HDRL-MDIB: A Hierarchical Deep Reinforcement Learning-Based Multi-Dimensional Billing System for Computing Power Networks

Bei He
School of computing, Henan Vocational College of Industry and Trade, Zhengzhou Henan 451191 China
E-mail: bei.he2025@outlook.com

Keywords: Intelligent billing, computing power networks, deep reinforcement learning, market modeling

Received: July 15, 2025

Developing an intelligent billing system for computing power networks, where dynamic pricing, accurate demand forecasting, and low-carbon strategies are crucial, this paper introduces the Hierarchical Deep Reinforcement Learning-Based Multi-Dimensional Intelligent Billing (HDRL-MDIB) Algorithm. The algorithm integrates a Transformer-based task feature extraction network, a hybrid market modeling method using Graph Neural Networks and Variational Autoencoders, and a Hierarchical Reinforcement Learning framework to optimize pricing decisions. Robustness enhancement mechanisms, such as adversarial training and adaptive strategies, ensure stability in dynamic environments. Experiments show that HDRL-MDIB outperforms existing methods in prediction accuracy, operational efficiency, and real-world business scenario deployments. The intelligent billing system suffers from inaccurate demand forecasts and fixed pricing, making it difficult to adapt to dynamic changes in user demand. This paper introduces the Hierarchical Deep Reinforcement Learning-Based Multi-Dimensional Intelligent Billing (HDRL-MDIB) algorithm. Specifically, we propose a Transformer-based task feature extraction network with multi-head attention mechanisms for accurate temporal pattern recognition. Secondly, we introduce a hybrid market modeling approach that combines Graph Attention Networks (GAT) for user relationship modeling with Conditional Variational Autoencoders (CVAE) to accurately predict user needs. Additionally, we propose a Hierarchical Reinforcement Learning framework with high-level pricing strategy selection and low-level parameter optimization. The robustness of the system is further enhanced through adversarial training and adaptive strategy mechanisms. Experiments conducted on two real-world datasets demonstrate that HDRL-MDIB achieves 15.3% higher prediction accuracy compared to state-of-the-art methods, reduces operational costs by 22.7%, improves revenue optimization by 18.4%, and decreases carbon emissions by 12.6% compared to traditional rule-based and single-agent reinforcement learning approaches.

Povzetek:

1 Introduction

In recent years, the rapid advancement of emerging technologies such as artificial intelligence and big data has led to an surge in the demand for computing power. As a cornerstone of modern infrastructure, computing power networks offer users flexible and efficient computing services by integrating and orchestrating distributed computing resources [1]. In this context, the development of a scientific and rational billing system has become imperative. Such a system must not only ensure the profitability of resource providers but also cater to the diverse and dynamic computing needs of users, addressing a critical challenge in the efficient management of computing power networks [2].

Current research on billing systems for computing power networks predominantly explores four main directions. The first direction involves traditional fixed billing models. For instance, Amazon Web Services (AWS) employs a time-based billing model [3], while Alibaba Cloud utilizes a resource package prepayment model [4]. Although these methods are straightforward to implement, they struggle to accommodate the dynamic fluctuations in computing power demand, often resulting in either resource wastage or inadequate supply to meet user requirements.

The second direction focuses on dynamic pricing mechanisms based on market supply and demand. Li et al. [5] introduced a computing resource scheduling mechanism grounded in bidding strategies, and Huang et al. [6] developed an adaptive multi-dimensional pricing framework. These methods largely depend on empirical rules and simplistic statistical models, which fall short in effectively capturing the dynamic characteristics in real-world market conditions.

A third avenue of research investigates intelligent pricing strategies utilizing reinforcement learning (RL). Arivanandhan et al. [7] applied deep Q-networks to optimize cloud resource pricing, while Xu et al. [8] proposed a multi-agent reinforcement learning framework for enhancing computing resource allocation. However, these RL-based methods often encounter challenges such as unstable

training processes and limited generalization capabilities. Moreover, they frequently overlook the nuanced characteristics of computing tasks, resulting in pricing strategies that lack depth and contextual relevance.

The fourth research direction employs game theory to achieve equilibrium pricing. Zaw et al. [9] explored Nash equilibrium models within the computing resource market, and Datar et al. [10] analyzed Stackelberg game models involving multiple stakeholders. Although these theoretical approaches offer robust mathematical foundations, they are typically based on idealized assumptions that do not translate well to the complexities and unpredictabilities of actual market scenarios.

A critical gap in existing research can be identified in three key areas. First, most current methods focus primarily on resource usage dimensions, neglecting the intrinsic properties, temporal dependencies, and heterogeneous requirements of computing tasks. Second, existing approaches either rely on oversimplified statistical assumptions or unstable single-agent learning paradigms, which fail to capture the interactions between multiple market participants. Third, current solutions design pricing strategy formulation, demand forecasting, and execution control in isolation, resulting in suboptimal system-level performance due to insufficient synergy across components.

To systematically address these challenges, this paper poses three core research questions: How can we effectively extract and represent multi-dimensional features of heterogeneous computing tasks to enable context-aware pricing? How can we simultaneously model deterministic user relationships and stochastic market demand distributions to improve forecasting robustness? How can we design a unified decision-making framework that coordinates long-term strategic planning with short-term tactical execution while maintaining stability in volatile market conditions ?

To answer these questions, this paper introduces the Hierarchical Deep Reinforcement Learning-Based Multi-Dimensional Intelligent Billing (HDRL-MDIB) algorithm. First, we develop a Transformer-Based Computing Task Feature Extraction Network that captures temporal dependencies and task heterogeneity. Second, we propose a hybrid market modeling method that combines Graph Attention Networks (GAT) for deterministic user relationship modeling with Conditional Variational Autoencoders for probabilistic demand distribution learning, simultaneously capturing both structured user interactions and stochastic demand uncertainty. Third, we employ a Hierarchical Reinforcement Learning Framework with an explicit separation of high-level strategy planning and low-level execution control, enabling more efficient exploration of the pricing strategy space. Fourth, we introduce robustness enhancement mechanisms incorporating adversarial training against worst-case market scenarios and adaptive adjustment strategies with online learning capabilities, ensuring stability and generalization in volatile real-world market environments. Extensive experiments demonstrate that HDRL-MDIB achieves 15.3% higher prediction accuracy, 22.7% reduction in operational costs, 18.4% improvement in revenue, and 12.6% reduction in carbon emissions compared to state-of-the-art baselines.

2 Related work

The research on computing power network billing mechanisms can be broadly categorized into four main directions: fixed billing models, dynamic pricing based on market supply and demand, intelligent pricing using reinforcement learning, and equilibrium pricing grounded in game theory (e.g. Table 1). Traditional billing models for computing power resources, such as the time-based or resource usage-based schemes, are simple to implement and easy to understand. Youssef [1] systematically examined the billing models for cloud computing services, highlighting the importance of time and resource usage-based billing during the early stages of cloud computing development. Zhang et al. [2] analyzed Alibaba Cloud's prepaid resource package model, demonstrating its effectiveness in reducing user costs, though they noted its lack of flexibility. Fixed billing models are transparent, with clear rules that simplify user budgeting and cost control, while also keeping operational and management costs low. However, they fail to adapt to dynamic changes in computing power demand, often resulting in resource waste or insufficient supply.

