Journal of Intelligent & Fuzzy Systems 46 (2024) 6851–6865
DOI:10.3233/JIFS-236043
IOS Press

Dynamic pricing on maximum concurrency for heterogeneous instances using hyperparameter optimization in dueling deep reinforcement learning in a multi-cloud scenario

Rajesh Arivanandhan[a,*], Kalaivani Ramanathan[b] and Senthilkumar Chellamuthu[c]
[a]*Department of Computer Science and Engineering, Erode Sengunthar Engineering College, Perundurai, India*
[b]*Department of Electronics and Communication Engineering, Erode Sengunthar Engineering College, Perundurai, India*
[c]*Department of Computer Science and Engineering, Erode Sengunthar Engineering College, Perundurai, India*

Abstract. Users possess the option to rent instances of various sorts, in a variety of regions, and a variety of availability zones, thanks to cloud service carriers like AWS, GCP, and Azure. In the cloud business right now, fixed price models are king when it comes to pricing. However, as the diversity of cloud providers and users grows, this approach is unable to accurately reflect the market's current needs for cost savings. As a consequence, a dynamic pricing strategy has become a desirable tactic to better handle the erratic cloud demand. In this study, a deep learning model was used to propose a dynamic pricing structure that ensures service providers are treated fairly in a multi-cloud context. The computational optimization of DL approaches can be severely hampered by the requirement for human hyperparameter selection. Traditional automated solutions to this issue have inadequate durability or fail in specific circumstances. To choose the hyper-parameters in the Dueling Deep Q-Network (DDQN), the hybrid DL approach in this study uses the concept-based wild horse optimization (WHO) method. A community of untamed horses is evolved, and the fitness of the population is evaluated concurrently to estimate the optimum hyper-parameters. The plan changes the price appropriately to promote the use of underutilized resources and discourage the use of overutilized resources. The evaluation's findings demonstrated that the suggested strategy can lower end-user costs while conducting compute- and data-intensive activities in a multi-cloud environment. The research was concluded by comparing current models after the results were analyzed using various performance indicators.

Keywords: Cloud providers, dynamic pricing scheme, Deep Learning, hyper-parameter selection, Oppositional-Based Learning, Wild Horse Optimization and Dueling Deep Q-Network

1. Introduction

The number of IT firms that provide network access to a "Cloud" has increased recently. Examples include Salesforce, Amazon EC2, Google App Engine, Windows Azure, and Amazon Web Services. "A model for assisting ubiquitous, convenient,

*Corresponding author. Rajesh Arivanandhan, Department of Computer Science and Engineering, Erode Sengunthar Engineering College, Perundurai – 638057, India. E-mail: rajesh46347334@gmail.com.

ISSN 1064-1246/$35.00

on-demand network of things connection to a pool of shared configurable computing assets (networks, servers, applications, data storage, and services) which can be swiftly allocated and screened with little to no management or cloud provider interaction," according to the definition given by the cloud [1]. It is commonly known that cloud computing has benefits [2]. These advantages entice companies to move their current IT workload to the cloud, which may have a favorable and economically significant impact on manufacturing output and worker productivity [3]. For instance, to host their applications and media, CloudPrime and Foursquare utilize Amazon EC2, while Harvard Medical School and Yelp employ high-performance computing from the same provider. Dell and AVON both use the Salesforce customer relationship management tool to grow their companies.

A 2010 poll of companies using cloud computing found that 74% of companies agree that "the internet has helped to alleviate internal IT resource pressures," 72% of companies say their end-user experiences have improved, and 73% of companies have been able to lower their infrastructure expenses [4]. The Department of the Treasury's Office of Budget and Management (OMB) stated that "government is currently running under a cloud-first policy" in December 2010, which mandates that agencies incorporate cloud-based technology into projects first and foremost [5]. The global market for cloud computing will increase from $40.7 billion in 2011 to $241 billion in 2020, predicts Forrester Research [6]. Cloud computing enables service providers to measure consumption for billing purposes while allowing consumers to pay for usage. The global market for cloud computing will increase from $40.7 billion in 2011 to $241 billion in 2020, predicts Forrester Research [6]. Cloud computing enables service providers to measure consumption for billing purposes while allowing consumers to pay for usage.

It is difficult to create a resource allocation method for cloud computing due to the rise of virtual distributed resources. The issues with resource allocation in the cloud computing environment are dynamic as a result of the heterogeneous character of cloud computing regarding applications and resources [7–9]. Resource sharing, poor usage, resource waste, and income loss have all been caused by the static allocation policies. Recent studies boost the capacity and dependability of a multi-cloud setup by enabling end users to choose assets from various service providers [10, 11]. The users' main challenge, though, is deciding which service provider would best meet their needs and do it at the lowest possible cost. In a multi-cloud context, this study suggests a dynamic pricing structure that ensures service providers are treated fairly. The price is adjusted by the plan to promote the use of natural assets with low rates of use and to discourage the use of commodities with high rates of utilization. We emphasize the need to take into account the value metric for the fundamental job, which includes the real service and resource needs. This paper's main contributions are to:

- Putting forth a hybrid DL model with integrated cloud brokering and federation strategy for pricing that changes in a multi-cloud context.
- The suggested approach can recognize patterns present in actual pricing dynamics and make arbitration decisions that maximize long-term income.
- Examine the performance of the suggested plan's cost-effectiveness by analyzing experimental data on the dynamic valuing mechanism for resource allocation.

The structure of this essay is as follows: The associated works are discussed in Section 2. The proposed plan is thoroughly presented in Section 3. The overall architecture of the suggested scheme is described in Section 4. The procedures and findings of the evaluation are covered in Section 5. The study is wrapped up in the last section.

2. Related work

For IaaS cloud platforms, Mukhopadhyay & Tewari [12] created a dynamic pricing model that can determine the fluctuating cost of execution depending on changes in user requirements while also setting a lower bound on the fundamental price for a cloud service. The author has also developed an effective quantitative cost analysis model that takes into account every static as well as dynamic cost component conceivable to determine a reasonable execution cost. Additionally, to validate the outcomes, a novel algorithm was introduced and put into practice within a simulated version of the service architecture. Here, extensive simulations were used to compare the proposed model to existing models. The cost of calculation fluctuates erratically as a result of the users' changing demands. Therefore, preserving the trade-off between speed and computing cost has become essential.

The FPM (Flexible Pricing Mechanism) is a method used by resource providers to calculate TVB, according to Adabi et al., [13]. The aforementioned difficulty is even more difficult in widely recognized cloud markets wherein capacity-type instances are given with different QoS levels. This is because choosing an approach to negotiation based on the estimated TVB should prevent renting commodity-type instances that have reduced QoS at a price greater than renting the identical commodity-type instances with higher QoS. The goal of this research is to provide a flexible pricing method that supports price-quality relationships in cloud marketplaces wherever resource-type instances are offered with different QoS levels. According to the simulation results, the suggested negotiators outperform EMDA (Enhanced Market Driven Agent) and FNSSA (Fuzzy Negotiation Strategy Selection Agent), two additional negotiators. However, because Cloud resources are released and assigned dynamically, the suggested strategy is no longer as practical.

The adaptive scheduling technique known as the Dynamic price-dependent Sequential Auction allocation mechanism was presented by Kumar & Kartheeban [14]. Through dynamic pricing and the combinatorial auction, it will be used to improve resource utilization and consumer happiness. The suggested market-based scheduling algorithm aims to increase the contentment of Cloud suppliers and clients by utilizing the notion of an auction mechanism. This method reconstructs the current resource allocation preferences in order to allocate resources in advance for unexpected virtual machine needs. The results of the simulation experiment show that the suggested scheduling technique overall Collective-target enhancement numerical prototypes can effectively improve resource utilization, supplier profit, and quality of service (QoS). Even if cloud assets are assigned automatically and released, such a scheduling approach is ineffective for Cloud.

