IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 5 August 2023, accepted 2 September 2023, date of publication 8 September 2023, date of current version 13 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3313425

RESEARCH ARTICLE

A Novel Dynamic Pricing Approach for Preemptible Cloud Services

HUIJIE PENG AND YAN CHENG
School of Business, East China University of Science and Technology, Shanghai 200237, China
Corresponding author: Yan Cheng (yancheng@ecust.edu.cn)

ABSTRACT Dynamic pricing for preemptible cloud services (DPPCS) is highly demanded to effectively utilize the excess capacity in cloud computing. However, the dynamic nature of excess capacity exhibits high non-stationarity, which is characterized by multi-temporal stochastic patterns with time-varying statistical properties. The non-stationarity results in the DPPCS problem being a Non-Stationary Markov Decision Process (NSMDP) with unknown transition probabilities. Moreover, DPPCS is constrained by a certain maximum preemption rate, further complicating the DPPCS problem as a Constrained NSMDP (CNSMDP). We transform the CNSMDP into a piecewise Lagrangian dual model, which converts the CNSMDP into an unconstrained optimization problem. To solve the above problem, we propose a novel Q-Learning approach for DPPCS. We first present estimation methods for the unknown environment parameters, including a detection method for identifying temporal pattern changes, and a diffusion approximation method for estimating the actual preemption rate. Then, we introduce a Lagrange multiplier updating method, which can strike a balance between revenue and the preemption rate in the reward function. Building upon the above methods, we develop a Constrained Non-Stationary Q-Learning (CNSQL) algorithm for DPPCS, which dynamically adjusts its learning process to adapt to the multi-temporal patterns. Through simulated experiments, we demonstrate the effectiveness of our proposed approach compared to state-of-the-art algorithms. It performs well in improving revenue generated from excess capacity while maintaining the actual preemption rate within the specified constraint.

INDEX TERMS Cloud computing, preemptible service, dynamic pricing, non-stationarity, Q-learning.

I. INTRODUCTION

Preemptible cloud services (PCS) are a type of low-priority service offered by cloud operators (e.g., Amazon Web Services) at a discounted price, without a Service Level Agreement (SLA) guarantee [1]. These services allow cloud operators to attract customers sensitive to service price while keeping the right to reclaim capacities when necessary. The primary goal of PCS is maximizing revenue by utilizing excess capacity from the conflict between the static capacity of computing resources and the volatile workloads of high-priority services. High-priority services, supported by SLAs, are mainstream offerings with fixed regular prices [2]. Resources required by high-priority services must be provided whenever consumers need them to avoid SLA violations. Workloads of high-priority services typically exhibit non-stationarity with specific temporal patterns such as periodic diurnality, bursting, growing, and on/off, resulting from regular business behaviors or unexpected external events [3], [4]. Due to the relatively fixed total capacity of computing resources, the non-stationarity of workloads for high-priority services also leads to corresponding non-stationarity in excess capacity. This means that the excess capacity not only fluctuates stochastically but also changes with time-varying statistical properties. Therefore, a significant amount of unused excess capacity occurs during off-peak periods of workload [5]. The excess capacity can be utilized by offering preemptible services at substantial discounts. Conversely, during peak workload periods, capacity scarcity occurs. Preemptible services that exceed the available capacity can be preempted to prioritize high-priority services with SLA guarantees.

The associate editor coordinating the review of this manuscript and approving it for publication was Abderrahmane Lakas.

In practice, a significant portion of tasks referred to as fault-tolerance tasks,[1] can tolerate service interruption. Because of the flexible nature of these workloads, consumers are willing to withstand preemptions in exchange for a lower price. As a result, the preemptible services have gained popularity among several well-known cloud operators,[2] enabling the utilization of unused excess capacity by fulfilling fault-tolerance computing tasks. Therefore, cloud operators can effectively leverage the unused excess capacity by implementing efficient pricing management strategies for the preemptible cloud services.

Dynamic pricing is highly demanded in the operation of PCS by adapting to the supply-and-demand dynamics [6]. This pricing strategy not only has the potential to maximize revenue but also reduces the probability of being preempted. In recent years, dynamic pricing has gained significant attention from cloud operators such as AWS and Ali Cloud [7]. Unfortunately, empirical evidence shows that the price fluctuations in dynamic pricing lack effective theoretical and methodological guidance, which leads to random price changes within a tight price range [8], [9]. Currently, there is a lack of academic research on dynamic pricing specifically for preemptible cloud services. In contrast, dynamic pricing for regular cloud services, which do not involve preemption, has been extensively studied [10], [11], [12]. Cloud computing is a perishable resource in which unused computing resources cannot be carried over to future periods. Dynamic pricing in cloud computing has often been studied using a revenue management framework derived from economics. This framework aims to optimize the pricing of perishable resources, maximizing expected revenue from price-sensitive customers. The dynamic pricing problem is generally formulated as an MDP, and dynamic programming techniques are used to solve the optimization problem. However, these studies assume a fixed resource capacity and model a stationary MDP framework, where all elements of the stationary MDP model are represented. These assumptions and solutions may not be suitable for pricing the preemptible cloud services.

The dynamic pricing of preemptible cloud services (DPPCS) entails two distinctive challenges. Firstly, the non-stationarity exhibited by the excess capacity renders the dynamic pricing problem as an NSMDP with unknown transition probabilities. The excess capacity after satisfying high-priority services, is an exogenous variable beyond the control of the prices of preemptible service. As a result, It leads to DPPCS being an NSMDP that has time-varying transition probabilities [13]. Moreover, the occurrence of random transitions across multiple temporal patterns, marked by unknown changepoints, introduces uncertainty into the state transition probabilities within the NSMDP framework. Secondly, the DPPCS is constrained by a maximum preemption rate, further complicating it as a Constrained NSMDP (CNSMDP). Cloud operators often set and publicly disclose a maximum preemption rate constraint to mitigate the perceived uncertainty of interruption and attract consumers.[3] In an MDP, we can effectively handle this constraint by balancing the supply-and-demand dynamics. However, the uncertainty within the NSMDP framework introduces the risk of the actual preemption rate exceeding the maximum constraint. This is attributed to the substantial variation in the resulting actual preemption rate across different temporal patterns, even when applying the same price policy under similar supply-and-demand conditions. Consequently, to solve the DPPCS problem, it is necessary to initially address the inherent uncertainty in state transition probabilities within the CNSMDP framework. This challenge becomes particularly pronounced when attempting to utilize traditional dynamic programming techniques and closed-form solutions.

However, Q-Learning (QL), as a model-free algorithm in Reinforcement Learning (RL), has been widely used for solving MDP [14]. It focuses on training an automatic agent to learn how to make optimal decisions by interacting with uncertain environments, without relying on explicit knowledge of state transition probabilities. Driven by the advantages of ''no need for expert knowledge'' and ''model-free'', QL has become one of the important tools to optimize dynamic pricing problems [15], [16], [17], [18]. However, these approaches are designed for stationary MDPs with constant state transition probabilities and rewards. In our research, the CNSMDP problem not only involves non-stationarity with time-varying state transition probabilities and rewards, but also needs to achieve the maximum constraint. The traditional QL approaches may result in suboptimal pricing policies and high preemption rates in solving the DPPCS problem.