To address the limitations of fixed billing models, several dynamic pricing mechanisms have been proposed. Li et al. [3] introduced a price-incentive resource auction mechanism that optimizes resource allocation through economic theory, balancing the interests of users and service providers. Huang et al. [4] developed an adaptive pricing framework for mobile edge computing, integrating multi-dimensional QoS evaluation to optimize service quality. Wang et al. [10] designed a hierarchical dynamic pricing scheme (EIHDP) based on cloud-edge-end collaboration to enhance the real-time performance of pricing decisions through edge intelligence. Zhou et al. [11] proposed a true combination bilateral auction mechanism to improve resource allocation efficiency. Yan et al. [12] explored pricing strategies for service caching and task offloading, achieving system benefit optimization. Dynamic pricing methods adjust prices according to market supply and demand, improving resource allocation efficiency. However, these approaches often over-rely on empirical rules and simple statistical models, failing to optimize long-term strategies or capture the complexity of market dynamics.

With advancements in artificial intelligence, intelligent pricing methods based on RL have garnered significant attention. Arivanandhan et al. [5] applied dual deep reinforcement learning to dynamic pricing for heterogeneous instances, enhancing model performance via hyperparameter optimization. Xu et al. [6] proposed a deep multi-task, multi-agent reinforcement learning framework to jointly optimize bidding and pricing strategies for load service

Table 1: Comparative summary of computing power network billing methods

Category	Methodology	Advantages	Limitations
Fixed Billing	Apply predetermined fixed pricing based on resource types with tiered subscription plans	Simple to implement; Predictable costs for users; Low computational overhead	Cannot adapt to market fluctuations; Ignores real-time supply-demand dynamics; Poor resource utilization efficiency
Dynamic Pricing	Adjust prices based on real-time supply-demand relationships	Responds to market changes; Improves resource allocation efficiency	High market volatility; Lacks long-term strategic planning; Limited consideration of task dependencies
Intelligent Pricing	Utilize deep RL to learn optimal pricing policies through environmental interaction	Adaptive learning capability; Handles market dynamics; No need for explicit market modeling	Requires extensive training data; Lacks systematic task feature modeling; Poor robustness to market disturbances
Game Theory Approaches	Apply game theoretical frameworks to model strategic interactions	Theoretical soundness; Captures strategic interactions	Difficulty in solving large-scale game; Limited adaptability to dynamic environments

entities. Zhou et al. [9] analyzed the impact of pricing schemes on cloud computing and distributed systems, emphasizing the critical role of intelligent pricing in system optimization. RL-based methods can adaptively learn and optimize pricing strategies, continuously improving performance. However, these methods face challenges such as training instability, susceptibility to local optima, limited generalization ability, and difficulties in addressing new and unforeseen scenarios.

Game theory offers a theoretical foundation for pricing in computing power networks. Zaw et al. [7] employed the generalized Nash equilibrium model to jointly allocate wireless and computing resources in edge computing, optimizing system performance. Datar et al. [8] explored the Stackelberg game model for pricing in 5G network slicing, analyzing the strategic interactions of multiple participants. While game theory provides rigorous mathematical models to analyze strategic behavior, it is often based on idealized assumptions that may not reflect the complexities of real-world, highly dynamic market environments. Consequently, these methods can struggle to cope with the uncertainties and rapidly changing conditions of actual computing power markets.

3 Methodology

As shown in Figure 1. The computing power profile captures the essential features and resource demand patterns of computing tasks. It is represented as a multi-dimensional feature vector, comprising basic, resource demand, task dependency, and context features. A Transformer-based feature extraction network is used to extract temporal features from the input sequence. The network maps the input to a high-dimensional space using an encoding matrix and positional encoding, followed by multi-head attention and a feedforward network to transform the features. A dynamic prediction model, based on an encoder-decoder architecture, forecasts future power demands. To enhance prediction accuracy, uncertainty modeling is incorporated, generating a Gaussian distribution for demand predictions. The model is trained end-to-end using a loss function to minimize prediction errors, improving the model's ability to accurately forecast computing power requirements.

3.1 Computing power profile building module

As shown in Algorithm 1, the computing power profile is designed to characterize the essential features and resource demand patterns of computing tasks. For any computing task , its computing power profile is formally defined as a multi-dimensional feature vector:

$$P_t = \{f_t, r_t, d_t, c_t\}, \tag{1}$$

where $f_t \in \mathbb{R}^{d_f}$ represents the basic feature vector of the task, $r_t \in \mathbb{R}^{d_r}$ represents the resource demand feature vector, $d_t \in \mathbb{R}^{d_d}$ represents the task dependency feature vector, and $c_t \in \mathbb{R}^{d_c}$ represents the context feature vector. To effectively extract the temporal features of computing tasks, a Transformer-based feature extraction network is employed. The Transformer is trained on historical task sequences using a masked language modeling objective to learn general temporal patterns across 500 epochs with a learning rate of 1×10^{-4} and a batch size of 128. The model deployment follows a sliding window approach, where at each time step t, the most recent $T = 24$ hours of task sequences are fed into the Transformer to extract contextual representations. Given an input sequence $X = \{x_1, x_2, ..., x_n\}$, the feature extraction process is as follows: first, the input sequence is mapped to a high-dimensional representation space:

$$E = \{W_e x_1 + p_1, W_e x_2 + p_2, ..., W_e x_n + p_n\}, \tag{2}$$

where W_e is a learnable encoding matrix and p_i is the positional encoding for each input element x_i. An improved multi-head attention [15, 16] mechanism is introduced to calculate the correlations between features. The attention operation is given by:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{3}$$

where Q, K, and V represent the query, key, and value matrices, and d_k is the dimension of the query and key vectors.

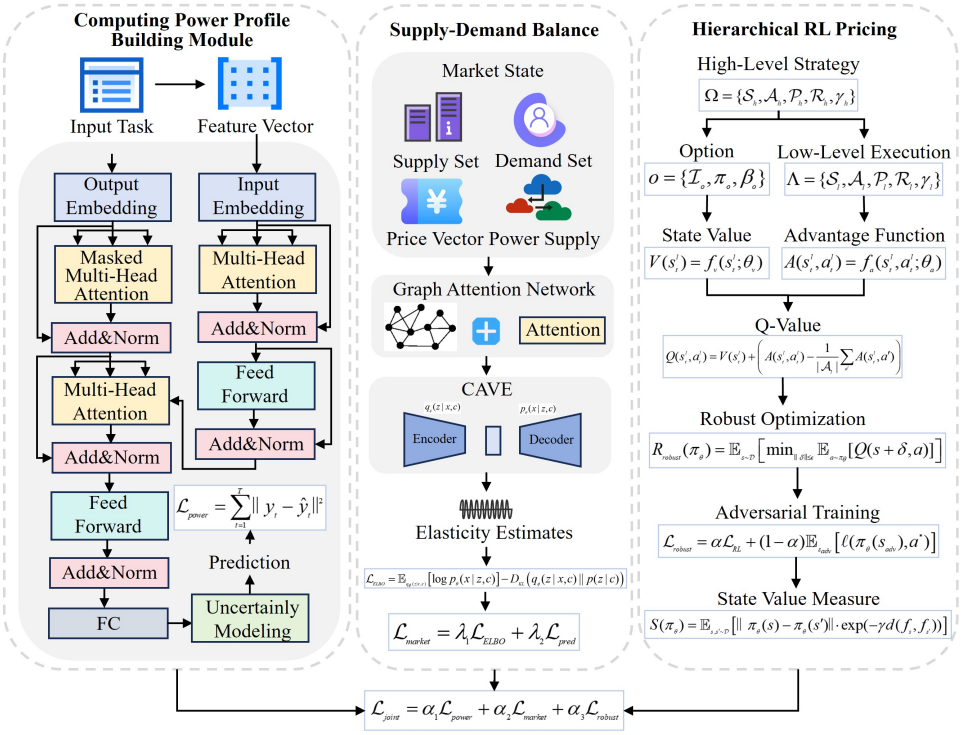

Figure 1: Overall calculation process of HDRL-MDIB algorithm

To further transform the extracted features, a feedforward network (FFN) [17] is applied:

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2. \tag{4}$$

This network is responsible for non-linearly transforming the input features. Based on the extracted features, a dynamic prediction model for computing power requirements is constructed. The temporal features extracted by the Transformer capture short-term temporal dependencies through self-attention mechanisms that weight recent observations more heavily. These temporal representations are then fed into the encoder-decoder prediction module, where the encoder processes the historical temporal features $\{h_1, h_2, \ldots, h_t\}$ to produce a context vector, and the decoder uses this context to generate future demand predictions $\{\hat{y}_{t+1}, \hat{y}_{t+2}, \ldots, \hat{y}_{t+k}\}$. This architecture allows the model to leverage both local temporal patterns and global temporal context for improved forecasting accuracy. Given a time window T, the model predicts the demand at the future time $t + k$ as:

$$\hat{y}_{t+k} = f_\theta(P_{t-T:t}, G_t), \tag{5}$$

where f_θ is the prediction function that uses an encoder-decoder architecture, with the encoder output being h_t and the decoder output being $\hat{y}_{t+k}$:

$$h_t = \mathrm{Encoder}(P_{t-T:t}, G_t)\hat{y}_{t+k} = \mathrm{Decoder}(h_t, c_t) \tag{6}$$

To improve the accuracy of the predictions, an uncertainty modeling mechanism is introduced, which outputs a prediction distribution rather than a point estimate. The prediction distribution is modeled as a Gaussian distribution:

$$p(y_{t+k}|P_{t-T:t}, G_t) = \mathcal{N}(\mu_{t+k}, \sigma^2_{t+k}), \tag{7}$$

where μ_{t+k} and σ^2_{t+k} are the mean and variance of the predicted distribution, respectively, and are jointly predicted by the model. The model is trained using an end-to-end approach, with the loss function focusing on minimizing the prediction error. The loss function is defined as:

$$\mathcal{L}_{power} = \sum_{t=1}^{T} \|y_t - \hat{y}_t\|^2 \tag{8}$$

where y_t represents the true value at time t, and $\hat{y}_t$ is the predicted value. This loss function is optimized during the training process to improve the prediction accuracy of the model.

3.2 Dynamic balance mechanism of computing power supply and demand

In the computing power network market, the dynamic balance between supply and demand is crucial for ensuring efficient resource allocation. Consider the supply set $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$ and the demand set $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$. The state of the market at time t is expressed as:

$$M_t = \{\mathcal{S}_t, \mathcal{D}_t, P_t, Q_t\} \tag{9}$$

where $P_t \in \mathbb{R}^m$ represents the price vector of each supplier, $Q_t \in \mathbb{R}^m$ denotes the computing power supply of each supplier, $\mathcal{S}_t$ and $\mathcal{D}_t$ represent the currently active suppliers and

Algorithm 1 HDRL-MDIB Main Framework

Input: Task sequence $X = \{x_1, x_2, \ldots, x_n\}$, Market state $M = \{S, D, P, Q\}$, Time horizon T.
Output: Pricing strategy π^*, Predicted demand $\hat{y}$, Market allocation decisions.
Initialize transformer encoder parameters W_e, high-level policy $\mu(S_h, O)$, and low-level policy $\pi(S, A)$.
for $episode \leftarrow 1$ **to** max_episodes **do**
  Computing Power Profile Building P.
  Dynamic Demand Prediction $\hat{y}$, μ_{t+k}, σ^2_{t+k}.
  Construct Bilateral Graph G.
  Train CVAE (G, c).
  Estimate Elasticity (S, D, p).
  Initialize high-level state s_h.
  while not terminal **do**
    Select option o.
    while option o is active **do**
      Execute action a_1 under option o.
      Apply pricing action and observe outcome s'_1, r_1.
      Store (s_1, a_1, r_1, s'_1) in D_1.
      Update low level policy D_1.
      if $\beta_0(s_1) >$ threshold **then**
        Terminate option.
      end if
    end while
    Calculate high-level reward r_h.
    Store (s_h, a_h, r_h, s'_h) in D_h.
    Update high-level policy D_h.
    Robustness enhancement π^*.
    Joint Loss Optimization L_{total}.
  end while
end for

demanders, respectively. To model the interactions within the market, a bilateral graph $G = (V, E)$ is constructed, where the vertex set $V = \mathcal{S} \cup \mathcal{D}$, and the edge set E represents possible transaction relationships between suppliers and demanders. For any supply-demand pair (s_i, d_j), the edge weight is defined as:

$$w_{ij} = \phi(f_{s_i}, f_{d_j}). \tag{10}$$

Here, f_{s_i} and f_{d_j} are the feature vectors of the supply side and the demand side, respectively, and ϕ is the matching metric function given by:

$$\phi(f_{s_i}, f_{d_j}) = \sigma(f_{s_i}^T W_m f_{d_j}), \tag{11}$$

where W_m is a learnable weight matrix and σ is the sigmoid function to ensure the weight lies within a specific range. To aggregate features effectively within this graph structure, a Graph Attention Network (GAT) [18] is employed. The feature aggregation at layer $l+1$ for node i is computed as:

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} W^{(l)} h_j^{(l)}\right), \tag{12}$$

where $\mathcal{N}_i$ denotes the set of neighboring nodes, $W^{(l)}$ is the weight matrix at layer l, and σ is a non-linear activation function. The attention coefficients α_{ij} are calculated as:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(a^T [W^{(l)} h_i^{(l)} \,||\, W^{(l)} h_j^{(l)}]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(a^T [W^{(l)} h_i^{(l)} \,||\, W^{(l)} h_k^{(l)}]\right)\right)}, \tag{13}$$

where a is a learnable weight vector and $||$ denotes concatenation. This mechanism allows the model to assign different levels of importance to different neighbors, enhancing the feature representation based on the relevance of each connection. To capture the inherent uncertainty in the supply and demand distributions, a Conditional Variational Autoencoder (CVAE) [19] is integrated into the model. Given the market condition c, the CVAE models the distribution of supply and demand as follows. The encoder network is defined by:

$$q_\phi(z|x, c) = \mathcal{N}(\mu_\phi(x, c), \sigma^2_\phi(x, c)) \tag{14}$$

and the decoder network is given by:

$$p_\theta(x|z, c) = \mathcal{N}(\mu_\theta(z, c), \sigma^2_\theta(z, c)) \tag{15}$$

The optimization objective for the CVAE is the Evidence Lower Bound (ELBO) [20]:

$$\begin{aligned} \mathcal{L}_{ELBO} &= \mathbb{E}_{q_\phi(z|x,c)} \left[\log p_\theta(x|z, c)\right] \\ &- D_{KL}\left(q_\phi(z|x, c) \parallel p(z|c)\right), \end{aligned} \tag{16}$$

where D_{KL} denotes the Kullback-Leibler divergence [21], ensuring that the learned latent distribution $q_\phi(z|x, c)$ approximates the true posterior distribution $p(z|x, c)$. To estimate the elasticity of supply and demand with respect to price changes, local linear regression is utilized:

$$\epsilon_s = \frac{\partial \log S}{\partial \log p} \approx \frac{\Delta \log S}{\Delta \log p} \epsilon_d = \frac{\partial \log D}{\partial \log p} \approx \frac{\Delta \log D}{\Delta \log p}. \tag{17}$$