In an evolving market where the number of participating cloud users is changing, Shi et al.'s [15] analysis examined whether a bidding-based cloud service provider determines the auction price efficiently while contending against other cloud providers. Numerous variables, including its competitors' auction prices, the cost imposed to clients in the previous auction, how cloud users bid, and more, have an impact on the pricing approach. In order to create a competitive pricing strategy, we represent the issue as a minimally observable Markov game and use a gradient-based multi-agent deep learning technique. According to the experimental findings, the created pricing strategy can outperform other approaches to pricing in terms of long-term gains and the number of users who participate, and it can also successfully learn the marginal values of cloud users and their cloud provider preferences.

The Stackelberg game was used by Zhu et al. [16] to model the earnings maximization problem and examine the existence as well as the distinctiveness of the competitive equilibrium. Additionally, to increase the profitability generated by SaaS and IaaS providers, we further suggest an adaptive pricing mechanism that takes into account the influence of a resource price on users' willingness to access service. The simulation results show that the proposed mechanism is preferable in terms of maximization of revenue and resource usage when compared with conventional fixed-price and auction-based pricing systems. While SaaS companies want to reduce the cost of employing infrastructure resources while simultaneously adhering to service-level obligation contracts with customers, IaaS providers seek a suitable cost policy for virtual machines to increase revenue.

To benefit both parties, Sharma et al. [17] presented an online Computation Commodity (C3) pricing model termed Clabacus(Cloud-Abacus). Additionally, modify the computed resource price to take into account the inherent hazards of the Cloud provider utilizing financial value-at-risk (VaR) analysis. Additionally, provides ways based on fuzzy logic and evolutionary algorithms to calculate the Value-at-Risk (VaR) based on the provider's resources. Additionally, we have added this feature to the Clabacus architecture. Finally, it will analyze the implications of the caliber of assistance, rate of depreciation rates of inflation, and capital expenditure on the online access pricing for both client and supplier. Additionally, demonstrates that SLA may be ensured if price adjustments are made between the reduced and upper bound.

Cong et al. [18] created a dynamic cloud pricing method based on reinforcement learning (RL) to maximize both the profit and cost of the cloud provider for diverse consumers with different personalities. To effectively incorporate the dynamics of users' perceived values concerning cloud services, we first present a unique personality-guided user-perceived appraisal prediction scheme. In the cloud computing market, the prediction system simulates the relationship between user personality traits, performance price, of service (QoS), user pleasure, and

perceived value. Second, an RL-based cloud pricing mechanism is created based on the prediction model to teach sequential service price decision-making for cost and profit optimization. In particular, a discrete-time systems Markov decision process (MDP) is used to represent and solve the profit and cost optimization problem. Finally, in-depth simulation studies have been carried out to validate our model for predicting user-perceived value and our pricing model for cloud services. Although somewhat deterministic, these web pricing models and methods could not function well in a real-world setting because users' perceptions of the value of cloud services are constantly shifting and highly customized.

Cong et al.'s [19] dynamic pricing model, which is based on the idea of user-perceived value, successfully represents the actual supply and demand dynamics in the market for cloud services. The dynamic pricing model is then used to construct a profit-maximizing scheme that maximizes the cloud service provider's profits while abiding by the service-level agreement. The cost of cloud services and multiserver configurations are finally adjusted by the fluctuations of the cloud computing setting, such as changing electricity and rental costs, using a dynamic closed-loop control method. The efficiency of the suggested user observed based on pricing strategy along the constantly changing profit maximization scheme has been validated by extensive simulations utilizing the data retrieved from real-world applications.

To increase cloud revenue, Alzhouri et al. [20] examined dynamic pricing of idle resources. Our suggested method specifically handles multiple kinds of virtual machines to provide the highest predicted revenue within a limited discrete time frame. To achieve this, the suggested method makes use of Markov chain reactions with a variety of attributes that, when combined with ideal controlling conditions, define the behavior of a model. Additionally, this method uses linear programming for approximation of stochastic dynamic programming in order to produce a useful model. Experiment results demonstrate that this dynamic pricing strategy may scale prices up or down effectively and economically in response to load thresholds and resource stagnation. These findings offer crucial information for increasing IaaS cloud income. This research on current spot pricing methods demonstrates, however, that these schemes use synthetic pricing policies that don't take into account the dynamic character of these events.

To swiftly and consequently decide on federation across various service provider domains, Martn-Pérez et al. [21] put forward an evolving cost and revenue-driven assistance association tactics based on a Deep Q-Network (DQN). Each domain presents dynamic service price offerings to its customers and other domains. In this study, an agile arrival process as a result of price changes is provided for expressing a service confederation procedure as a Markov Decision Problem (MDP). This process is based on the examination of real pricing data obtained from public cloud providers. The problem is solved using a variety of reinforcement learning algorithms in this work, and the results show that the DQN method surpassed existing state-of-the-art strategies and reached 90% of the optimal revenue. It also can learn the dynamics of federation pricing so that it can make the best decisions for the federation in response to price changes.

Inference: In conclusion, to the greatest extent of our knowledge, real pricing dynamics have not yet been taken into account in the literature when determining the ideal domain to distribute services. To achieve this, additionally rely on data-driven, model-free strategies based on optimal deep learning, which, to maximize long-term income, identifies correlations in information while generating any deductive inferences about the system.

3. Proposed methodology

This study analyses a public cloud provider's price fluctuations and uses them as a benchmark for multi-cloud service pricing. Additionally, by taking into account the noted price variations, can increase their market share in the multi-cloud market. Define the multi-domain context as a web-based decision-making problem to maximize revenue, and then develop an adaptive arrival process that is affected by changes in the service prices. The OBL-WHO approach is used in this work's design and implementation of the hyperparameter optimization method in combat with a deep reinforcement learning system, as shown in Fig. 1. Both require a training phase to determine a strategy for revenue maximization, as well as to thoroughly evaluate the solution's performance using data and compare it to cutting-edge approaches. The aim is to increase the service provider's long-term revenue as much as possible. Intuitively, lower charges encourage a higher customer arrival rate. As a result, the pricing structure does affect how requests for the service arrive.

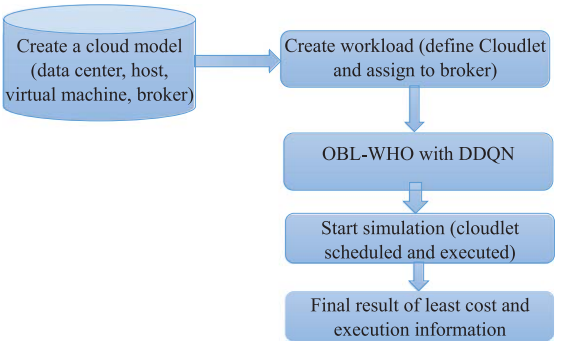

Fig. 1. SmartArt of proposed methodology.

3.1. System architecture

The proposed system structure for the suggested multi-clout environment is shown in Fig. 2. Diverse vendors of services offer the resources by their price structures, as depicted by the architecture. By providing the criteria to the broker, the end user can submit task orders amongst different service providers in the market for cloud services. The broker serves as a go-between for the client as well as the service providers. The broker has adopted the suggested dynamic pricing mechanism for resource allocation. The end users can see how resources are allocated in real-time. The broker is in charge of choosing the service provider with the lowest end-user cost.

The CloudSim simulator is used as the foundation for the suggested dynamic pricing mechanism [22]. The CloudSim simulator covers the modeling of cloud economic entities, including service providers, brokers, and end users. The simulator also supports a multi-provider Cloud environment with resource management and scheduling. The interaction involving the product or service provider (DataCenter) as well as the broker (DataCenterBroker) is depicted in Fig. 3. To match user/broker requests to the best service provider, the Center for Internet Security (CIS) organization offers matchmaking services.