To address the above problem, we propose a novel QL approach for DPPCS, which aims to adapt to the non-stationarity while maintaining the maximum constraint. The key idea behind our approach is to decompose the CNSMDP into multiple CMDPs, and then each CMDP is transformed into a Lagrangian dual problem. By extending the reward function with the Lagrangian multiplier, the optimal value of the Lagrangian multiplier and optimal policy can be learned by the novel QL approach. Besides, there is another advantage is that our approach can adapt QL to learn optimal policies for different CMDP by embedded online detection method. Thus, there is no need to re-learn the policy when the temporal pattern changes. We only need to select an appropriate Lagrangian multiplier that makes its corresponding optimal policy under the changed temporal pattern. It can help our approach respond quickly to the various temporal

[1] According to MS Azure, fault-tolerant tasks are ''flexible workloads, like large processing jobs, dev/test environments, demos, and proofs of concept''. https://azure.microsoft.com/en-us/blog/low-priority-scale-sets. Accessed on March 13, 2023.

[2] Amazon Web Services (AWS), Alibaba Cloud (Ali Cloud), Microsoft Azure (MS Azure), and Google Cloud Platforms (GCP). Netapp. Excess capacity at AWS, Google, Azure and Alibaba cloud [EB/OL]. https://spot.io/what-is-excess-capacity/, 2022-04.

[3] AWS discloses different maximum preemption rates in public for different types of preemptible instances: 5%, 5-10%, 10-15%, and 15-20%. https://aws.amazon.com/cn/ec2/spot/instance-advisor/, accessed on June 19, 2023.

patterns in changing the excess capacity of cloud computing. The contribution of this work is fourfold:

(1) We formulate the DPPCS as a CNSMDP model. The CNSMDP model captures the uncertainty inherent in the dynamic pricing process and incorporates realistic supply-and-demand dynamics. We transform it into a piecewise Lagrange dual problem, which converts the CNSMDP into an unconstrained problem.

(2) A novel QL approach is proposed to solve the DPPCS problem. The proposed approach consists of two parameter estimation methods, and a policy learning method, in which two parameter estimation methods are proposed to identify the changed temporal patterns and estimate the actual preemption rates.

(3) For the policy learning method, an improved QL algorithm called CNSQL for DPPCS is proposed. CNSQL combines the advantages of the bisection method and the detection method of pattern change. It addresses the problem of automatically adjusting its learning process among temporal patterns and striking a balance between maximizing revenue and constraining the preemption rate.

(4) A series of experiments are conducted on the simulated and real-world scenarios with non-stationarity to validate the performance of the novel dynamic pricing approach. Compared to baseline algorithms, the proposed approach performs well in revenue maximization within a maximum constraint.

The remainder of the paper is organized as follows. In Section II, the related works on pricing strategies for excess computing capacity and the RL approach for dynamic pricing are reviewed. In Section III, we formulate the DPPCS as CNSMDP, which is transformed into a piecewise Lagrangian dual problem. To solve the above problem, we introduce the novel dynamic pricing approach and present the detailed algorithm in Section IV. Finally, a series of experiments on synthetic and real-world scenarios are conducted to verify the performance of our proposed approach in Section V and we summarize the paper in Section VI.

II. RELATED WORK

In this section, we review the previous studies and identify the research gaps. This paper is closely related to two streams of literature. The first stream focuses on pricing strategies for excess computing capacity. The second stream focuses on Reinforcement Learning for solving dynamic pricing.

A. PRICING STRATEGIES FOR THE EXCESS COMPUTING CAPACITY

The excess computing capacity refers to unused resources after satisfying the high-priority services. To effectively utilize the excess capacity, two common pricing strategies are adopted in the cloud industry. The first strategy is static pricing with a uniform preemption rule. The second strategy is dynamic pricing with an auction-like preemption rule. GCP and MS Azure both adopt static pricing with a uniform preemption rule, where users are charged a uniform price and a uniform preemption rate is implemented[4,5]. The uniform price and preemption rate can be transparent, and predictable, and everyone is treated equally. Chen et al. [1] examined a fixed-pricing strategy based on a uniform preemption rate and determined the context in which the pricing strategy is suitable. However, the uniform price cannot adapt to the varying supply-and-demand dynamics. On the contrary, AWS and Ali Cloud adopt the dynamic pricing strategy with an auction-like preemption rule[6,7]. In this strategy, consumers bid for resources, and preemption occurs when the real-time price exceeds their bids. Research on dynamic pricing with auction-like mechanisms has gained broad attention. Agmon Ben-Yehuda et al. [9] analyzed the dynamic price historical traces publicized by AWS and found that usually, prices are not market-driven, as sometimes previously assumed. Rather, they are likely to be generated most of the time at random from within a tight price range via a dynamic hidden reserve price mechanism. Kumar et al. [19] and Deldari and Salehan [20] surveyed the related research about the auction-like pricing for preemptible services, concluding that auction-like spot pricing is unpredictable and not easy to understand for users. Peng et al. [21] first considered the utilization of excess capacity with non-stationarity and proposed a real-time pricing method with a spot pricing scheme.

Theoretically, pricing the excess computing capacity dynamically can generate a higher revenue than static pricing. However, the auction mechanism makes the price fluctuate highly and then leads to the preemption rate varying widely and unpredictably. Also, consumers may engage in collaborative bidding to game the market. Currently, there is still a debate over the practicability of pricing schemes for preemptible services, due to the complexity of dynamic pricing with auction mechanisms and economically inefficiency of static pricing. Therefore, a more practical and efficient pricing strategy for preemptible service is required.

Dynamic pricing with posted prices is an alternative market-driven strategy for optimizing revenue in dynamic pricing. The seller sets and publicly displays prices, taking into account supple-and-market conditions and other relevant factors. Consumers act as price-takers, accepting the prices set by the seller. This strategy overcomes the limitations of auctions and has recently gained attention from AWS and Alibaba Cloud [8]. These operators are transitioning away from dynamic pricing with auction mechanisms and instead determining prices based on historical supply-demand trends, with updates occurring at regular intervals, typically every five minutes. The preemption rate in this pricing scheme is disclosed by a uniform value for all users, rather than

[4]https://azure.microsoft.com/en-us/blog/low-priority-scale -sets, accessed on May 21, 2023.

[5]https://cloud.google.com/compute/docs/instances/preemptible, accessed on May 21, 2023.

[6]https://help.aliyun.com/document_detail/52088.html?spm=5176.ecsnewbuy.help.dexternal.1fbd3675N6BQYb, accessed on June 10, 2023.

[7]https://docs.aws.amazon.com/zh_cn/AWSEC2/latest/UserGuide/using-spot-instances.html, accessed on June 10, 2023.

through a bidding process. However, there is limited research on this dynamic pricing strategy with a uniform preemption rate. Studies on dynamic pricing with posted prices for perishable goods offer valuable theoretical insights. For example, Harrison [22], Zhang and Weatherford [23], and Den Boer [24], [25], provided excellent reviews of the theoretical models. Based on these studies, some researchers have proposed dynamic pricing solutions for regular cloud services. Xu and Li [10] developed a stationary MDP to formulate the revenue maximization problem as a stochastic dynamic program. Furthermore, Alzhouri et al. [11], [26] further investigated dynamic pricing of stagnant resources using MDP, and used linear programming to approximate the stochastic dynamic programming.

However, existing pricing solutions heavily rely on traditional methods such as analytical models and deterministic rules for perishable services. These approaches may not be optimal and fail to adapt to the non-stationarity in the excess capacity. Applying deterministic rules to ever-changing non-stationary systems cannot guarantee optimality, and any changes in variables may result in significant financial losses. Additionally, the analytical models don't consider the preemption process, leading to potentially high preemption rates when directly applied. In this paper, we propose a framework to address these limitations on optimizing the dynamic pricing strategy with a certain maximum constraint on the preemption rate.