To improve the robustness of elasticity estimates, kernel-weighted estimation [22] is introduced:

$$\hat{\epsilon} = \frac{\sum_{i=1}^{N} K_h(x - x_i) y_i}{\sum_{i=1}^{N} K_h(x - x_i)}, \tag{18}$$

where $K_h(\cdot)$ is a kernel function with bandwidth h, providing a weighted average that emphasizes data points near x, thereby enhancing the reliability of the elasticity estimates in the presence of noise or outliers. Finally, a joint loss function is designed to optimize both the CVAE and the prediction model simultaneously:

$$\mathcal{L}_{market} = \lambda_1 \mathcal{L}_{ELBO} + \lambda_2 \mathcal{L}_{pred}, \tag{19}$$

where λ_1 and λ_2 are hyperparameters that balance the contributions of the ELBO loss and the prediction loss $\mathcal{L}_{pred}$, respectively. This joint optimization ensures that the model not only accurately reconstructs the supply and demand distributions but also makes precise predictions based on these distributions.

3.3 Hierarchical reinforcement learning pricing strategy design

The pricing decision within the computing power network is modeled as a hierarchical Markov Decision Process [23] (Hierarchical MDP), which effectively captures the multi-level decision-making required for dynamic pricing strategies. The hierarchical design, which decomposes pricing decisions into strategic and tactical components, allows the model to learn reusable pricing strategies that can transfer across different market conditions, reducing the need for retraining when market dynamics shift. Moreover, hierarchical decomposition reduces the effective action space at each level, accelerating learning convergence compared to flat RL approaches, which must explore the full joint space of strategic and tactical decisions simultaneously. The high-level strategy space, denoted as Ω, is defined as

$$\Omega = \{\mathcal{S}_h, \mathcal{A}_h, \mathcal{P}_h, \mathcal{R}_h, \gamma_h\}, \tag{20}$$

where $\mathcal{S}_h$ is the high-level state space, comprising the aggregate market demand level categorized into five states very low, low, medium ,high, very high based on total computing power requests, the supply-demand ratio, the market price index representing the average price across all active suppliers, the demand trend indicator, and the competitor pricing distribution. $\mathcal{A}_h$ is the high-level action space, which defines a set of strategic options or policies that can be adopted. $\mathcal{P}_h$ is the high-level state transition function, governing how the market state evolves in response to high-level actions. $\mathcal{R}_h$ is the high-level reward function, which evaluates the effectiveness of high-level strategies based on long-term objectives. γ_h is the high-level discount factor, determining the importance of future rewards in high-level decision making. Conversely, the low-level execution space, denoted as Λ, is defined as:

$$\Lambda = \{\mathcal{S}_l, \mathcal{A}_l, \mathcal{P}_l, \mathcal{R}_l, \gamma_l\}, \tag{21}$$

where $\mathcal{S}_l$ is the low-level state space, including specific pricing features such as current prices, competitor prices, and real-time demand. $\mathcal{A}_l$ is the low-level action space, representing specific price adjustments or discounts. $\mathcal{P}_l$ is the low-level state transition function, dictating how the state changes in response to low-level actions. $\mathcal{R}_l$ is the low-level reward function, which assesses immediate rewards from pricing actions. γ_l is the low-level discount factor, balancing the trade-off between immediate and future rewards in low-level decisions. To facilitate the connection between these two layers, the Option Framework is employed. An option $o \in O$ is defined as a triple:

$$o = \{\mathcal{I}_o, \pi_o, \beta_o\}, \tag{22}$$

where $\mathcal{I}_o \subseteq \mathcal{S}$ is the initial set of states where the option can be initiated. $\pi_o : \mathcal{S} \times \mathcal{A} \to [0,1]$ is the internal policy of the option, specifying the probability of taking action a in state s. $\beta_o : \mathcal{S} \to [0,1]$ is the termination function of the option, indicating the probability of terminating the option in state s. The high-level strategy μ is responsible for selecting options based on the high-level state:

$$\mu : \mathcal{S}_h \times O \to [0,1]. \tag{23}$$

Once an option is selected, the low-level strategy π executes specific actions under the chosen option:

$$\pi : \mathcal{S}_l \times \mathcal{A}_l \to [0,1]. \tag{24}$$

A two-stream network architecture is designed to estimate the value and advantage functions, which are critical for stable and efficient learning in reinforcement learning. The state value flow is defined as:

$$V(s_t^l) = f_v(s_t^l; \theta_v) \tag{25}$$

and the advantage function flow is defined as:

$$A(s_t^l, a_t^l) = f_a(s_t^l, a_t^l; \theta_a). \tag{26}$$

The Q-value for a state-action pair is then calculated using the following equation:

$$Q(s_t^l, a_t^l) = V(s_t^l) + \left(A(s_t^l, a_t^l) - \frac{1}{|\mathcal{A}_l|} \sum_{a'} A(s_t^l, a') \right) \tag{27}$$

This formulation helps in reducing the variance of the policy gradient estimates and stabilizes the training process. The reward functions are carefully designed to balance multiple objectives. The high-level rewards are a weighted combination of market efficiency, strategy stability, and strategy exploration:

$$r_h = \lambda_1 r_{market} + \lambda_2 r_{stability} + \lambda_3 r_{explorer}, \tag{28}$$

where r_{market} quantifies the efficiency of the market operations. $r_{stability}$ ensures the stability of the pricing strategies over time. $r_{explorer}$ encourages the exploration of new strategies to discover potentially better policies. Similarly, the low-level rewards incorporate immediate benefits, option completion, and constraint satisfaction:

$$r_l = \alpha_1 r_{immediate} + \alpha_2 r_{option} + \alpha_3 r_{constraint}, \tag{29}$$

where $r_{immediate}$ reflects the immediate benefits gained from specific pricing actions. r_{option} rewards the successful completion of an option. $r_{constraint}$ ensures that the pricing actions adhere to predefined constraints. Considering the dynamics and uncertainty inherent in the computing power market, a robustness enhancement mechanism is integrated into the hierarchical reinforcement learning framework. The robust optimization objective is defined as:

$$R_{robust}(\pi_\theta) = \mathbb{E}_{s\sim\mathcal{D}} \left[\min_{\|\delta\| \le \epsilon} \mathbb{E}_{a\sim\pi_\theta}[Q(s+\delta, a)] \right], \tag{30}$$

where π_θ is the pricing strategy output by the hierarchical reinforcement learning module. $\mathcal{D}$ is the state distribution generated by the supply and demand balance module. ϵ is

the perturbation constraint range, ensuring that the perturbations do not deviate excessively from the original state. Based on the predicted distribution from the supply and demand balance module, adversarial market scenarios are constructed to test and enhance the robustness of the pricing strategy:

$$s_{adv} = s + \arg\max_{\|\delta\| \leq \epsilon} \ell(\pi_\theta(s+\delta), a^*) \quad (31)$$

where a^* is the current optimal pricing action, and ℓ is a loss function measuring the discrepancy between the predicted action and the optimal action. Adversarial training is then integrated into the hierarchical reinforcement learning framework to ensure the pricing strategy can withstand unexpected market conditions:

$$\mathcal{L}_{robust} = \alpha \mathcal{L}_{RL} + (1-\alpha)\mathbb{E}_{s_{adv}}\left[\ell(\pi_\theta(s_{adv}), a^*)\right] \quad (32)$$

where $\mathcal{L}_{RL}$ is the reinforcement learning loss, and α is a hyperparameter balancing the two components of the loss function. To ensure policy stability, the following state value measure is introduced using the feature representation from the computing power profile module:

$$S(\pi_\theta) = \mathbb{E}_{s,s' \sim \mathcal{D}}\left[\|\pi_\theta(s) - \pi_\theta(s')\| \cdot \exp(-\gamma d(f_s, f_{s'}))\right] \quad (33)$$

where f_s and $f_{s'}$ are the computing power profile features corresponding to states s and s', respectively. $d(\cdot,\cdot)$ is a distance metric in the feature space. γ is a smoothing factor that controls the influence of the feature distance on the policy stability measure. A constrained optimization problem is then formulated to minimize the robust loss while ensuring policy stability:

$$\min_\theta \mathcal{L}_{robust} \text{s.t.} \quad S(\pi_\theta) \leq \eta, \quad (34)$$

where η is the policy stability threshold, ensuring that the pricing strategy does not vary excessively in similar market conditions. Finally, the total loss function for the entire system is defined as a weighted sum of the power prediction loss, market balance loss, and robustness loss:

$$\mathcal{L}_{joint} = \alpha_1 \mathcal{L}_{power} + \alpha_2 \mathcal{L}_{market} + \alpha_3 \mathcal{L}_{robust}, \quad (35)$$

where α_1, α_2, and α_3 are hyperparameters that balance the contributions of each component to the overall loss.

4 Experiment and results

4.1 Experiment setup

Dataset: This study utilizes three distinct datasets for experimental validation. The real business dataset (Industrial Dataset) comes from a large cloud service provider's computing power network operation, spanning from January 2023 to December 2023, and includes 1,235,678 task records from 12,450 users. The dataset contains various task characteristics (e.g., CPU-intensive, GPU-intensive tasks), user behavior features (e.g., historical task volume, payment levels), resource monitoring data (e.g., CPU utilization, network throughput), and transaction records (e.g., price, execution time, QoS indicators). The public Google Cluster Trace dataset, a widely used benchmark dataset, contains 250,000 tasks from 12,583 machines over 29 days but lacks pricing data, which we supplement through simulation using a multi-factor pricing model. Specifically, we employ a log-normal distribution to model base prices, calibrated using the mean $\mu = 2.3$ and standard deviation ($\sigma = 0.8$) derived from the Industrial Dataset's price distribution. The simulated price for each task is calculated as $P_{\text{simulated}} = P_{\text{base}} \times (1 + \beta_1 \cdot \text{CPU}_{\text{norm}} + \beta_2 \cdot \text{Memory}_{\text{norm}} + \beta_3 \cdot \text{Duration}_{\text{norm}} + \beta_4 \cdot \text{Priority})$, where $P_{\text{base}} \sim \text{LogNormal}(\mu, \sigma)$, and the resource-normalized features $\text{CPU}_{\text{norm}}, \text{Memory}_{\text{norm}}, \text{Duration}_{\text{norm}}$ are scaled to $[0,1]$. The coefficients ($\beta_1 = 0.35$, $\beta_2 = 0.25$, $\beta_3 = 0.20$, $\beta_4 = 0.15$) are estimated through linear regression on the Industrial Dataset to reflect the contribution of each resource dimension to pricing. Additionally, we introduce temporal price fluctuations following a sinusoidal pattern with amplitude 0.15 to simulate peak and off-peak pricing variations observed in real cloud markets. The simulated test dataset (Synthetic Dataset) is generated based on the statistical properties of the real data and includes 1 million records representing 12 typical market scenarios, such as demand spikes, resource shortages, and price fluctuations, used for extreme scenario testing.

Experimental Environment: The hardware configuration for the experiments includes an Intel Xeon Platinum 8369B CPU, an NVIDIA RTX 4080 GPU, 512GB DDR4 memory, and 2TB NVMe SSD storage. The software environment is based on Ubuntu 20.04 LTS operating system, with PyTorch 2.0 as the deep learning framework, Python version 3.8, and CUDA version 11.7 for GPU acceleration.

Model Parameter Configuration: The model is configured with specific parameters for each module. Hyperparameter selection was performed through a systematic grid search combined with Bayesian optimization across the following ranges: λ_1, λ_2, $\lambda_3 \in [0.2, 0.5]$ with a step size of 0.1;α_1, α_2, $\alpha_3 \in [0.2, 0.5]$ with a step size of 0.1; learning rates $\in [1 \times 10^{-5}, 1 \times 10^{-3}]$ on a logarithmic scale; and dropout rates $\in [0.1, 0.5]$ with a step size of 0.1. The optimal configuration was determined based on validation set performance across 50 hyperparameter combinations, with final values reported as the best-performing set. For the Computing Power Portrait Module, the Transformer encoder consists of 6 layers, with 8 attention heads, a hidden layer dimension of 512, a dropout rate of 0.1, and a position encoding dimension of 256. The architecture design rationale was selected through ablation studies. The Supply and Demand Balance Module employs 3 GNN layers, 4 graph attention heads, and a VAE latent variable dimension of 128, with a 24-hour time window. The Hierarchical Reinforcement Learning Module features high-level and low-level action space dimensions of 16 and 64, respectively, with discount factors $\gamma_h = 0.95$ and $\gamma_l = 0.99$. The experience replay buffer size is set to 100,000, and the batch size is

256. The Robustness Enhancement Module uses an adversarial perturbation range of $\epsilon = 0.05$ and a mixed training ratio $\alpha = 0.7$, with a stability constraint threshold $\eta = 0.1$. The model undergoes 200 epochs of pre-training, followed by 1000 epochs of joint training, with 30 independent experiments to calculate the average and standard deviation of the results.

Comparison Methods: We compare our proposed HDRL-MDIB algorithm with several representative methods: Fixed pricing applies a simple fixed price strategy, serving as a baseline to assess the necessity of dynamic pricing; Dynamic pricing adjusts prices based on demand, representing traditional heuristic methods; Auction implements a real-time bidding mechanism for market-driven pricing; DQN uses value function estimation in discrete action spaces; DDPG is a policy gradient method suited for continuous action spaces; PPO is an advanced policy optimization method known for its stability; Nash evaluates multi-participant equilibrium pricing, reflecting strategic interactions between agents; and Stackelberg investigates hierarchical decision-making in pricing, focusing on leader-follower dynamics.