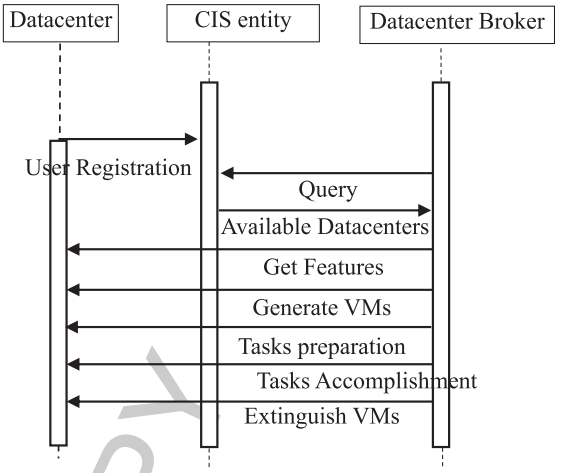

Fig. 3. Information interchange between service provider and broker.

Our proposed plan is added to the DataCenterBroker object in this study. The improved broker can manage resource allocation, task scheduling, and the current condition of resource use. By the resource usage state, the broker predicts the execution times of each request performed in every provider of services when it gets the task requests. The broker can then assign the undertaking to the supplier with the lowest price.

3.2. Dynamic pricing scheme

The suggested plan supports resources that be shared over both time and space. A time-shared resource will be used to conduct a job if it calls for a lot of computation [23]. Processing power can be

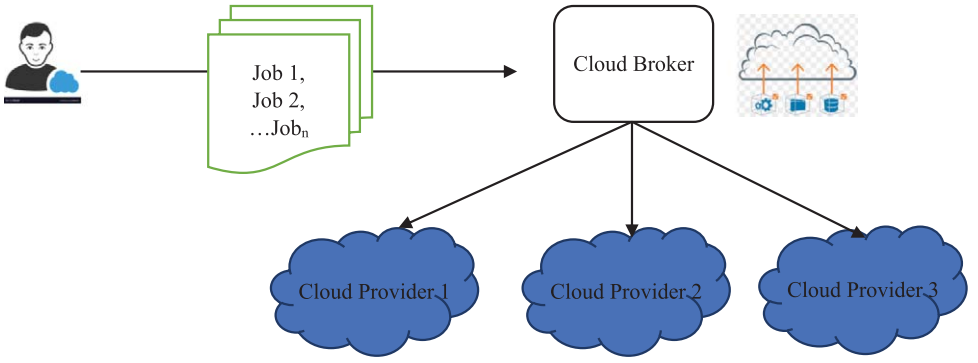

Fig. 2. System architecture of the proposed multi-clout environment.

distributed across numerous cores while still supporting multithreading thanks to time-shared resources. However, if a request is a data-intensive activity, it can be handled in the space-shared capacity. Memory space, which could be memory or storage, can be shared via space-shared resources. The plan put forward is mostly dependent on the use of this resource pool. The momentary demand in each pool is determined by resource consumption. As the amount of resource use rises, so does the price. This method will impact whether or not end users switch from constrained pools of resources to other pools. The provider of services establishes the constant price for each unit of a resource, which is referred to as the fundamental B price *Bprice*. The overall commodity unit in a data center DC_i is. RT_i, whereas the sum of the free resources in a center DC_i is RF_i. The variable price *Dprice* is calculated using the subsequent Equation (1):

$$Dprice = \frac{RT_i}{RF_i} \times Bprice_i \tag{1}$$

According to [24], processors with low usage may waste power when it is not being used. Similar to this, workstations are unlikely to meet the necessary throughput and may face delays as the normal workload gets close to 100% [25]. As a result, the proposed design's possible spectrum for acceptable utilization levels falls between 30% and 70%. The objective is to encourage the use of resources with low utility while restricting the use of resources with high utility. A rate of discount is offered in the suggested system to encourage the productive use of low-utilization resources. On the other side, a surcharge is applied to dissuade the use of resources with excessive consumption. Let us define $\mathfrak{U}$ at a fair level as being between $\mathfrak{U}^{low}$ and $\mathfrak{U}^{high}$. The following represents the recommended costing strategy with discount factor *DFprice*:

$$DFprice_i = \begin{cases} Bprice_i - discount & \mathfrak{U}_i < \mathfrak{U}_i^{low} \\ Bprice_i & \mathfrak{U}_i^{low}\ \mathfrak{U}_i \le \mathfrak{U}_i^{high} \\ Bprice_i + extra & \mathfrak{U}_i > \mathfrak{U}_i^{high} \end{cases} \tag{2}$$

The reduction in pricing function $DFBprice_i$ is made up of two parts: the constant element foundation pricing rate $Bprice_i$ and the dynamic portion that uses the discount factor. The following function of polynomials is used to compute $PF(\mathfrak{U})$:

$$PF_i(\mathfrak{U}_i) = \sqrt{Bprice_i * 10(\mathfrak{U}_i - 0.5)^3}$$

$$DFprice_i = Bprice_i + PF_i(\mathfrak{U}_i) \tag{3}$$

The suggested scheme's goal is to deliver end customers with the most affordable price for the material consumed. To achieve the goal and be fair to the service providers, the placement of resources strategy must use the expenditure component as the main factor. To determine the lowest-cost resources *cr*, assess the expense of maintaining an operation on each resource. Let $\approx_{ij}$ represent the completion period for job j on resource i. The execution time is estimated by:

$$\approx_{ij} = \frac{TI_j}{TC_i \times cr_i} \tag{4}$$

TI signifies the overall amount of j instructions performed within i. A compute resource's total capacity TC is measured in millions of commands per minute (mips) per a virtual machine. To calculate the expense $\mathfrak{E}$ for operating an assignment on the available resources.

$$\mathfrak{E}_{ij} = \approx_{ij} * price(i) \tag{5}$$

Where price (i) is the pricing system used by resource i. The suggested dynamic pricing scheme's pseudocode. For each interval, the work requests are sorted by the longest job first (LJF). The consumption rate in each center is determined to calculate the $price_i$, which is then utilized to determine $\mathfrak{E}_{ji}$ for each $\mathfrak{E}_{ji}$, and every expenditure is compared to obtain the DC_i with the lowest level of $\mathfrak{E}_{ij}$. Ultimately, the request for employment will be assigned to resources on the chosen DC_i. The operation keeps going until all requests have been fulfilled, as specified in Algorithm 1.

Algorithm 1. Pseudocode of dynamic pricing scheme

Input: The user submits a task request j, and DC_i lists the data centers in the market for cloud services along with the resources that are available and their associated costs.
Output: An anticipated job cost timeline $\mathfrak{E}_{ij}$ of j using resource r at DC_i
For each resource
  Calculate $Dprice = \frac{RT_i}{RF_i} \times Bprice_i$
  If ($\mathfrak{U}_i < \mathfrak{U}_i^{low}$)
    return $Bprice_i - discount$
  Else if($\mathfrak{U}_i^{low} \le \mathfrak{U}_i \le \mathfrak{U}_i^{high}$)
    return $Bprice_i$
  Else:
    return $Bprice_i$ + extra
  End if
  Calculate Pricing Factor
  $PF_i(\mathfrak{U}_i) = \sqrt{Bprice_i * 10(\mathfrak{U}_i - 0.5)^3}$
  Estimate Completion Time $DFprice_i = Bprice_i + PF_i(\mathfrak{U}_i)$

```
Allocate Resources (jobs, centers):
for each job in jobs:
  Sort Jobs By Length (jobs)
  for each center in centers
      ≈ij = TIj / (TCi × cri)
    pricei = resource.getPrice()
      Eij = ≈ij * price (i)
  End for
End for
If (Eij < lowest_cost)
  lowest_cost=Eij
  selected_center = center
  selected_resource = resource
End if
End for
Return selected_center.assign Job(job, selected_resource)
```