B. REINFORCEMENT LEARNING FOR DYNAMIC PRICING

Reinforcement Learning (RL) is a model-free algorithm that uses simulation-based stochastic techniques to solve complex optimization problems within the framework of MDPs [14]. It aims to learn the optimal policy for maximizing expected total rewards over time through trial and error. RL is particularly advantageous in handling MDP with complex or unknown transition mechanisms. It has become an important tool for solving dynamic pricing in uncertain environments. We focus on monopoly pricing, and these dynamic pricing problems are usually formulated as MDP. For, example, Raju et al. [27] used RL (e.g., QL algorithm) to price products dynamically with customer segmentation with limited available demand information. They considered an infinite horizon learning problem where there is no deadline for the sale of stock. Cheng [28] applied a QL algorithm for RL to solve dynamic pricing problems for selling a given stock with a finite horizon. The study investigated the pricing process and how an RL framework is used to set prices dynamically to adapt to uncertain demand and large-scale states. Rana and Oliveira [15], [16] proposed a model-free RL framework to solve dynamic pricing problems with a given inventory by a fixed deadline under inter-dependent and time-varying demand. They found that QL with eligibility traces outperforms the standard QL algorithm in non-stationary demand. Moreover, the QL algorithm also has been extensively utilized in dynamic pricing for the smart grid [17], [18], [29]. They showed that through ongoing learning and adaption, RL could be effectively employed to address the dynamic pricing of electricity systems with uncertainty and flexibility of users' power load.

RL has gained significant attention in the cloud industry, such as task scheduling [30], and resource allocation [31]. However, there has been limited research on using RL for dynamic pricing of cloud computing. For example, Cong et al. [32] studied the dynamic pricing of cloud computing in a context where user-perceived values concerning cloud services are dynamically changing and highly personalized. They developed an RL-based cloud pricing mechanism to learn sequential service pricing decision-making. However, cloud workloads typically exhibit constant and irregular changes in an unknown and unpredictable manner, resulting in non-stationary features. This poses challenges for RL, leading to imprecise learning outcomes and convergence issues.

To adapt to the non-stationary environment, we propose an improved QL algorithm that incorporates change detection into the RL process, enabling real-time recognition of pattern changes. By considering the deviation in interaction experience between different patterns, the learned policy can be improved. Additionally, focusing solely on revenue maximization may result in excessively high preemption rates. Hence, based on the detected temporal pattern, each learned pricing policy needs to adhere to the same maximum constraint on the preemption rate. To achieve this, the paper introduces the Lagrange multiplier method to incorporate the constrained preemption rate into the RL reward function. A bi-section search is employed to iteratively update the Lagrange multiplier until the optimal value that satisfies the constraint is found.

III. MODEL FORMULATION

A. MODEL ASSUMPTIONS

In this study, we consider a cloud operator and focus on a segment of customers who have fault-tolerant (interruptible) tasks and are sensitive to service prices. We consider a finite time horizon denoted as T, such as a 24-hour diurnal cycle and use the continuous-time index $t(0 \leq t \leq T)$ to represent time within this horizon. For the sake of clarity, before delving into our model details, we present the assumptions that are often aligned with the realistic operation of preemptible services.

1) USER REQUEST MODEL

In the preemptible market, users have the flexibility to schedule (add or eliminate) the PCS operating on fault-tolerant tasks according to the change in the price of the PCS. In line with previous studies [10], [11], we assume that consumers' demand for PCS is determined by two independent Poisson processes: an arrival process and a departure process. The arrival process models the newly coming requests of PCS as

a function of price, while the departure process corresponds to the leaving of the existing requests due to price changes.

We assume p_t represents the price per unit time one instance of PCS at time t. As a low-priority service, the price of PCS is lower than the price of the high-priority services. We assume that the price p_t can take any value from an interval $[0, p^{max}]$. The existence of a maximum price can be justified by the common-tiered service strategy in practice. For cloud computing, vendors adopt a two-tiered service strategy, that is, the regular service with the higher priority of service quality and preemptible service with lower priority. The price of a preemptible instance obviously cannot be higher than the price of the regular service. Without loss of generality, we let $p^{max} = 1$ throughout the paper.

Let $f(p_t)$ represent the Poisson arrival rate (expected value of the newly arrived units of requests per time unit). For $\forall 0 \leq p_t \leq 1$, we assume $f(p_t)$ satisfies the following properties [10].

$$f(p_t) \geq 0, f'(p_t) < 0, f''(p_t) < 0, f'(0) = 0, f'(1) = -\infty, \text{and } f(1) = 0 \quad (1)$$

where $0 \leq p_t \leq 1$. The above properties are consistent with reality, where 1) the arrival rate never be negative; 2) decreasing price p_t will attract more new users to use PCS. The Poisson departure process, denoted as $g(p_t)$, represents the expected number of users who leave the system per time unit at time t due to price adjustment. It quantifies the expected impact of price adjustments on the user departures. The departure process satisfies the following properties [10].

$$g(p_t) \geq 0, g'(p_t) > 0, g''(p_t) > 0, g'(0) 0, g'(1) = \infty, \text{ and } g(0) = 0 \quad (2)$$

We assume that the functions $f(p_t)$ and $g(p_t)$ can be estimated using widely studied methods in the field of online services. The specific details of the estimation methods are not provided in our research. Additionally, we do not consider preemptible risk and time factors in the demand functions. This is because cloud operators typically publish a fixed maximum constraint on the preemption rate for all users. This allows customers to form a common belief about the average preemption rate based on the disclosed level and historical preemption frequency observed by a third party, such as a cloud services consulting company. Furthermore, due to the weak time sensitivity of fault-tolerant tasks, users who have such tasks are highly price-sensitive. The price is the dominant factor that directly influences whether consumers choose to adopt the PCS, which leads to the demand for PCS is primarily driven by the price of the service rather than the specific arrival time of tasks.

2) NON-STATIONARY MODEL OF THE EXCESS CAPACITY

We partition the non-stationary stochastic process of excess capacity into a piecewise stationary model [33]. Time series analysis is widely used to model a non-stationary stochastic process. In our research, the time series is a sequence of observations of the unused capacity collected over time. According to the time series analysis, the behavior of a time series is studied as a function of its own past data. We assume that the stochastic process $\mathbf{c}_{1:T} = \langle c_1, c_2, \ldots, c_t, \ldots, c_T \rangle$ is the time series of the excess capacity, where $c_t (0 \leq c_t \leq C)$ denotes the excess capacity in time t, e.g., unused CPU capacity. Let $\mathbf{w}_{1:T} = \langle w_1, w_2, \ldots, w_t, \ldots, w_T \rangle$ denote the time series of the stochastic process for the time-varying workload of the high-priority service, where w_t represents workloads in time t, e.g., CPU usage. Thus, there is $c_t = \mathrm{R} - w_t$, where R denotes the total available capacity, e.g., the total capacity in one cluster for the operator. We represent the time series of the excess capacity as a Gaussian autoregressive (AR) model defined in Equation (3).

$$c_t = \mu_t + \phi \times c_{t-1} + \varepsilon_t \quad (3)$$

where $\phi (0 < \phi < 1)$ is constant value. $\mu_t \in R^+$ is changing with time t. We assume ε_t follows independent and identically Gaussian distribution with mean 0 and variance σ_t^2. The mean and variance of this time series are $\mathrm{E}(c_t) = \mu_t/(1-\phi)$ and $\mathrm{VAR}(c_t) = \sigma_t^2/(1-\phi^2)$.