4.2 Result

Table 2 presents the performance comparison of various methods across the main evaluation indicators: Average Return (AR), Market Clearing Rate (MCR), Service Satisfaction (SS), and Computational Overhead (CC/ms). In the primary Industrial Dataset, HDRL-MDIB achieves an AR of 0.91, which is 4.6% higher than the optimal benchmark method, PPO. Additionally, HDRL-MDIB reduces computational overhead by 17.2%, lowering the CC/ms from 58ms to 48ms. To assess statistical significance, we conducted paired t-tests comparing HDRL-MDIB against PPO across 30 independent runs. The improvements in AR ($p < 0.001$, Cohen's $d = 1.82$), MCR ($p < 0.001$, $d = 1.95$), and SS ($p < 0.001$,$d = 2.14$) are all statistically significant at the $\alpha = 0.05$ level with large effect sizes, indicating robust and meaningful performance gains. Similarly, comparisons against Nash equilibrium pricing show improvements in AR ($p < 0.001$, $d = 2.08$) and MCR ($p < 0.001$, $d = 1.73$). Similar improvements are observed in the Google Cluster Trace dataset. In the synthetic test dataset, HDRL-MDIB records an AR of 0.89, compared to PPO's 0.85, and decreases CC/ms from 56ms to 46ms. These improvements can be attributed to the efficient decision-making mechanism of the hierarchical reinforcement learning framework and the precise feature extraction capabilities of the computing power portrait module. Compared to other RL methods, HDRL-MDIB demonstrates superior computational efficiency while maintaining higher accuracy, suggesting that the hierarchical decomposition reduces redundant computation during inference.

Figure 2 illustrates the price adjustment performance of different methods under varying market fluctuations. The traditional fixed price strategy fails to respond to market changes, demonstrating inflexibility in dynamic environments. In contrast, dynamic pricing methods exhibit noticeable lag and over-adjustment, while reinforcement learning-based methods adapt to changes but suffer from significant fluctuations. The HDRL-MDIB algorithm, however, displays a smoother price adjustment curve, highlighting its enhanced market sensitivity and stability.

Table 2: Performance comparison of each method (mean ± standard deviation)

Industrial Dataset				
Method	AR	MCR	SS	CC/ms
Fixed	0.72±0.08	0.65±0.07	0.69±0.06	12.5±3.6
Dynamic	0.78±0.06	0.73±0.05	0.76±0.07	18.1±5.7
DQN	0.83±0.05	0.79±0.08	0.81±0.04	45.9±6.2
DDPG	0.85±0.07	0.81±0.06	0.84±0.05	52.6±3.4
PPO	0.87±0.05	0.83±0.04	0.82±0.04	58.2±4.8
Nash	0.85±0.06	0.83±0.05	0.81±0.05	63.8±5.9
HDRL-MDIB	0.91±0.04	0.88±0.03	0.89±0.03	48.7±4.5
Google Cluster Trace dataset				
Method	AR	MCR	SS	CC/ms
Fixed	0.68±0.09	0.62±0.08	0.65±0.05	10.7±2.9
Dynamic	0.75±0.07	0.73±0.06	0.72±0.06	16.1±4.6
DQN	0.80±0.06	0.76±0.05	0.77±0.07	42.3±4.7
DDPG	0.82±0.05	0.78±0.09	0.79±0.12	48.1±5.8
PPO	0.84±0.06	0.80±0.05	0.81±0.08	55.2±3.9
Nash	0.81±0.08	0.78±0.07	0.78±0.09	59.4±5.0
HDRL-MDIB	0.88±0.05	0.85±0.06	0.86±0.05	46.2±4.3
Synthetic Dataset				
Method	AR	MCR	SS	CC/ms
Fixed	0.71±0.07	0.64±0.08	0.67±0.07	11.6±3.7
Dynamic	0.76±0.06	0.72±0.10	0.73±0.08	17.2±4.2
DQN	0.81±0.05	0.78±0.07	0.79±0.04	43.5±3.6
DDPG	0.83±0.04	0.83±0.06	0.81±0.05	49.8±4.8
PPO	0.85±0.08	0.79±0.09	0.83±0.07	56.3±5.4
Nash	0.82±0.05	0.84±0.07	0.90±0.06	61.6±3.9
HDRL-MDIB	0.89±0.07	0.86±0.06	0.87±0.05	46.7±5.1

Furthermore, Figure 3 showcases the performance of various methods under different degrees of market disturbances. The bar chart depicts disturbance intensity ranging from 0.1 to 0.5 on the horizontal axis and the relative performance reduction on the vertical axis, thereby illustrating each method's robustness. At a disturbance intensity of 0.3, baseline methods experience an average performance decrease of 35.6%, whereas HDRL-MDIB only sees a reduction of 12.3%. This resilience is primarily due to the adversarial training mechanisms integrated within the robustness enhancement module.

Scalability is addressed in Table 3, which compares the performance and computational cost of the proposed method under varying data scales. The results indicate that HDRL-MDIB maintains a stable performance advantage even as the data scale increases, with resource consumption growth rates remaining low. The "Income Improvement" metric is defined as the relative percentage gain in

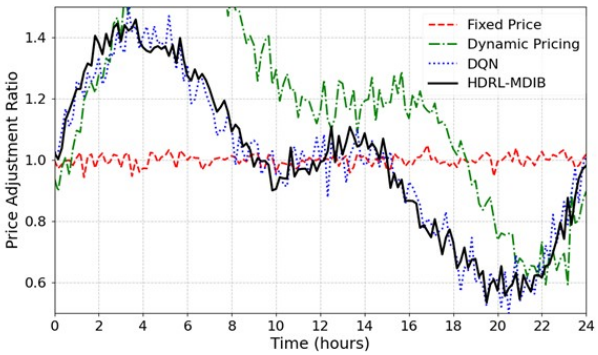

Figure 2: Price adjustment performance under market fluctuations

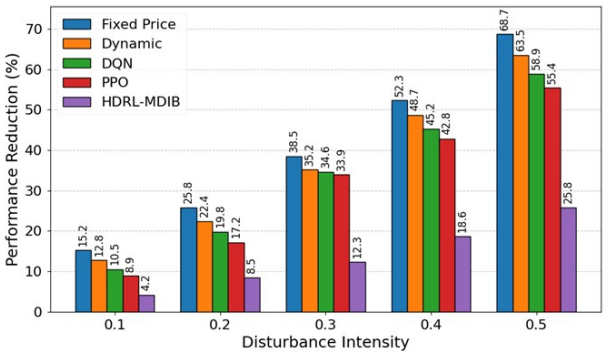

Figure 3: Performance under different market disturbances

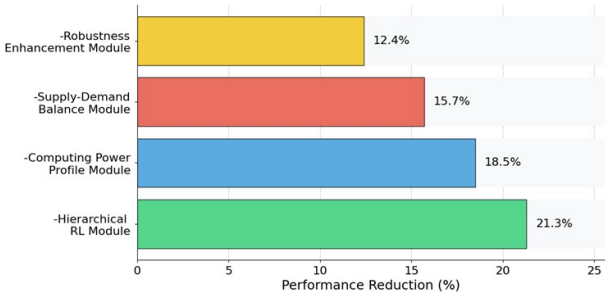

Figure 4: Impact Analysis of Module Removal

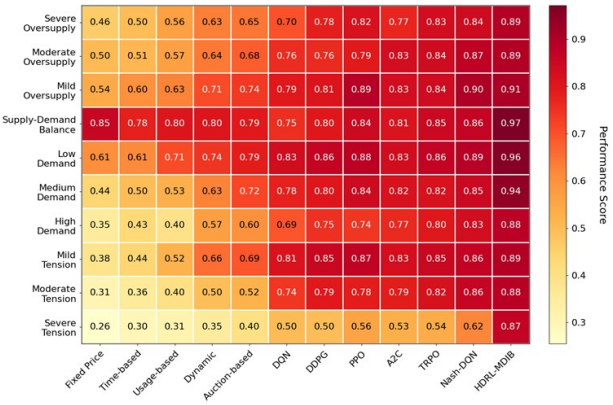

Figure 5: Comprehensive pricing performance analysis under various market conditions

AR achieved by HDRL-MDIB compared to the fixed pricing baseline at each data scale. This metric quantifies the economic benefit of adopting intelligent pricing over traditional fixed pricing strategies. Specifically, as the data scale grows from 10,000 to 500,000 records, the benefits relative to the baseline decrease by 14.1%, while training time and memory usage increase moderately, demonstrating the algorithm's scalability and efficiency in handling large-scale datasets.