3.3. Dynamic pricing detection Using Hybrid IBO-WHO with DDQN Model

Detection in dynamic pricing refers to the process of identifying optimal pricing strategies or patterns within a changing market environment. It involves recognizing pricing variations, trends, or opportunities to adapt and optimize pricing models for improved profitability or performance. To attain optimal inventory and income, a general concept for an autonomous adaptive presale environment is provided. The dynamic price problem is solved using a DDQN algorithm, effectively enhancing the retailer's long-term profitability, in a limited presale horizon. The algorithm utilized in this research can learn pricing methods more effectively than the current reinforcement learning algorithms, according to experiments. There are currently only a few papers in the literature that tackle the dynamic pricing problem in presale mode using the notion of deep reinforcement learning. As a result, this represents an innovation to enhance the multi-period dynamic price strategy in this article's presale mode. To train and assess the DDQN dynamic price algorithm's effectiveness and demonstrate that the model is broadly applicable in a market context with uncertain demand, a larger multi-cloud computing simulation environment is created. It is investigated how decisions are made and profits are affected by various stocking prices and cost deviation coefficients, and recommendations for merchants' marketing or service approaches are made. In this case, the IBO-WHO approach is used to fine-tune the hyperparameters of the DDQN. The hyperparameters are $N = 10{,}000$, which stands for Experience replay; $B = 32$, which stands for Batch size; $l = 0.001$, which stands for Learning rate; $\Upsilon = 0.95$, which stands for Discount factor; ε_{init} 1.0, which stands for Initial exploration rate; ε_{end} 0.01, which stands for Final exploration rate; and ε_{decay} 100, which stands for Decay factor.

3.3.1. Q-learning method

By building a structure to record the Q-value under various states and actions and choosing the best action, the value-based reinforcement learning method known as Q-learning is utilized to maximize reward [26]. The predicted benefit of carrying out an ac_tion act by strategy while the state is s_{et} is represented by the current action value function Qπ (st, ac).

$$\mathcal{Q}_\pi(st, ac) = E[\sum_{t'=t}^{T} \Upsilon^{t'-t} r_{t'} | st_t = st, ac_t = ac, \pi] \tag{6}$$

By Equation (6), Bellman's existence optimal equation is followed by the state of the action value function for the best approach. The following is the optimal value function:

$$\mathcal{Q}^*(st, ac) = \sum_{st' \sim S} P\left(st' | st, ac\right) \left[r + \Upsilon \max_{ac'} \mathcal{Q}^*\left(st', ac'\right)\right] \tag{7}$$

However, due to the huge dimensions of an issue or activity in a real situation, calculations may become challenging and time-consuming. Every action taken in each state is recorded with the Q-value by the Q-learning algorithm. Since the amount of space containing states and actions is large, it requires an extended period to estimate the Q-table, and it is difficult to approach convergence because a large Q-value table must be constructed. Mnih et al. [27] suggested using a deep neural network to calculate $\mathcal{Q}(st, ac)$, to address the problem of dimensional disaster. Deep learning and reinforcement learning are both used in the DQN method. The neural network model is introduced to the Q-learning method to calculate the Q-value.

To eliminate the association between the data, the DQN algorithm also uses the procedure of replayed technique to store the data the agent experiences within the playback memory and extract a small batch of information through it during each update. To increase the precision of value function estimation, the DDQN [28] methodology utilizing a dual-network structure separated the Q-network into two parts: the value functions $VF(st, ac, m)$ and the

advantage function $AF(st, ac, \omega, l)$, as indicated in Equation (8). To increase the stability of the algorithm, as indicated in Equation (9), we centralize the Q-value calculation, make sure that the order of magnitude among all dominant functions stays the same in every state, and eliminate unnecessary degrees of freedom.

$$\mathcal{Q}(st, ac, \omega, m, l) = VF(st, \omega, m) + AF(st, ac, \omega, l) \tag{8}$$

$$\mathcal{Q}(st, ac, \omega, m, l) = VF(st, \omega, l) + \left[AF(st, ac, \omega, l) - \frac{1}{|AF|} \sum_{ac \in AF} AF(st, ac, \omega, l) \right] \tag{9}$$

In this work, the evaluation and target networks are constructed using the same DDQN technique. Equation (10), shows the computation formula for the target network. To keep the algorithm training stable, the target network is fixed during the updating process. The weighted average of the assessment structure is transmitted onto the target network after a predetermined number of repetitions, which lowers the association among the forecast Q-value and overall target Q-value, eliminates the likelihood of the loss value diverging during training, and increases the learning's stability. According to Equation (12), the average squared error (MSE) is adopted by the DQN loss function to reduce the difference between the target and forecast Q-values. The MSE is a typical regression estimation error-measurement technique used in machine learning to calculate the degree of discrepancy between the model's true value and its predicted value. Using backpropagation, the DQN algorithm trains the network of neurons according to the loss function in order to reduce the measurement discrepancy associated with the forecasted Q-value and the goal Q-value.

$$Target\mathcal{Q} = r + \gamma max_{ac} \mathcal{Q}\left(st', ac'; \theta\right) \tag{10}$$

$$L(\theta) = E[\left(Target\mathcal{Q} - \mathcal{Q}\left(st', ac'; \theta\right)\right)^2 \tag{11}$$

This algorithm chooses its course of action ε using the greedy technique, balancing exploration and utilization to maximize returns. Equation (12) illustrates the exponential attenuation of the exploration rate value and the usage of decay to regulate the attenuation rate. Algorithm 2 displays the DDQN algorithm's pseudo-code.

$$\varepsilon = \varepsilon_{end} + (\varepsilon_{init} - \varepsilon_{end}) exp\left(-\frac{step}{\varepsilon_{decay}}\right) \tag{12}$$

3.4. The DDQN architecture

The battling network, which we create here, is a single Q-network architecture. The dueling network's lower layers are convolutional, just like in the initial DQNs [19]. However, we use two separate chains (or streams) of layers that are completely linked as opposed to one pattern of entirely connected layers to follow the convolutional layers. The streams are designed in a way that allows them to deliver independent estimations of worth and advantage functions. An output $\mathcal{Q}$ function is created by combining the two streams in the end. The network produces an array of $\mathcal{Q}$ standards, one for each action, as in [19].

The DDQN network can be trained using any of the numerous known techniques, including DQN and reinforcement learning because its output is a Q function. Additionally, it can benefit from any advancements made to these algorithms, such as enhanced replay memory, improved exploration rules, intrinsic motivation, etc. It takes very careful design to create an element that brings together both streams of entirely interconnected levels to generate a Q estimate. It follows that $E_{ac\sim\pi(st)}\left[AF^{\pi}(st, ac)\right] = 0$. from the formulae for advantage $\mathcal{Q}^{\pi}(st, ac) = V^{\pi}(st) + A^{\pi}(st, ac)$ and state-value $V^{\pi}(st) = E_{ac\sim\pi(st)}\left[\mathcal{Q}^{\pi}(st, ac)\right]$. Furthermore, it follows that $\mathcal{Q}(st, ac^*) = VF(st)$ in a deterministic policy, where, $ac^* = \arg\max_{ac' \in AF} \mathcal{Q}\left(s, ac'\right)$, and $AF(st, ac^*) = 0$. Let's have a look at the dueling network where we output two streams of completely connected layers, one of which produces a scalar $VF(st; \theta, \beta)$, and the other of which produces a $|AF|-$ dimensional $AF(st, ac; \theta, \alpha)$.