Non-stationarity generally implies that the mean and/or variance of a time series are non-constant and vary over time, that is, they are dependent on time [33]. From the real-world workload traces shown in [3], the structural breaks ((e. g. level shifts or changing variances)) are common non-stationary structures in cloud computing. The structural breaks happen at specific points in time, usually due to environmental changes, such as unexpected marketing activities. We partition the uncertain stochastic process in Equation (3) into a piecewise stationary Gaussian AR model in Equation (4). Each stationary process with the same mean and variance is characterized as one pattern model. For notational brevity, we let $\theta_k = (\mu_k, \phi, \sigma_k)$ denote the vector of unknown parameters of pattern θ_k, and let $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_k, \ldots, \theta_K)$ denote the sequence set of unknown patterns in finite horizon T. K is the total number of the patterns. $t_{\theta_k} (1 \leq k \leq K)$ denotes the changepoint from pattern θ_k to another pattern θ_{k+1}. Herein, the time series of the excess capacity, $\mathbf{c}_{1:T} = (c_1, c_2, \ldots, c_t, \ldots, c_T)$, can be characterized as a piecewise stationary Gaussian AR model shown in Equation (4).

$$c_t = \begin{cases} \mu_1 + \phi \times c_{t-1} + \varepsilon_1, & 0 \leq t < t_{\theta_1} \\ \mu_2 + \phi \times c_{t-1} + \varepsilon_2, & t_{\theta_1} \leq t < t_{\theta_2} \\ \ldots & \\ \mu_K + \phi \times c_{t-1} + \varepsilon_K, & t_{\theta_{\mathrm{K}-1}} \leq t < t_{\theta_{\mathrm{K}}} \end{cases} \quad (4)$$

where μ_k and ε_k are constant values for each pattern θ_k, and $\varepsilon_k \sim \mathrm{N}(0, \sigma_k^2)$. In Eq. (4), the mean and variance of the pattern θ_k are $\mathrm{E}(c_t) = \mu_k/(1-\phi)$ and $\mathrm{Var}(c_t) = \sigma_k^2/(1-\phi^2)$. When we get (μ_k, ϕ, σ_k), the mean and variance value of pattern θ_k will be got. Take level shifts as an example, we set t_1 to be the changepoint of two stationary time series. If $t \leq t_1$, $\mu_t = \mu_1$; or else, $\mu_t = \mu_2$. In this example, there are two different

mean values $\mathrm{E}_1(c_t) = \mu_1/(1-\phi)$ and $\mathrm{E}_2(c_t) = \mu_2/(1-\phi)$ with one same variance $\mathrm{VAR}(c_t) = \sigma^2/(1-\phi^2)$.

However, because of the uncertainty of the workload of high-priority service, decision-makers are unable to predict when the current pattern shifts to another. For example, when a sudden business activity occurs, excess capacity will suddenly change into another unknown pattern model. Therefore, we make the following assumptions about the non-stationarity: (1) the pattern models change at least once (that is, $K \geq 2$), and the number of pattern model K is finite; (2) the pattern sequence set $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_k, \ldots, \theta_K)$, the vector of model parameters (μ_k, ϕ, σ_k) of each pattern θ_k, and the changepoint sequence set $(t_{\theta_1}, t_{\theta_2}, \ldots, t_{\theta_k}, \ldots, t_{\theta_K})$ are all unknown to the decision-maker and need to be estimated with ongoing time. The method of the detection of changepoint t_{θ_k} and the estimation of θ_k are described in detail in Section IV.

3) CONSTRAINED PREEMPTION RATE MODEL

Once the number of running preemptible instances exceeds the available excess capacity, the exceeded preemptible instances will be preempted, to alleviate service congestion and avoid the violation of the high-priority services. In our paper, we define ρ as the expected value of the actual preemption rate over the entire time horizon T. It represents the ratio of preempted instances to the total number of actively running instances at any given time. Let $q_t(p_t)$ represent the number of preempted instances at time t, and $d_t(p_t)$ represent the number of running preemptible instances at time t. The actual preemption rate, ρ, is subject to a maximum constraint, as indicated by Equation (5).

$$\rho = \mathbf{E}\left[\frac{\int_0^T q_t(p_t)\mathrm{d}t}{\int_0^T d_t(p_t)\mathrm{d}t}\right] \leq \varphi \tag{5}$$

Here, φ represents the maximum constraint on the actual preemption rate. This constraint sets an upper threshold on the allowed preemption rate. This constraint can help the operators control the high uncertainty in the actual preemption caused by the non-stationarity described in the above Section. The estimation method for the actual preemption rate, ρ, is described in Section IV.

B. CNSMDP MODEL FOR THE DPPCS

The objective of DPPCS is to determine the best price of the PCS to maximize the revenue generated from excess capacity within a maximum preemption rate constraint, in a given horizon T. Sequential decision-making problems with constraint are mostly formulated as Constrained Markov Decision Process (CMDP) [34]. Based on the CMDP, we formulate our problem as a family of CMDPs denoted as $\{M_{\theta_k}\}_{\theta_k \in \boldsymbol{\theta}}$, where $M_{\theta_k} = \langle \theta_k, S, A, P_{\theta_k}, R_{\theta_k}, \pi_{\theta_k}, \varphi \rangle$. When $K \geq 2$, we refer to this model $\{M_{\theta_k}\}_{\theta_k \in \boldsymbol{\theta}}$ as a Constrained Non-Stationary MDP (CNSMDP). The CNSMDP comprises multiple CMDP models. For each M_{θ_k} of CMDP models, the components are in general defined as follows. θ_k represents one specific temporal pattern. S is the set of states. A is the set of actions. P_{θ_k} is the transition probability function. R_{θ_k} is the reward function, which maps a state-action pair (s, a) to a real-valued reward. π_{θ_k} is the stationary policy for making decisions. φ is the constraint factor. An optimal policy $\pi^*_{\theta_k}$ can be found for each CMDP model. This optimal policy aims to maximize revenue of each pattern θ_k while satisfying the given constraints.

The detailed definitions of these elements in CNSMDP with discrete decision epochs are described as follows:

1) DECISION EPOCH N

We divide the time horizon T into N discrete epochs at which prices for the services are updated. Let $\Delta t (0 < \Delta t < T)$ be the interval for each decision epoch. Let $n (n = 1, 2, \ldots, N)$ denote the index of the decision epoch. N is the last time a price can be changed and at the end of this epoch, the unused capacity in horizon T cannot be stored and reused for later consumption. Discretizing the continuous-time environment has two main reasons. The first reason is that similar states are expected to have similar optimal price decisions and similar value functions. This simplification can make the pricing problem more tractable and computationally efficient. The second reason is that this discretization is more in line with practical applications, e.g., AWS and Ali Cloud update the price every five minutes.

2) PATTERN SET θ

$\boldsymbol{\theta}=(\theta_1, \theta_2, \ldots, \theta_k, \ldots, \theta_K)$ denotes the sequence set of unknown temporal patterns within a finite horizon T, e.g., $\boldsymbol{\theta}=(\theta_1, \theta_2, \theta_3)$, where $K = 3$ The value K denotes the total number of temporal patterns. Each temporal pattern θ_k is represented as $\theta_k = (\mu_k, \phi, \sigma_k)$. The set $\Upsilon = \{t_{\theta_k}\}_{1 \leq k \leq K}$ is the set of changepoints among the temporal patterns within the finite horizon T. If the detected changepoint t_{θ_k} falls within the interval $[(n-1) * \Delta t, n * \Delta t]$, where Δt represents the time interval, then the value of n is considered as the discrete index that corresponds to the specific changepoint t_{θ_k}.