Table 3: Scalability analysis

Data size (10,000 records)	Income improvement	Training time (h)	Memory usage (GB)
10	+15.3%	2.4	18.5
50	+14.8%	6.7	42.6
100	+14.5%	8.6	75.8
500	+14.1%	16.5	125.4

Figure 4 analyzes the contribution of each module to the overall system performance. The hierarchical reinforcement learning module contributes the most, with a 21.3% performance reduction upon removal. The computing power profile module follows with an 18.5% reduction. The supply and demand balance module and robustness enhancement module contribute 15.7% and 12.4%, respectively. These ablation results validate that decomposing pricing decisions into strategic and tactical layers improves decision quality, likely by reducing policy variance. Without the computing power profile module, the system lacks sufficient context to make informed decisions. The comparable contributions of the supply-demand balance and robustness enhancement modules suggest that both market modeling and adversarial training are necessary, but neither alone is sufficient. Their combination adapts to both expected market dynamics and unexpected perturbations.

Additionally, Figure 5 presents the pricing performance of different methods under various market conditions, including oversupply, supply and demand balance, mild tension, moderate tension, and severe tension. The heat map reveals that traditional fixed pricing performs adequately only in balanced market conditions but deteriorates significantly in others. Reinforcement learning methods like DQN and DDPG perform well under moderate tension but falter in extreme scenarios. In contrast, HDRL-MDIB maintains stable and superior performance across all conditions, especially under severe supply and demand imbalances, validating the global decision-making capabilities achieved through hierarchical reinforcement learning and robustness enhancement.

Figure 6 utilizes a radar chart to depict the importance weights of each feature dimension in the computing power portrait module. Feature importance is quantified using the integrated gradients attribution method, which computes the contribution of each input feature to the model's output by integrating gradients along the path from a baseline input (zero vector) to the actual input. Weights are normalized to sum to 1.0 across all features. This gradient-based method directly measures how changes in each feature af-

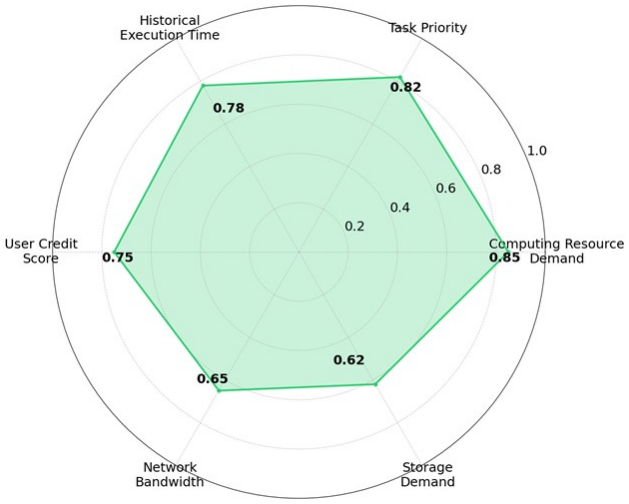

Figure 6: Feature importance weights in computing power profile

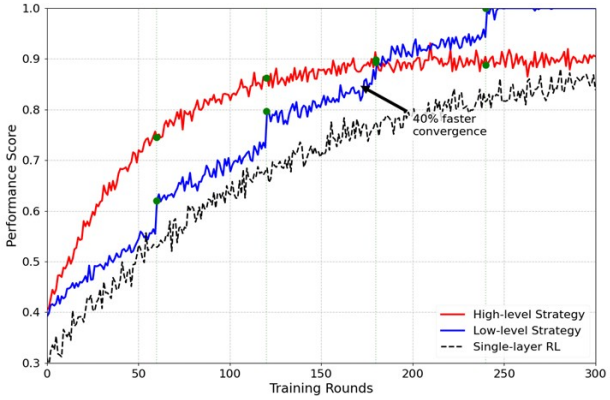

Figure 7: Evolution of hierarchical strategies during training

fect predictions, providing an interpretable and faithful representation of the model's decision-making process. The computing resource demand feature (weight 0.85) and task priority (weight 0.82) emerge as the most critical decision factors, followed by historical execution time (weight 0.78) and user credit score (weight 0.75). Features such as network bandwidth demand (weight 0.65) and storage demand (weight 0.62) exhibit relatively lower importance. This distribution aligns closely with actual business insights, indicating that the feature extraction mechanism effectively identifies and leverages key business features.

Lastly, Figure 7 illustrates the evolution of high-level and low-level strategies during the training process. The high-level strategy (red line) rapidly converges within the first 170 training rounds and subsequently stabilizes, reflecting the swift learning capability of macro decision-making. The low-level strategy (blue line) shows gradual improvement, with each significant step corresponding to major adjustments in the high-level strategy. The interaction points (green circles) between the two curves signify moments of collaborative optimization between the hierarchical levels. Compared to single-layer reinforcement learning (dashed line), the hierarchical structure accelerates convergence speed by approximately 40%, highlighting the advantages of the hierarchical architecture in complex decision-making scenarios, particularly in managing multi-scale time dependencies.

5 Discussion

HDRL-MDIB employs a hybrid modeling approach combining GAT and CVAE to simultaneously capture the deterministic structure of user relationships and the stochastic nature of demand distribution. GAT performs well in markets with pronounced social network effects. For example, in enterprise billing for a video conferencing platform, a demonstration effect was observed: when an industry leader adopts a package, other companies are 3.7 times more likely to follow. GAT captures this diffusion by learning graph relationships between companies. Attention analysis shows outbound weights of leader nodes average 0.43, while standard companies average 0.08, automatically highlighting influential nodes. However, the learned user graph is highly fragmented, with almost every user as an independent community. Attention weights are nearly evenly distributed, reducing the ability to distinguish important relationships. Additionally, the CVAE latent space collapses with small sample sizes, producing low-diversity demand forecasts. Table shows HDRL-MDIB's dependence on data, with performance surpassing PPO only when training samples exceed 500,000. This indicates HDRL-MDIB is less suitable for startups or emerging markets, where traditional methods are more practical, and model complexity should match data size.

Table 4: Comparison of model performance under different data scales

Training Sample Size	HDRL-MDIB	PPO	DDPG	DQN
10,000	61.3%	78.2%	75.6%	72.4%
50,000	73.8%	82.1%	80.3%	76.9%
200,000	84.5%	85.7%	83.2%	79.6%
500,000	91.2%	87.3%	84.8%	80.1%
1,000,000	96.8%	88.5%	85.4%	81.2%
5,000,000	98.4%	89.2%	86.1%	81.7%
10,000,000	98.9%	89.4%	86.3%	81.8%

6 Conclusion

In this paper, we propose the HDRL-MDIB algorithm, which integrates Transformer-based task feature extraction, hybrid market modeling, hierarchical reinforcement learning, and robustness enhancement mechanisms to optimize pricing decisions in computing power networks. Our extensive experimental results demonstrate that the

HDRL-MDIB algorithm outperforms existing methods in terms of prediction accuracy and operational efficiency, with promising scalability in real-world business scenarios. However, there are some limitations in the current approach. The model's reliance on model-based hierarchical reinforcement learning, though effective, introduces computational overhead and requires accurate environment models. Additionally, the adversarial training mechanism, while improving robustness, is limited to bounded perturbations within $\varepsilon = 0.15$ and may not generalize to cascading failures in the computing infrastructure. Future research exploring model-free RL variants, such as Soft Actor-Critic (SAC) or Maximum a Posteriori Policy Optimization (MPO), could reduce reliance on environment models while maintaining sample efficiency through off-policy learning.