Here, the terms "convolutional layers" and "convolutional streams" refer to the parameters that define the two channels of fully connected layers, respectively. We might be enticed to build the accumulating module as follows using the concept of advantage:

$$\mathcal{Q}(st, ac; \theta, \alpha, \beta) = VF(st; \theta, \beta) + AF(st, ac; \theta, \alpha), \tag{13}$$

Remember that this expression holds for all (s, a) instances, thus we must replicate the scalar |AF| in order to represent Equation (13), in matrix form. $\mathcal{Q}(st, ac; \theta, \alpha, \beta)$ is merely a parametric estimate of the genuine Q-function, though, so we must keep that

in mind. Furthermore, it would be incorrect to claim that $VF(st;\theta,\beta)$ offers a reliable estimate regarding the state-value function or that $AF(st,ac;\theta,\alpha)$ does the same for the advantage function. Equation (7) is unidentified in that, given Q, we are unable to retrieve V and A singularly. This can be demonstrated by adding a constant to V (s; θ, β) and deducting the identical constant from $AF(st,ac;\theta,\alpha)$. This constant cancels out, leaving the Q value unchanged. When this equation is applied directly, it performs poorly in practice, mirroring its lack of identifiability. We can make the benefits of the function estimator have no advantage at the selected action to solve the identifiability problem. In other words, we let the network's final module implement the forward mapping.

$$\mathcal{Q}(st,ac;\theta,\alpha,\beta) = VF(st;\theta,\beta) + \left(AF(st,ac;\theta,\alpha) - \max_{ac'\in|A|} AF\left(st,ac';\theta,\alpha\right)\right). \tag{14}$$

Now, for $ac^* = \arg\max_{ac'\in AF} \mathcal{Q}\left(st,ac';\theta,\alpha,\beta\right) = \arg\max_{ac'\in AF} AF\left(st,ac';\theta,\alpha\right)$, we obtain $\mathcal{Q}(st,ac^*;\theta,\alpha,\beta) = VF(st;\theta,\beta)$.

As a result, whereas the other stream generates a prediction of the advantage function, the stream $VF(st;\theta,\beta)$ offers an approximation of the value function. An alternative module uses an average instead of the max operator:

$$\mathcal{Q}(st,ac;\theta,\alpha,\beta) = VF(st;\theta,\beta) + \left(AF(st,ac;\theta,\alpha) - \frac{1}{|AF|}\sum_{ac'} AF\left(st,ac';\theta,\alpha\right)\right). \tag{15}$$

The original meaning of V and A is lost since they have been moved inaccurately by an integer, but on the plus side, the optimization is more stable because in (9) the positive aspects are only required to change as quickly as the mean, as opposed to in (8) where they must compensate for any change in the optimal action's advantage. We also tested a soft maximum version of Equation 8, but we discovered that it produced outcomes that were comparable to those of the more straightforward module of Equation 9. Therefore, the component of Equation (9), is used in all the experiments given in this paper. While removing the mean from Equation (9) improves identifiability, it has the unfavorable effect of maintaining any greedy or non-greedy strategy based on Equation (7)'s Q values because it does not alter the order of magnitude among the A (as well as hence Q) values. When acting, assessing the advantage stream is sufficient to guide judgments. It's vital to notice that the formula (9) is treated as a component of the network rather than as a distinct algorithmic step and is therefore implemented as such. As with regular $\mathcal{Q}$ networks (such as the extensive U-network of Mnih et al. (2015)), training dueling architectures just requires back-propagation. The estimates $VF(st;\theta,\beta)$ and $AF(st,ac;\theta,\alpha)$ are generated automatically without the need for additional oversight or algorithmic adjustments.

3.4.1. Federation based DDQN

We can reuse any learning techniques using Q networks (like DQN and SARSA) to learn the dueling architecture because it has the identical input-output interface as traditional Q networks. We begin by reviewing the business model that is important to our project. In this study, we investigate a system where a service provider provides cloud services or resources at a service cost rate as $spr^{(t)}$ that may change as time passes depending on the operator's pricing model. This system is inspired by the market for cloud services.

When a user decides to deploy a service for such a fee, it submits a request, which enters the network at time $ar(\sigma)$, and exits at time $de(\sigma)$. After receiving a user request, the service provider chooses where the service should be deployed: either inside its infrastructure or across an additional domain throughout the federation. As a result, the service provider has the option to respond to each service deployment request from a user by taking an action $x(\sigma) := \{0, 1, 2\}$ that indicates either the product has been installed regionally, distributed in the united domain, or denied. Note that neither the user nor the federation is aware of the resources that are accessible in the infrastructure of the service provider. Users only know the price that has been offered for their request, or $spr^{(t)}(\sigma)$. Regarding an overview of our notation, see Table I.

Maximizing the service provider's long-term revenue is what we aim to achieve. The pricing structure does affect how quickly service requests are received; hence, lower rates encourage quicker user arrival. However, after a contract between a client and a provider has been reached, it is crucial to note that the consumer must pay the agreed-upon amount or $spr^{ar(\sigma)}$ for each time slot ts that the facility is active, or for each $ts : ar(\sigma) \le ts \le de(\sigma)$. The service will be refused if it is not installed regionally or in the federated domain. In this scenario, the customer will not pay the service charge, leaving the service provider with no money. This business approach enables us to

take advantage of pricing swings that are opportunistically (uncertain), which can result in significant cost savings, while still giving end users stability, which is crucial for vertical sectors. We consequently have two parallel cash flows for each t: The request is granted by the service provider using local resources, therefore the agent's remuneration is equal to (Fig. 1a).

$$spr^{ar(\sigma)}(\sigma), \forall ts : ar(\sigma) \leq ts \leq de(\sigma); \quad (16)$$

The application supplier uses federated resources, hence the provider receives $spr^{ar(\sigma)}(\sigma) - fc^{(t)}(\sigma)$, where $fc^{(t)}(\sigma)$ is the variable federation cost.

Denote the agent's income, which at time ts represents the service provider's immediate revenue, as follows:

$$ir^{(ts)}(\mathcal{X}_{ts}) := \sum_{\substack{\sigma : x(\sigma) = 0 \\ ar(\sigma) \leq ts \leq de(\sigma)}} spr^{ar(\sigma)}(\sigma) + \sum_{\substack{\sigma : x(\sigma) = 1 \\ ar(\sigma) \leq ts \leq de(\sigma)}} \left[spr^{ar(\sigma)} - fc^{(t)}(\sigma)\right] \quad (17)$$

where $\mathcal{X}_{ts} := \{x(\sigma)\}_{\sigma : ar(\sigma) \leq ts}$.

If the service provider exhausts all of its local assets, its agent may federate the service $x(\sigma) = 1$ at a cost to the services provider $f(t)(\sigma)$ in (17). Therefore, the accessibility of resources affects the service provider's immediate revenue.

Algorithm 2: Pseudo-code of the DDQN algorithm for dynamic pricing problem in multi-cloud

Input: Parameters of the cloud environment: c, c_i, p_0, k_0, δ, T; parameters of the Dueling DQN-DP algorithm: α, B, γ, ζ, ε_{init}, ε_{end}, ε_{decay}

Output: The best pricing approach $\pi*$

Set replay of experience memory D to its maximum size N.
Initialize the weights of the Q-network θ
Set up the target network weights θ'
For *episode* $= 1, M$ do
Environment reset and state activation $st_1 = (\lambda_1, k_1)$
For $t = 1, T$ do with probability ε select a random action ac_t
otherwise select $ac_t = argmax_{ac}(st_t, ac_t; \theta)$
Execute action at and observe reward r_t and st_{t+1}
Store transition $(st_t, ac_t r_t, st_{t+1})$ in D
Set $st_{t+1} = st_t$
Sample random minibatch B of transitions $(st_t, ac_t r_t, st_{t+1})$ from D

$$Set\ y_i = \begin{cases} r_j & if\ episode\ terminates\ at\ step\ j+1 \\ r_j + \gamma max_{ac'} \mathcal{Q}\left(st_{t=1}, ac'; \right) & otherwise \end{cases}$$