3) STATE SPACE $\mathbf{S}$

The state set $s_n = (c_n, d_n)$ is defined to capture the supply-and-demand dynamics which can be observed before taking action. c_n represents the current available excess capacity (after satisfying the high-priority service) for the preemptible market at the initial epoch n. d_n represents the current running preemptible instances at the initial epoch n.

4) ACTION SET $\boldsymbol{A}$

The action set $\boldsymbol{A}$ represents a set of discrete action $a_n (n = 1, 2, \ldots, N)$. When $t \in [(n-1) \times \Delta t, n \times \Delta t]$, there is $p_t = a_n$. That is, action a_n is static in the interval $[(n-1) \times \Delta t, n \times \Delta t]$. As a low-priority service, the price of preemptible service is lower than the price of the high-priority service. We assume that the maximum preemptible price $p^{max} = 1$. The price of preemptible services a_n is in $[0, p^{max}]$. This is in line with the real-world applications, where the cloud vendors can only select prices from a finite list of admissible prices that are

lower than the low-priority service, e.g., $\mathbf{A} = \{0.1, 0.2, \ldots, 0.9, 1\}$.

5) TRANSITION PROBABILITY FUNCTION P_{θ_k}

$P_{\theta_k} = Pr\,(s_{n+1}|s_n, a_n)$ models the uncertain supply-and-demand evolution of the discrete states of the system based on the adopted action. It represents the probability from the observed supply-and-demand state $s_n = (c_n, d_n)$ to the next observed supply-and-demand state $s_{n+1} = (c_{n+1}, d_{n+1})$. However, the transition probability from s_n to s_{n+1} is unknown. For the supply part c_n in state s_n, when c_n and c_{n+1} are in different temporal patterns, the transition from c_n to c_{n+1} is unknown, especially since this transition is affected by external variables, not the price, which makes it difficult to estimate the probability of changed supply c_{n+1}. For the demand part d_n in state s_n, as the running preemptible instances, after adopting the price decision a_n, the changed demand d_{n+1} has an infinite possible number, which is because multiple arrivals and departures occur in each decision epoch n. Based on the above analysis, the transition probability $Pr\,(s_{n+1}|s_n, a_n)$ is hard to present with a mathematical model and is assumed to be unknown in our research.

6) ONE-STEP REWARD FUNCTION R_{n,θ_k}

: $R_{n,\theta_k}(\pi_{\theta_k})$ refers to the one-step reward which is used for evaluating the quality of each pricing decision. Under the condition of a fixed preemption rate constraint, the higher revenue generated from the excess capacity after taking the price decision, the better the taken price decision. We adopt the immediate revenue generated from the excess capacity as the reward for guiding price decisions. $d_n\,(a_n)$ denotes the total requested preemptible instances in epoch n. $q_{n,\theta_k}(a_n)$ refers to the total expected value of the number of instances being preempted in decision epoch n within interval Δt.$R_{\theta_k}\,(s_n, a_n)$ is calculated by Equation (6).

$$\begin{aligned} R_{\theta_k}\,(s_n, a_n) &= a_n \times \left[d_n\,(s_n, a_n) - q_{n,\theta_k}(s_n, a_n)\right] \times \Delta t \\ &= a_n \times [d_n + M\,[f\,(a_n)] - M\,[g\,(a_n)] \\ &\quad -q_{n,\theta_k}(s_n, a_n)] \times \Delta t \end{aligned} \tag{6}$$

The higher the revenue $R_{\theta_k}\,(s_n, a_n)$, the more encouraged to take the price decision a_n. Note that $q_{n,\theta_k}(a_n)$ is unknown to the decision-maker, which needs to be estimated when making the price decision a_n. $M\,[f\,(a_n)]$ is calculated by arrival rate $f\,(a_n)$, and $M\,[g\,(a_n)]$ is calculated by departure rate $g\,(a_n)$. $f\,(a_n)$ and $g\,(a_n)$ can be estimated and then assumed to be known by the decision-maker.

7) MAXIMUM PREEMPTION RATE CONSTRAINT

We define $\varphi(0 \leq \varphi < 1)$ as the maximum constraint on the probability of a running preemptible instance being preempted per unit of time. It represents the allowable maximum probability of an instance being preempted. In the realistic operation of the preemptible service, the value of φ is generally prescribed and disclosed by cloud operators. For example, AWS discloses different maximum values for the preemption rate in public for different types of preemptible instances: 5%, 10%, 15%, and 20%.[8] Ali Cloud prescribes the range of the preemption rate between 0 and 3% in all geographical regions published on the cloud trade platforms.[9]

8) PRICING POLICY SET π

$\boldsymbol{\pi} = \{\pi_{\theta_1}, \pi_{\theta_2}, \ldots, \pi_{\theta_k}, \ldots, \pi_{\theta_K}$ represents the set of randomized decision rules in the entire horizon T.π_{θ_k} is the short of $\pi_{\theta_k}\,(a_n|s_n)$, which denotes the probability of making a price decision a_n in state s_n at the k'th temporal pattern θ_k. The decision-maker might have to vary the pricing decision rule in different temporal patterns of the excess capacity being in. For example, it is instinctive that a high-pricing policy can be adopted in a temporal pattern where the excess capacity exhibits low expectation and high variance. Instead, a low-pricing policy can be adopted in a temporal pattern where the excess capacity exhibits high expectations and low variances.

The objective of the CNSMDP problem is to obtain an optimal pricing policy set $\boldsymbol{\pi}^* = (\pi^*_{\theta_1}, \pi^*_{\theta_2}, \ldots, \pi^*_{\theta_K})$. It maximizes the expected total revenue generated from the excess capacity while satisfying the maximum constraint φ. Let $J\,(\boldsymbol{\pi})$ represents the expected total revenue in the whole horizon T. Therefore, the objective of the CNSMDP can be formulated as Equation (7).

$$\begin{aligned} &\max_{\boldsymbol{\pi} \subset \boldsymbol{\Pi}} J(\boldsymbol{\pi}) = \max_{a_n \in A, \theta \in \boldsymbol{\theta}} \mathbf{E}\left[\sum\nolimits_{n=0}^{N} \gamma^n \times R_\theta\,(s_n, a_n)\right] \\ &\text{s.t. } \rho(\boldsymbol{\pi}) = \mathbf{E}_{\pi_\theta}\left[\sum\nolimits_{n=0}^{N} \rho_n\,(\pi_\theta)\right] \leq \varphi \end{aligned} \tag{7}$$

where $\gamma \in [0, 1]$ is the discount factor, $\boldsymbol{\Pi}$ is the set of all pricing policies. $\mathbf{E}$ is the expected value of the total revenue generated from excess capacity. The CNSMDP in Equation (7) is a constrained optimization problem. It is subject to a specified maximum constraint φ, which requires ensuring that the actual preemption rate caused by pricing policies $\boldsymbol{\pi}$ satisfies the specified constraint level. To convert the constrained optimization problem into an unconstrained one, we transform the CNSMDP problem into a piecewise Lagrangian dual problem.

C. PIECEWISE LAGRANGIAN DUAL MODEL

The CNSMDP in Equation (7) is composed of multiple CMDPs that are partitioned based on temporal patterns $\boldsymbol{\theta}$. Each CMDP belongs to an action CMDP. This is because the action, which is the price decision made in each state s_n and temporal pattern θ_k, is constrained by the maximum value φ. This kind of problem is commonly formulated as a Lagrangian problem [34]. Therefore, the CNSMDP in

[8] https://aws.amazon.com/cn/ec2/spot/instance-advisor/, accessed on June 19, 2023.

