep Learning, Vol. 1, MIT press Cambridge, 2016.
[44] I.A. Kash, P. Key, W. Suksompong, Simple pricing schemes for the cloud, ACM Trans. Econ. Comput. (TEAC) 7 (2) (2019) 1–27.
[45] J. Wilkes, More Google cluster data, Google research blog, 2011, Posted at http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html.
[46] M.A. Sedney, Comments on median split procedures for scoring androgyny measures, Sex Roles 7 (2) (1981) 217–222.
[47] N. Matloff, Introduction to Discrete-Event Simulation and the Simpy Language Davis, CA, Dept of Computer Science. University of California At Davis, 2008, pp. 1–33, Retrieved on August 2 (2009).
[48] E. Angelelli, N. Bianchessi, C. Filippi, Optimal interval scheduling with a resource constraint, Comput. Oper. Res. 51 (2014) 268–281.
[49] H. Zhou, G. Bai, S. Deng, Optimal interval scheduling with nonidentical given machines, Cluster Comput. 22 (3) (2019) 1007–1015.
[50] Q. Wu, J.-K. Hao, A clique-based exact method for optimal winner determination in combinatorial auctions, Inform. Sci. 334 (2016) 103–121.
[51] M.M. Nejad, L. Mashayekhy, D. Grosu, Truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds, IEEE Trans. Parallel Distrib. Syst. 26 (2) (2014) 594–603.
[52] L. Mashayekhy, M.M. Nejad, D. Grosu, A PTAS mechanism for provisioning and allocation of heterogeneous cloud resources, IEEE Trans. Parallel Distrib. Syst. 26 (9) (2014) 2386–2399.
[53] X. Liu, W. Li, X. Zhang, Strategy-proof mechanism for provisioning and allocation virtual machines in heterogeneous clouds, IEEE Trans. Parallel Distrib. Syst. 29 (7) (2017) 1650–1663.
[54] J. Zhang, N. Xie, W. LI, K. Yue, X. Zhang, Truthful multi requirements auction mechanism for virtual resource allocation of cloud computing, J. Electron. Inf. Technol. 40 (1) (2018) 25–34.
[55] D.C. Parkes, Q. Duong, An ironing-based approach to adaptive online mechanism design in single-valued domains, in: AAAI, Vol. 7, 2007, pp. 94–101.
[56] F.T. Bruss, A unified approach to a class of best choice problems with an unknown number of options, Ann. Probab. (1984) 882–889.
[57] J. Zhang, X. Yang, N. Xie, X. Zhang, A.V. Vasilakos, W. Li, An online auction mechanism for time-varying multidimensional resource allocation in clouds, Future Gener. Comput. Syst. (2020).
[58] H. Mao, M. Alizadeh, I. Menache, S. Kandula, Resource management with deep reinforcement learning, in: Proceedings of the 15th ACM Workshop on Hot Topics in Networks, 2016, pp. 50–56.
[59] M. Babaioff, N. Immorlica, D. Kempe, R. Kleinberg, Online auctions and generalized secretary problems, ACM SIGecom Exch. 7 (2) (2008) 1–11.
[60] B. Du, C. Wu, Z. Huang, Learning resource allocation and pricing for cloud profit maximization, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 7570–7577.

Pankaj Prakash Mishra is currently pursuing a Joint PhD in Computer Science from Nagoya Institute of Technology, Japan and University of Wollongong, Australia. He also did his master's in computer science from Nagoya Institute of Technology, under the supervision of Professor Takayuki Ito. His major research interests include multi-agent systems reinforcement learning, mechanism design, auction theory, and Image Processing.

Dr. Ahmed Moustafa is Associate Professor in the Computer Science department at Nagoya Institute of Technology, Japan. He successfully completed his PhD in computer science from the University of Wollongong, Australia. His research interests include self-organising multi-agent systems, reinforcement learning and their applications.