Calculate the federation cost as well using $ir^{(ts)}(\mathcal{X}_{ts})$
To develop the error function LF about the network parameters, use gradient descent.
θ as $LF(\theta) = E\left[(Target\mathcal{Q} - \mathcal{Q}(st, ac; \theta))^2\right] * ir^{(ts)}(\mathcal{X}_{ts})$
Every ξ step updates the target network parameters
$\theta^- \leftarrow \theta$
End For
End For

3.5. MWHO-based hyperparameter optimization model

The accuracy of detection of the DDQN algorithm is optimized by using the MWHO algorithm as a kind of hyperparameter tuning technique [29]. The WHO strategy relied on the traits of wild horses' social behavior. They generally reside in herds of stallions, mares, and young horses [29]. They have shown off a variety of traits, including dominating, pursuing, mating, and grazing. The WHO's main procedure is explained in the following manner. The initial population is split up into several groups. The stallion S is the leader of each group, while the rest members—mares and foals—are evenly dispersed. The following factors affect the grazing nature:

$$\mathcal{X}^j_{i,G} = 2Zcos(2\pi RZ) \times \left(\mathcal{S}^j - \mathcal{X}^j_{i,G}\right) + \mathcal{S}^j \quad (18)$$

The location of the existing foal is indicated by $\mathcal{X}^j_{i,G}$ in the expression. A constant stochastic number is specified by R, and Z represents the adaptive process described below.

$$P = \rightarrow \vec{R}_1 < TDR; IDX = (P = 0);$$

$$Z = R_2 \Theta IDX + \rightarrow \vec{R}_3 \Theta (\sim IDX) \quad (19)$$

Now that P denotes a vector with a range of 0 to 1, $\vec{R}_1$ and R2 display the configurable integer ranges between 0 and 1 and define values within [0, 1]:

$$TDR = 1 - it \times \left(\frac{1}{maxit}\right) \quad (20)$$

The highest possible iteration count is now displayed by maxit. The foal pushes from i swarming into the impermanent group to develop the horse's mating characteristics, whereas the young horse originates from the j swarm to the temporary group:

$$\mathcal{X}^P_{G,K} = Crossover\left(\mathcal{X}^q_{G,i}, \mathcal{X}^Z_{G,j}\right)$$
$$i \neq j \neq k,\ p = q = end, \quad (21)$$
$$Crossover = Mean$$

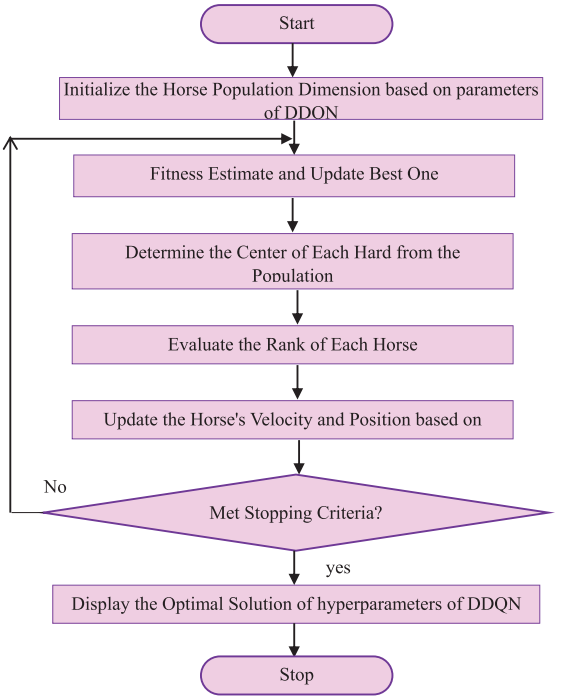

Fig. 4. Flowchart of WHO algorithm.

In this piece, the Mane conducts the swarm toward the water hole where they fight for control of it. The remaining group also takes advantage of the water hole, but mostly the dominant swarm does:

$$\overline{\mathcal{S}_{G_i}} = \begin{cases} 2Zcos\,(2\pi RZ) \times (WH - \mathcal{S}_{Gi}) \\ \quad -WH\ if\ R_3 \le 0.5 \\ 2Zcos\,(2\pi RZ) \times (WVH - \mathcal{S}_{Gi}) \\ \quad -WH\ if\ R3 \le 0.5 \end{cases} \tag{22}$$

The leader's next location is indicated by $\mathcal{S}_{G_i}$, and the water hole's location is indicated by WH. The leader is then selected in the following ways based on the standards of fitness in the succeeding phases:

$$\overline{\mathcal{S}_{G_i}} = \begin{cases} if cost\left(\mathcal{X}_{G,i}\right) > \cos t\left(\mathcal{S}_{G_i}\right) \\ \left(\mathcal{S}_{G_i} if \cos t\left(\mathcal{X}_{G,i} > \cos t\left(\mathcal{S}_{G_i}\right)\right)\right) \end{cases} \tag{23}$$

The WHO algorithm's flowchart is depicted in Fig. 4. Utilizing an oppositional-based learning (OBL) idea, the MWHO algorithm is developed. The OBL technique offers a distinctive counter-solution to the existing one [30] and even attempts to establish a superior approach that promotes faster convergence. The real number's inverse $\mathcal{X}^0$, (X ∈ [U, L]) was evaluated by:

$$\mathcal{X}^0 = U + L - -\mathcal{X} \tag{24}$$

Opposite point: $\mathcal{X}^0$. Assume that $\mathcal{X} = [\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_{Dim}]$ designates a point in a search space of Dim dimensions, and that $\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_{Dim} \in R$ and $\mathcal{X}_j\left[U_j, L_j\right]$. Consequently, the opposite of X corresponds to follows:

$$\mathcal{X}_j^0 = UB_j + L_j - \mathcal{X}_j, where j = 1....D \tag{25}$$

The two data points ($\mathcal{X}_0$ and $\mathcal{X}$) that were the most useful were chosen based on the fitness function's values, and the third point was disregarded. If $(\mathcal{X})\ f\left(\mathcal{X}^0\right)$, $\mathcal{X}$ was sustained for minimizing the problem; else, $\mathcal{X}_0$ was sustained.

4. Experimental results and discussions

The evaluation step uses the computationally and data-intensive assignments as input. The SPEC CINT2006 suite [31] is used for compute-intensive tasks, while MalStone [32] is used for data-intensive tasks. Both datasets specify the length and number of jobs as a comparable measure of performance need based on practical applications. Several renowned public cloud service suppliers make up the multi-cloud environment [33, 34]: Test data included Amazon EC2, Google GoGrid, and Rackspace cloud service providers, with ten hosts each. Ten hosts with four processing elements (PE) make up each service provider. There are around 100 and 180 open positions. Approximately three thousand and 38000 million computations (MI) and 1800 and 4800 gigabytes (GB) of data make up a compute-intensive activity. The training data involved utilizing historical benchmark datasets [27, 28] to create models and algorithms for dynamic pricing strategies and resource allocation. To shorten the simulation duration, the number of resources such as memory and hosts is reduced. The experiment's time interval is five seconds. The intermediary analyzes the task proposal from the buffer every five seconds. Table 2 provides the specifics of each data center's resource information. The asset parameter is defined by the small digital instance given by Amazon EC2, GoGrid, and Rackspace, which are all public Cloud providers [35]. The cost structure has been lowered downwards to cost for second rather than cost per hour for simulation purposes, and the expense per memory is scaled up by doubling each expenditure by ten. The number of terabytes affects how much storage

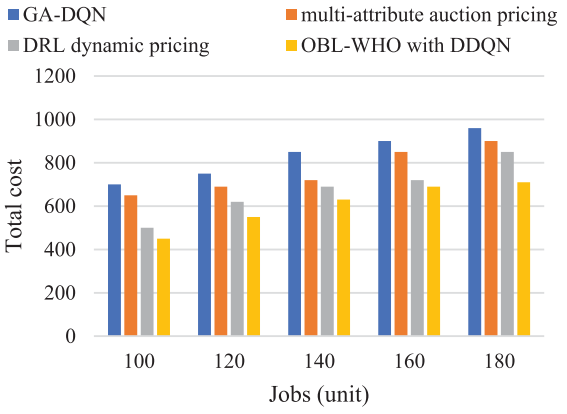

Fig. 5. The overall expense of executing computationally expensive tasks assuming dynamic pricing for resource reservations.

will cost. Given that the benchmark information utilized encompasses 1.8 to 4.8 terabytes, the cost per capacity goes upwards for every supplier since the monetary difference is too small to reveal.