[9] https://help.aliyun.com/document_detail/52088.html?spm=5176.ecsnewbuy.help.dexternal.bbbc3675thWZtr, accessed on June 19, 2023.

Equation (7) can be converted into a piecewise Lagrangian problem defined in Equation (8).

$$\max_{\lambda_\theta \in \lambda, \pi_\theta \in \boldsymbol{\pi}} L(\pi_\theta, \lambda_\theta) = \max_{\lambda_\theta \in \lambda, \pi_\theta \in \boldsymbol{\pi}} J(\pi_\theta) - \lambda_\theta \times [\rho(\pi_\theta) - \varphi] \quad (8)$$

where λ_θ ($\lambda_\theta \in R_+$) represents the Lagrangian multiplier variable associated with the temporal pattern θ. Here, θ is a shorthand notation for θ_k, indicating the specific temporal pattern. $\boldsymbol{\lambda}$ is a vector denoted as $\boldsymbol{\lambda} = (\lambda_{\theta_1}, \lambda_{\theta_2}, \ldots, \lambda_{\theta_k}, \ldots, \lambda_{\theta_K})$. It represents the set of the Lagrangian multiplier variables for the entire horizon T. Similarly, $\boldsymbol{\pi}$ is a vector denoted as $\boldsymbol{\pi} \mathcal{D} (\pi_{\theta_1}(\lambda_{\theta_1}), \pi_{\theta_2}(\lambda_{\theta_2}), \ldots, \pi_{\theta_k}(\lambda_{\theta_k}), \ldots, \pi_{\theta_K}(\lambda_{\theta_K}))$, where $\pi_{\theta_k}(\lambda_{\theta_k}) \in \boldsymbol{\Pi}$. It represents the set of the corresponding optimal pricing policy associated with each specific temporal pattern θ_k and Lagrangian multiplier value λ_{θ_k}.

According to assumptions 1 and 2 in Appendix A, an optimal policy π_θ^* exists and constraint $\rho(\pi_\theta) \leq \varphi$ is satisfied. These assumptions are true in the real world; otherwise, the preemptible services will not be allowed to be offered. Since Assumptions 1 and 2 in Appendix A, the Slater's condition [34] holds. Therefore, the problem of Equation (8) is equivalent to its Lagrangian dual problem, which can be expressed as Equation (9).

$$\min_{\lambda_\theta \in \lambda} \max_{\pi_\theta \in \pi} L(\pi_\theta, \lambda_\theta) = \min_{\lambda_\theta \in \lambda} \max_{\pi_\theta \in \pi} J(\pi_\theta) - \lambda_\theta \times [\rho(\pi_\theta) - \varphi] \quad (9)$$

By solving for the Lagrange multiplier λ_θ and corresponding pricing policy π_θ in Equation (9), we can identify the optimal solution ($\boldsymbol{\lambda}^*$,$\boldsymbol{\pi}^*$) that satisfies the constraint conditions and maximizes the expected total revenue in the DPPCS problem stated in Equation (7). The optimal solution is denoted as $\boldsymbol{\lambda}^*=(\lambda_{\theta_1}^*, \lambda_{\theta_2}^*, \ldots, \lambda_{\theta_K}^*)$ and $\boldsymbol{\pi}^*=(\pi_{\theta_1}^*\left(\lambda_{\theta_1}^*\right), \pi_{\theta_2}^*\left(\lambda_{\theta_2}^*\right), \ldots, \pi_{\theta_K}^*(\lambda_{\theta_K}^*))$. The proof of the existence of the solution to the Lagrangian dual problem of Equation (9) is provided in Appendix A.

D. ADAPTING REINFORCEMENT LEARNING MODEL TO DPPCS

We adopt Reinforcement Learning and utilize its model-free advantage to solve the DPPCS model without knowledge of state transition probabilities. The DPPCS contains multiple CMDPs, where each CMDP corresponds to a specific temporal pattern θ_k. The unknown changepoints between temporal patterns result in unknown transitions between CMDPs. Furthermore, the transition probabilities of each CMDP are hard to be presented. Therefore, closed-form solutions and traditional dynamic programming techniques can not be adopted to solve the DPPCS. We convert the Lagrangian dual problem into the RL model, thereby incorporating the constraint conditions into the reward function. Equation (10) represents the conversion process. N_θ denotes the total number of decision epochs for each temporal pattern θ.

$$\begin{aligned} L(\pi_\theta, \lambda_\theta) &= J(\pi_\theta) - \lambda_\theta \times [\rho(\pi_\theta) - \varphi] \\ &= \mathrm{E}_{\pi_\theta}\left[\sum_{n \in N_\theta} R_n(\pi_\theta)\right] \\ &\quad - \lambda_\theta \times \left[\frac{\sum_{n \in \mathrm{N}_\theta} q_n(\pi_\theta)}{\sum_{n \in \mathrm{N}_\theta} d_n(\pi_\theta)} - \varphi\right] \\ &= \mathrm{E}_{\pi_\theta}\left\{\sum_{n \in N_\theta}\left[R_n(\pi_\theta) - \lambda_\theta \times \frac{q_n(\pi_\theta)}{d_n(\pi_\theta)}\right]\right\} \\ &\quad + \varphi \times \lambda_\theta \\ &= \mathrm{E}_{\pi_\theta}\left[\sum_{n \in N_\theta} R_n^{\lambda_\theta}(\pi_\theta)\right] + \varphi\lambda_\theta \end{aligned} \quad (10)$$

where $R_n^{\lambda_\theta}(\pi_\theta) = R_n(\pi_\theta) - \lambda_\theta \times \rho_{n,\theta}(\pi_\theta)$ is the reward function of the RL framework, which subsumes $\rho_{n,\theta}(\pi_\theta)$ into the revenue generated from excess capacity defined in Equation (6). Here, the Lagrange multiplier λ_θ is used to balance the revenue $R_n(\pi_\theta)$ and the preemption rate $\rho_{n,\theta}(\pi_\theta)$. Different from the common RL problem, there is a Lagrangian multiplier variable λ in the reward function of the RL model for DPPCS presented in Equation (11).

$$\max_{\lambda \subset \Omega, \pi \subset \Pi} J(\lambda, \boldsymbol{\pi}) = \max_{a_n \in A, \theta \in \boldsymbol{\theta}} \mathbf{E}\left[\sum_{n=0}^{N} \gamma^n \times R_n^{\lambda_\theta}(s_n, a_n)\right] \quad (11)$$

Therefore, the optimal policy changes with the value of λ. The objective of the DPPCS can be achieved by finding the optimal λ_θ^* and optimal pricing policy π_θ^* to maximize $\mathbf{E}_{\pi_\theta}\left[\sum_{n \in N_\theta} R_n^{\lambda_\theta}(\pi_\theta)\right]$ while satisfying constraint φ. We view this RL problem as illustrated in Figure 1, where the changes between temporal pattern $\theta_1, \theta_2, \ldots, \theta_K$ dynamically. By solving this RL problem in the entire horizon T, we can obtain the optimal Lagrange multiplier set $\boldsymbol{\lambda}^*=(\lambda_{\theta_1}^*, \lambda_{\theta_2}^*, \ldots, \lambda_{\theta_K}^*)$ and optimal pricing policy set $\boldsymbol{\pi}^*=(\pi_{\theta_1}^*\left(\lambda_{\theta_1}^*\right), \pi_{\theta_2}^*\left(\lambda_{\theta_2}^*\right), \ldots, \pi_{\theta_K}^*(\lambda_{\theta_K}^*))$.