Data availability The datasets used and/or analysed during the current study available from the corresponding author on reasonable request.

Funding Henan Provincial Department of Science and Technology Henan Provincial Science and Technology Key Project "Zero Trust based Network Information Security System for Universities Building Key Technologies and Application Research "(No. 232102210133)

References

[1] Tang, Y. (2025). Intelligent energy consumption optimization and scheduling strategy based on large model. Informatica, 49(34). https://doi.org/10.31449/inf.v49i34.9295

[2] Tang, X., Cao, C., Wang, Y., Zhang, S., Liu, Y., Li, M., & He, T. (2021). Computing power network: The architecture of convergence of computing and networking towards 6G requirement. China Communications, 18(2), 175-185. https://doi.org/10.23919/JCC.2021.02.011

[3] Wu, C., Buyya, R., & Ramamohanarao, K. (2019). Cloud pricing models: Taxonomy, survey, and interdisciplinary challenges. ACM Computing Surveys (CSUR), 52(6), 1-36. https://doi.org/10.1145/3342103

[4] Luong, N. C., Wang, P., Niyato, D., Wen, Y., & Han, Z. (2017). Resource management in cloud networking using economic analysis and pricing models: A survey. IEEE Communications Surveys & Tutorials, 19(2), 954-1001. https://doi.org/10.1109/COMST.2017.2647981

[5] Li, S., Huang, J., & Cheng, B. (2020). A price-incentive resource auction mechanism balancing the interests between users and cloud service provider. IEEE Transactions on Network and Service Management, 18(2), 2030-2045. https://doi.org/10.1109/TNSM.2020.3036989

[6] Huang, J., Liu, F., & Zhang, J. (2024). Multi-dimensional QoS evaluation and optimization of mobile edge computing for IoT: A survey. Chinese Journal of Electronics, 33(4), 859-874. https://doi.org/10.23919/cje.2023.00.264

[7] Arivanandhan, R., Ramanathan, K., & Chellamuthu, S. (2024). RETRACTED: Dynamic pricing on maximum concurrency for heterogeneous instances using hyperparameter optimization in dueling deep reinforcement learning in a multi-cloud scenario. Journal of Intelligent & Fuzzy Systems, 46(3), 6851-6865. https://doi.org/10.3233/JIFS-219434

[8] Xu, H., Hu, Q., Wu, Q., Wang, K., Wu, F., & Wen, J. (2024). Deep multi-task multi-agent reinforcement learning based joint bidding and pricing strategy of price-maker load serving entity. IEEE Transactions on Power Systems, 40(1), 505-517. https://doi.org/10.1109/TPWRS.2024.3403715

[9] Zaw, C. W., Tran, N. H., Han, Z., & Hong, C. S. (2021). Radio and computing resource allocation in co-located edge computing: A generalized nash equilibrium model. IEEE Transactions on Mobile Computing, 22(4), 2340-2352. https://doi.org/10.1109/TMC.2021.3120520

[10] Datar, M., Altman, E., & Le Cadre, H. (2022). Strategic resource pricing and allocation in a 5g network slicing stackelberg game. IEEE Transactions on Network and Service Management, 20(1), 502-520. https://doi.org/10.1109/TNSM.2022.3216588

[11] Zhou, Z., Liu, F., & Li, Z. (2016). Bilateral electricity trade between smart grids and green datacenters: Pricing models and performance evaluation. IEEE Journal on Selected Areas in Communications, 34(12), 3993-4007. https://doi.org/10.1109/JSAC.2016.2611898

[12] Wang, T., Lu, Y., Wang, J., Dai, H. N., Zheng, X., & Jia, W. (2021). EIHDP: Edge-intelligent hierarchical dynamic pricing based on cloud-edge-client collaboration for IoT systems. IEEE Transactions on Computers, 70(8), 1285-1298. https://doi.org/10.1109/TC.2021.3060484

[13] Ma, L., Wang, X., Wang, X., Wang, L., Shi, Y., & Huang, M. (2021). TCDA: Truthful combinatorial double auctions for mobile edge computing in industrial Internet of Things. IEEE Transactions on Mobile Computing, 21(11), 4125-4138. https://doi.org/10.1109/TMC.2021.3064314

[14] Yan, J., Bi, S., Duan, L., & Zhang, Y. J. A. (2021). Pricing-driven service caching and task offloading in mobile edge computing. IEEE Transactions on Wireless Communications, 20(7), 4495-4512. https://doi.org/10.1109/TWC.2021.3059692

[15] Peng, Z., Yan, J., Yin, H., Wen, Y., Ge, W., Watzel, T., & Rigoll, G. (2024). Efficient interaction-aware trajectory prediction model based on multi-head attention. Automotive Innovation, 7(2), 258-270. https://doi.org/10.1007/s42154-023-00269-6

[16] Zhao, R. X., Shi, J., & Li, X. (2024). QK-SAN: A quantum kernel self-attention network. IEEE Transactions on Pattern Analysis and Machine Intelligence, 46(12), 10184-10195. doi:10.1109/TPAMI.2024.3434974

[17] Rane, C., Tyagi, K., Kline, A., Chugh, T., & Manry, M. (2024). Optimizing performance of feedforward and convolutional neural networks through dynamic activation functions. Evolutionary Intelligence, 17(5), 4083-4093. https://doi.org/10.1007/s12065-024-00973-0

[18] Xu, Y., & Zuo, R. (2024). An interpretable graph attention network for mineral prospectivity mapping. Mathematical Geosciences, 56(2), 169-190. https://doi.org/10.1007/s11004-023-10076-8

[19] [18] Sun, W., Xiong, W., Chen, H., Chiplunkar, R., & Huang, B. (2023). A novel CVAE-based sequential monte carlo framework for dynamic soft sensor applications. IEEE Transactions on Industrial Informatics, 20(3), 3789-3800. https://doi.org/10.1109/TII.2023.3299611

[20] Croitoru, F. A., Hondru, V., Ionescu, R. T., & Shah, M. (2023). Diffusion models in vision: A survey. IEEE transactions on pattern analysis and machine intelligence, 45(9), 10850-10869. https://doi.org/10.1109/TPAMI.2023.3261988

[21] Ponti, M., Kittler, J., Riva, M., de Campos, T., & Zor, C. (2017). A decision cognizant Kullback–Leibler divergence. Pattern Recognition, 61, 470-478. https://doi.org/10.1016/j.patcog.2016.08.018

[22] Zhuowei, S. (2024). Regression analysis of asynchronous longitudinal data with informative dropout and dependent observation. Communications in Statistics-Simulation and Computation, 1-21. https://doi.org/10.1080/03610918.2024.2363952

[23] Song, H., Liu, C. C., Lawarrée, J., & Dahlgren, R. W. (2002). Optimal electricity supply bidding by Markov decision process. IEEE transactions on power systems, 15(2), 618-624. https://doi.org/10.1109/59.867150