Several studies are carried out to compare accomplishments in terms of the overall cost of job execution. The studies involve a dynamic pricing scheme with various resource allocation methods, such as GA-DQN, multi-attribute auction pricing, DRL dynamic pricing technique, and proposed OBL-WHO with DDQN. Figure 5 depicts the overall cost of performing compute-intensive jobs in the resource reserve using the present as well as proposed dynamic pricing schemes. The graph indicates that the overall cost continuously rises as the amount of jobs grows. However, the existing pricing plan has a higher total cost than the constantly changing pricing scheme. When compared to the current pricing scheme, the dynamic pricing plan cuts total costs by 35%–45%. This is because the current pricing policy reserved all resources until the job execution was done. On the other hand, in the proposed OBL-WHO with DDQN variable price structure, resources are released after tasks are completed. When the user adopts an inexpensive pricing approach and the action in question space lacks a higher price, the IBO-WHO cannot arrive at the ideal amount of DDQN using the suggested technique. Table 1 displays the numerical findings of total cost versus jobs.

The overall cost of performing compute-intensive operations using various allocation techniques is depicted in Fig. 6. The figure illustrates that the suggested method lowered total costs by 20%, respectively, when compared to DRL dynamic pricing, GA-DQN, and multi-attribute auction pricing. The

Table 1
The numerical results of total cost vs jobs for dynamic pricing

Jobs (unit)	GA-DQN	Multi-attribute auction pricing	DRL dynamic pricing	OBL-WHO with DDQN
100	700	650	500	450
120	750	690	620	550
140	850	720	690	630
160	900	850	720	690
180	960	900	850	710

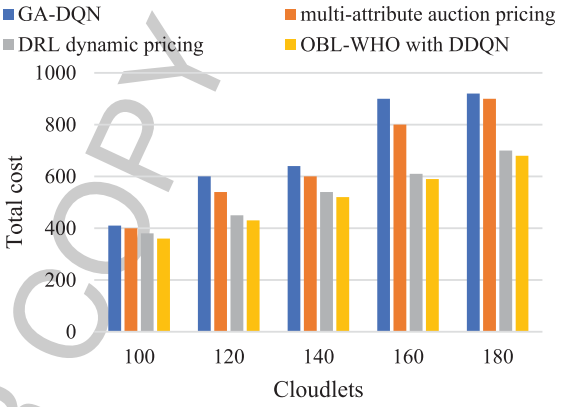

Fig. 6. The overall cost of utilizing dynamic pricing techniques to conduct data-intensive workloads in cloudlets.

proposed approach adjusts the price based on the consumption of the resources. Reduced costs for less-utilized resources ensure fairness among numerous service providers while also lowering total end-user costs existing schemes ensure equitable allocation among service providers, although at a higher cost than the suggested approach. Right now, OBL-WHO and DDQN can more successfully cut costs. OBL-WHO with DDQN exhibits practically identical sensitivity to the user's demanding quantity and computation time length when the user's demand intensity is high, thus we can effectively lower the cost of the system. Although the medium intensity group's experimental conditions in both sets of studies are the same, their experimental outcomes are likewise the same, much like OBL-WHO with DDQN's experiment. Table 2 provides the numerical outcomes of total cost vs. cloudlets.

In Fig. 7, the comparison to DRL dynamic pricing, GA-DQN, and multi-attribute auctioneer pricing, the suggested approach is 35%–5% more effective. Regarding cost-effectiveness, the proposed strategy outperforms alternatives. To reduce latency, the suggested approach can route the assignment to fewer congested service providers. Because GA-DQN

Table 2
The numerical results of total cost vs jobs for dynamic pricing

Cloudlets	GA-DQN	Multi-attribute auction pricing	DRL dynamic pricing	OBL-WHO with DDQN
100	410	400	380	360
120	600	540	450	430
140	640	600	540	520
160	900	800	610	590
180	920	900	700	680

Table 3
The numerical results of total cost vs data size for dynamic pricing

Data size	GA-DQN	Multi-attribute auction pricing	DRL dynamic pricing	IBO-WHO with DDQN
1800	200	180	170	165
2400	300	280	220	215
3000	380	340	320	315
3600	450	420	400	395
4200	510	490	440	435
4800	700	610	590	585

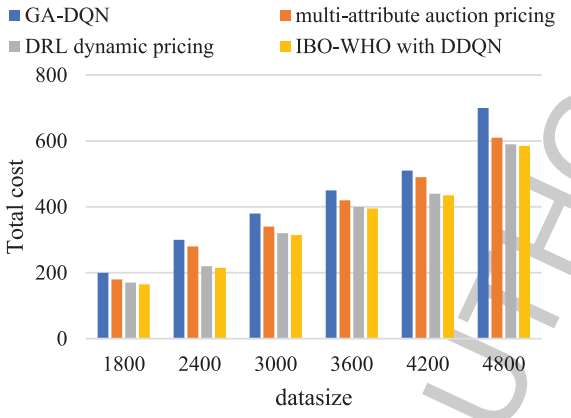

Fig. 7. Overall expense of performing data-intensive tasks with dynamic pricing models for various data sizes.

distributes jobs evenly among service providers regardless of network state, a saturated service provider may generate a bottleneck, resulting in an extended time to transfer data. The same is true for the multi-attribute auction pricing model, which distributes resources at random among service suppliers. Although the substantial difference value is a barrier for the current learning methods to successfully enhance the dynamic pricing model, the space of actions must be accurate and constrained whenever the consumer employs the uniform arrival rate. This might be resolved by giving the user access to a more focused and intricate action space. The user must raise the price when employing a low pricing policy to boost income rewards. If not, it will be difficult to enhance the player's revenue because the action done as a discount would just increase parking occupancy rates rather than contribute to doing so. Table 3 provides the numerical outcomes of total cost vs. cloudlets.

5. Conclusion and future work

The difficult problems of dynamic pricing models and the effectiveness of using resources across many cloud service providers are the main topics of this work. To reduce end users' costs and ensure fairness among service providers, a hybrid IBO-WHO with a DDQN-based dynamic pricing scheme is suggested. The price is adjusted by the plan to promote the use of natural assets with low rates of use and to discourage the use of commodities with high rates of utilization. By suggesting a DDQN agent that addresses the installation decision problem of deciding whether or not to join together services upon dynamic pricing changes in the federation, this research contributes to filling that gap. Despite its merits, the study has limitations. It exhibits promise by introducing a dynamic pricing scheme but operates in a controlled simulation environment, potentially diverging from real-time complexities. While showing cost reduction, its simplified pricing model might oversimplify real-world market dynamics, limiting its comprehensive applicability. Additionally, the assumptions on user behavior, though rational, might overlook the diverse and nuanced decision-making patterns in actual scenarios. The experimental findings demonstrate that, in terms of overall end-user cost, the suggested method beats the conventional pricing scheme and other widely used resource allocation algorithms. Future development should include a dynamic pricing scheme with additional optimization parameters including electrical consumption and user input. It is important to make assumptions regarding consumers' strategic behavior. To get the most out of their expected utility, individuals undertake cross-period transactions depending on cost, quality, and other criteria.