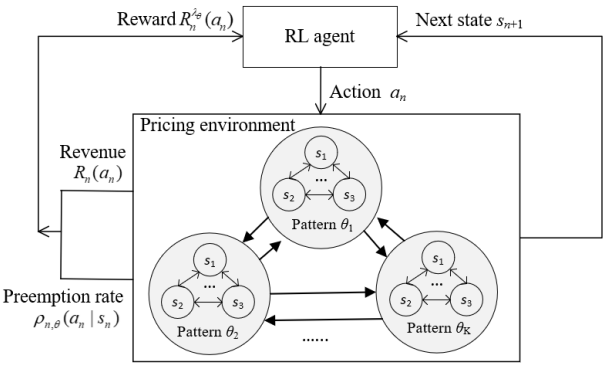

FIGURE 1. The interactions between the RL agent and the pricing environment.

Unfortunately, the epochs at which the temporal patterns changes θ occur are unknown to the RL agent. The RL agent can't perceive the temporal pattern change. In this case, to solve the RL problem, the RL is required to depend on

the sequence of all the historical states and actions observed up to epoch n, denoted as $h_n = (s_0, a_0, s_1, \ldots, a_{n-1}, s_n)$, where $h_n \in H_n$, and H_n is the set of all possible histories at epoch n. The algorithms designed to solve the RL problem presented in Equation (11) need to learn a policy $\pi(a_n|H_n)$ that maximizes the long-term expected total rewards, denoted as $\mathbf{E}\left[\sum_{n=1}^{N} \gamma^n R_n^{\lambda}\left(\pi(a_n|H_n)\right)|H_0 = h_0\right]$, for all initial histories $h_0 \in H_0$. However, multiple issues arise when solving the DPPCS compounded by the lack of model information.

(1) Due to the RL agent's inability to observe the changes in temporal patterns, the pricing policy only considers the current supply-and-demand state may not be optimal and will fall into constant oscillations. Thus, any algorithm for solving the DPPCS is needed to search over the space randomized history-dependent policies which is an intractable problem.

(2) Due to the maximum constraint on the preemption rate, the RL agent needs to identify the optimal Lagrange multipliers. It can strike a balance between maximizing revenue and adhering to the preemption rate constraint. Furthermore, accurately estimating the actual preemption rate is crucial because the RL agent cannot directly observe it.

IV. THE PROPOSED APPROACH

In this section, we explore the above issues and provide solutions to address them. We present the framework of the proposed novel dynamic pricing approach shown in Figure 2. The framework includes two main parts. The first part is the estimation methods for the unknown environment parameters, including a detection method for identifying temporal pattern changes, and a diffusion approximation method for estimating actual preemption rate. The second part is the Constrained Non-Stationary Q-Learning (CNSQL) algorithm for DPPCS, including updating the Lagrangian multiplier, learning the pricing policy given each Lagrangian multiplier, and evaluating the learned policy by estimating the actual preemption rate.

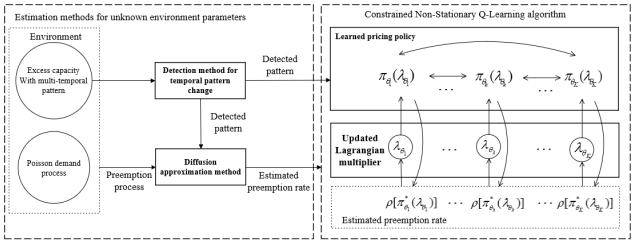

FIGURE 2. The framework of the novel dynamic pricing approach.

The proposed approach aims to learn the optimal Lagrangian multiplier set $\boldsymbol{\lambda}^*$ and optimal pricing policy $\boldsymbol{\pi}^*$, which can maximize the revenue generated from the excess capacity within the maximum constraint. By applying the detection method, the CNSQL algorithm can identify the change in the temporal pattern and adaptively update the learned pricing policy $\pi_{\theta_k}(\lambda_{\theta_k})$ for each pattern θ_k $(k = 1, 2, \ldots, K)$. The CNSQL learns the optimal pricing policy under each given Lagrangian multiplier λ_{θ_k}. Then, the learned policy $\pi^*_{\theta_k}(\lambda_{\theta_k})$ will be evaluated to identify whether the resulting actual preemption rate $\rho\left[\pi^*_{\theta_k}(\lambda_{\theta_k})\right]$ exceeds the maximum constraint or not. If $\rho\left[\pi^*_{\theta_k}(\lambda_{\theta_k})\right]$ exceeds the constraint, the Lagrangian multiplier will be updated, and then repeat the algorithm until the constraint is satisfied.

A. ESTIMATION METHODS FOR THE ENVIRONMENT PARAMETERS

1) DETECTION METHOD FOR ESTIMATING TEMPORAL PATTERN CHANGE

To assist the RL agent in identifying changes in the temporal pattern, we propose a detection method to estimate when a temporal pattern transitions into a different pattern, as well as the specific changed pattern model. The detection method incorporates a Bayesian Online Changepoint Detection (BOCD), and a Maximum Likelihood Estimation (MLE). BOCD is to identify whether the pattern changes. Then, MLE is adopted to estimate the model of the changed pattern. The changed pattern model is inputted into the QL algorithm to enable it to adapt to environmental change.

The BOCD proposed by Adams and MacKay [35] is the most popular sequential changepoint algorithm. The BOCD does not rely on the asymptotic assumptions about test statistics that are presented in the frequentist algorithm. Furthermore, the Bayesian approach can ignore the occasional abrupt peak or valley points and other abnormal points that do not affect the pattern change, but are extremely sensitive to the pattern change, which is well matched with the highly non-stationary workload [3]. Let τ_t denote the time length from the last changepoint to the current time t. Let $\mathbf{c}_{1:t}$ represent the set of observed data from the beginning of the time series data to the current time t. Having observed previous data points $\mathbf{c}_{1:t-1}$, the run length τ_{t-1} indicates whether the new datum c_t still belongs to the same pattern. If c_t still belongs to the same pattern, then $\tau_t = \tau_{t-1}+1$, otherwise, $\tau_t = 0$ and time t is a changepoint. The probability of τ_t based on the historical observed data $\mathbf{c}_{1:t}$ at every time t is calculated by Equation (12).

$$\begin{aligned}&\Pr(\tau_t, \mathbf{c}_{1:t})\\ &= \sum\nolimits_{\tau_{t-1}} \Pr\left(\tau_t|\tau_{t-1}\right)\Pr\left(w_t|\tau_{t-1}, \mathbf{c}_t^r\right)\Pr\left(\tau_{t-1}, \mathbf{c}_{1:t-1}\right)\end{aligned} \tag{12}$$

where $\Pr(\tau_t, \mathbf{c}_{1:t})$ includes two kinds of probabilities which are $\Pr(\tau_t = 0, \mathbf{c}_{1:t})$ and $\Pr(\tau_t = \tau_{t-1}+1, \mathbf{c}_{1:t})$. $\Pr(\tau_t = 0, \mathbf{c}_{1:t})$ is the rate of being a changepoint in time t, and $\Pr(\tau_t = \tau_{t-1}+1, \mathbf{c}_{1:t})$ is the rate of not being a changepoint in time t. In Equation (11), joint distribution $\Pr(\tau_t, \mathbf{c}_{1:t})$ is calculated by a recursive message-passing algorithm, which is constructed by two parts: (1) $\Pr\left(\tau_t|\tau_{t-1}\right)$ is calculated by hazard function, which is assumed as a discrete exponential distribution with timescale ϱ; (2) $\Pr\left(c_t|\tau_{t-1}, \mathbf{c}_t^r\right)$ is the prediction probability over the newly-observed datum c_t, give the data since the last changepoint $\mathbf{c}_t^{\tau}$.