Declaration

Ethics approval and consent to participate

No participation of humans takes place in this implementation process.

Human and animal rights

No violation of human and animal rights is involved.

Funding

No funding is involved in this work.

Data availability statement

Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

Conflict of interest

Conflict of interest is not applicable in this work.

Authorship contributions

All authors are contributed equally to this work.

Acknowledgments

There is no acknowledgement involved in this work.

References

[1] Peter Mell and Timothy Grance, The NIST definition of cloud computing, *Gaithersburg: National Institute of Standards and Technology*, 2011.

[2] Xun Xu, From cloud computing to cloud manufacturing, *Robotics and Computer – Integrated Manufacturing* **28** (2012).

[3] Kunsoo Han, Robert J. Kauffman and Barrie R. Nault, Research note: returns to information technology outsourcing, *Information Systems Research* **4**(22) (2011), 824–840.

[4] Mimecast, Cloud computing delivering on its promise but doubts still hold back adoption, http://www.mimecast.com/News-and-views/Press-releases/Dates/2010/7/Cloud-computing-delivering-onits-promise-but-doubts-still-hold-back- adoption/, 2011.

[5] Martin Lockheed Cloud Computing Research Study. http://www.FCW.com/DownloadingCloudComputing, 2011.

[6] Stefan Ried, Holger Kisker, Pascal Matzke, Andrew Bartels and Miroslaw Lisserman, Sizing the cloud – understanding and quantifying the future of cloud computing, *Cambridge: Forrester Research*, 2011.

[7] V. Vinothina, R. Sridaran and D.P. Ganapathi, A Survey on Resource Allocation Strategies in Cloud Computing, *International Journal of Advanced Computer Science and Applications (IJACSA)* **3**(6) (2012), 97–104.

[8] A. Belgacem, Dynamic resource allocation in cloud computing: analysis and taxonomies, *Computing* **104**(3) (2022), 681–710.

[9] K. Saidi and D. Bardou, Task scheduling and VM placement to resource allocation in Cloud computing: challenges and opportunities, *Cluster Computing* (2023), 1–19.

[10] M. Mihailescu and Y.M. Teo, Dynamic Resource Pricing on Federated Clouds, *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid)*, 2010, pp. 513–517.

[11] R. Nagarajan, P. Vinothiyalakshmi and R. Thirunavukarasu, An Intelligent Cloud Broker with Service Ranking Algorithm for Validation and Verification of Cloud Services in Multi-cloud Environment, (2023).

[12] N. Mukhopadhyay and B.P. Tewari, Dynamic cost-effective solution for efficient cloud infrastructure, *The Journal of Supercomputing* **79**(6) (2023), 6471–6506.

[13] S. Adabi, F. Alayin and A. Sharifi, A new flexible pricing mechanism considering price–quality relation for cloud resource allocation, *Evolving Systems* **12** (2021), 541–565.

[14] R.A. Kumar and K. Kartheeban, Resource allocation using Dynamic Pricing Auction Mechanism for supporting emergency demands in Cloud Computing, *Journal of Parallel and Distributed Computing* **158** (2021), 213–226.

[15] B. Shi, L. Huang and R. Shi, A deep reinforcement learning-based approach for pricing in the competing auction-based cloud market, *Service Oriented Computing and Applications* **16**(2) (2022), 83–95.

[16] Z. Zhu, J. Peng, K. Liu and X. Zhang, A game-based resource pricing and allocation mechanism for profit maximization in cloud computing, *Soft Computing* **24** (2020), 4191–4203.

[17] B. Sharma, R.K. Thulasiram, P. Thulasiraman and R. Buyya, Clabacus: A risk-adjusted cloud resources pricing model using financial option theory, *IEEE Transactions on Cloud Computing* **3**(3) (2014), 332–344.

[18] P. Cong, J. Zhou, M. Chen and T. Wei, Personality-guided cloud pricing via reinforcement learning, *IEEE Transactions on Cloud Computing* **10**(2) (2020), 925–943.

[19] P. Cong, L. Li, J. Zhou, K. Cao, T. Wei, M. Chen and S. Hu, Developing user perceived value based pricing models for cloud markets, *IEEE Transactions on Parallel and Distributed Systems* **29**(12) (2018), 2742–2756.

[20] F. Alzhouri, A. Agarwal and Y. Liu, Maximizing cloud revenue using dynamic pricing of multiple class virtual machines, *IEEE Transactions on Cloud Computing* **9**(2) (2018), 682–695.

[21] J. Martín-Pérez, K. Antevski, A. Garcia-Saavedra, X. Li and C.J. Bernardos, Dqn dynamic pricing and revenue driven service federation strategy, *IEEE Transactions on Network and Service Management* **18**(4) (2021), 3987–4001.

[22] R.A. Kumar and K. Kartheeban, Resource allocation using Dynamic Pricing Auction Mechanism for supporting emergency demands in Cloud Computing, *Journal of Parallel and Distributed Computing* **158** (2021), 213–226.

[23] T.F. Ang, L.Y. Por and C.S. Liew, Dynamic pricing scheme for resource allocation in multi-cloud environment, *Malaysian Journal of Computer Science* **30**(1) (2017), 1–17.

[24] U. Moghe, P. Lakkadwala and D.K. Mishra, Cloud computing: Survey of different utilization techniques, *Sixth International Conference on Software Engineering (CONSEG)*, 2012, pp: 1–4.

[25] Q. Zheng and B. Veeravalli, Utilization-based pricing for power management and profit optimization in data cen-

ters, *Journal of Parallel and Distributed Computing* **72**(1) (2012), 27–34.

[26] B. Gu and Y. Sung, Enhanced DQN framework for selecting actions and updating replay memory considering massive non-executable actions, *Applied Sciences* **11**(23) (2021), 11162.

[27] K. Kavukcuoglu Mnih, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, Playing atari with deep reinforcement learning, *arXiv preprint arXiv:1312.5602*, 2013.

[28] M. Sewak and M. Sewak, Deep Q Network (DQN), Double DQN, and Dueling DQN: A Step Towards General Artificial Intelligence, *Deep Reinforcement Learning: Frontiers of Artificial Intelligence* (2019), 95–108.

[29] I. Naruei and F. Keynia, Wild horse optimizer: A new meta-heuristic algorithm for solving engineering optimization problems, *Engineering with Computers* **38**(Suppl 4) (2022), 3025–3056.

[30] H. Saitou and H. Haraguchi, Grey Wolf Optimization using Improved mutation oppositional based learning for optimization problems, In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)* (2022, September), (pp. 1–8). IEEE.

[31] A.S. Phansalkar, Measuring program similarity for efficient benchmarking and performance analysis of computer systems, University of Texas at Austin, 2007.

[32] C. Bennett, R.L. Grossman, D. Locke, J. Seidman and S. Vejcik, Malstone: towards a benchmark for analytics on large data clouds, *Proceedings of the 16th ACM SIGKDD International conference on Knowledge discovery and data mining*, 2010, pp: 145–152.

[33] A. Rajaram, Improved NEH-Heuristic job scheduling for an optimal system using meta-heuristic GA–INSMG. *International Journal of u-and e-Service, Science and Technology* **9**(7) (2016), 213–226.

[34] G.J. Rathanam and A. Rajaram, Trust based meta-heuristics workflow scheduling in cloud service environment, *Circuits and Systems* **7**(04) (2016), 520.

[35] J.S. Jayaprakash, K. Balasubramanian, R. Sulaiman, M.K. Hasan, B.D. Parameshachari and C. Iwendi, Cloud Data Encryption and Authentication Based on Enhanced Merkle Hash Tree Method, *Computers, Materials & Continua* **72**(1) (2022).