We adopt MLE to obtain the latest changed pattern model $\theta = \left(\hat{\mu}, \hat{\phi}, \hat{\sigma}\right)$ by using the observations of the current changed pattern. We choose a sample of size L from the latest detected pattern to estimate the unknown parameters. According to section III, each stationary structure is modeled as Gaussian AR, that is, $c_t = \mu + \phi \times c_{t-1} + \varepsilon$. The log-likelihood for observations size L from the Gaussian AR process is presented in Equation (13).

$$\mathcal{L}(\theta) = -\frac{1}{2}\log\left(\frac{\sigma^2}{1-\phi^2}\right) - \frac{\left\{c_{t_\theta} - \left[\frac{\mu}{1-\phi}\right]\right\}^2}{\frac{2\sigma^2}{1-\phi^2}} - \frac{L}{2}\log(6.28) - \left(\frac{L-1}{2}\right)\log\left(\sigma^2\right) - \sum_{l=1}^{L}\left[\frac{\left(c_{t+l+1} - \mu - \phi c_{t+l}\right)^2}{2\sigma^2}\right] \tag{13}$$

We need to estimate the vector $\theta = \left(\hat{\mu}, \hat{\phi}, \hat{\sigma}\right)$ to maximize the log-likelihood $\mathrm{L}(\theta)$. However, the partial derivatives of the argmax($\mathrm{L}(\theta)$) in each parameter of vector θ are nonlinear, making it difficult to give analytical solutions of the pattern model $\theta = \left(\hat{\mu}, \hat{\phi}, \hat{\sigma}\right)$ by using a sample of size L. Therefore, we adopt numerical maximization to approximately obtain the estimated pattern model $\theta = \left(\hat{\mu}, \hat{\phi}, \hat{\sigma}\right)$ by grid search method.

The full algorithm is shown in Algorithm 1. The posterior probability distribution is calculated online based on the recursion message passing algorithm with the real-time generated stream data. Because it is very hard to correctly evaluate a change and the changed pattern model after a single sample of a new distribution, we instead set "delayed length" for L samples and evaluate the probability of a change happening L samples prior. The value of delayed samples depends on the characteristics of the non-stationary structure. If the non-stationary structure changes significantly, the value of L can be set smaller.

Algorithm 1 Detection algorithm for the temporal pattern changes of the excess capacity

1: **Input**: Time horizon T, time series data $\mathbf{c}_{1:T} = [c_1, c_2, \ldots, c_T]$; prior probability distribution of the τ_t; hazard function; threshold η; delayed samples L.
2: **Output** $\boldsymbol{\Upsilon}$, $\boldsymbol{\theta}$**:**Changepoint set $\boldsymbol{\Upsilon}$ $\boldsymbol{\mathcal{D}}$ $\{t_{\theta_k}\}_{1\le k\le K}$ and pattern set $\boldsymbol{\theta}$ $\boldsymbol{\mathcal{D}}$ $(\theta_1, \theta_2, \ldots, \theta_K)$.
3:**for** c_t in $\mathbf{c}_{1:t}$ **do**
4: Pr $\left(\tau_t|\tau_{t-1}\right)$ ← Evaluate the hazard function for each possible τ_t from Bayesian inference;
5: Pr $\left(c_t|\tau_{t-1}, \mathbf{c}_t^r\right)$ ← Evaluate the predictive distribution for new datum c_t from Bayesian inference;
6: Pr$(\tau_t, \mathbf{c}_{1:t})$ ← Determine posterior joint probability distribution for each possible τ_t using Equation (12);
7: **if** Pr$(\tau_t, \mathbf{c}_{1:t}) > \eta$ **then**
8: **Return**changepoint, $t_{\theta_k} = t - L$
9: $\hat{\theta}_k = \left(\hat{\mu}_k, \hat{\phi}, \hat{\sigma}_k\right)$ ← Estimate $\hat{\theta}_k$ with data $\mathbf{c}_{t-L:t}$ using Equation (13)
10: **else**
11: No changepoint, **Return**

2) DIFFUSION APPROXIMATION METHOD FOR ESTIMATING PREEMPTION RATE

To help the RL agent balance the cloud revenue and actual preemption rate, we adopt the diffusion approximation technique to estimate the actual preemption rate $\rho_{n,\theta}(\pi_\theta)$ in discrete decision epoch. The preemption occurs, once the demand of preemptible instances exceeds the excess capacity offered, and the preempted instances can be resumed later when resources become available. The preemption process is a continous-time and is at a finer time granularity, compared to the discrete decision epoch at which the price is determined. The preemption rate needs to be calculated by considering two stochastic processes, that is, the time-varying excess capacity and the demand for the preemptible services.

The preemption process is considered a priority-queueing system with preemption, which is continuous and intractable with large demand and supply. Therefore, an appropriate diffusion process, called reflected Brownian motion process, is adopted to simulate the preemption process. The modeling framework is developed by Harrison [22]. It is commonly used to approximate the behavior of G/G/C queues under heavy traffic conditions [36].

For each discrete epoch n, we set continuous time t' to be in $[0, \Delta t]$. We first calculate the mean and variance of the stochastic process of excess capacity. Based on the estimated pattern model $\theta = \left(\hat{\mu}, \hat{\phi}, \hat{\sigma}\right)$, we can get the mean and variance of the current pattern: $\mathbf{E}(c_{t'})$ and $\mathbf{Var}(c_{t'})$. Let $C_\theta\left(t'\right)$ be the cumulative excess capacity in the interval $[0, t']$ for pattern θ. Therefore, the mean and variance of $C_\theta\left(t'\right)$ in $[0, t']$ is as follows.

$$\begin{cases} \mathbf{E}\left[C_\theta\left(t'\right)\right] = \dfrac{\hat{\mu}}{\left(1-\hat{\phi}\right)}t' \\ \mathbf{Var}\left[C_\theta\left(t'\right)\right] = \dfrac{\hat{\sigma}^2}{(1-\hat{\phi}^2)}t' \end{cases} \tag{14}$$

Let $D_n\left(t'\right)$ be cumulative running preemptible instance in the interval $[0, t']$. So, the mean and variance of $D_n\left(t'\right)$ in $[0, t']$ are as follows.

$$\begin{cases} \mathbf{E}\left[D_n\left(t'\right)\right] = d_n t' + \left[f\left(\pi_\theta\right) - g\left(\pi_\theta\right)\right]t' \\ \mathbf{Var}\left[D_n\left(t'\right)\right] = \left[f\left(\pi_\theta\right) - g\left(\pi_\theta\right)\right]t' \end{cases} \tag{15}$$

where π_θ is short of $\pi_\theta(a_n|s_n)$, d_n is the running preemptible instances at the initial epoch n. We set $B_{n,\theta}\left(t'\right)$ as a stochastic process. $B_{n,\theta}\left(t'\right) = D_n\left(t'\right) - C_\theta\left(t'\right)$. The mean $\mathbf{E}\left[B_\theta\left(t'\right)\right]$ is calculated as $\mathbf{E}\left[D_n\left(t'\right)\right] - \mathbf{E}\left[C_\theta\left(t'\right)\right]$, and the variance $\mathbf{Var}\left[B_\theta\left(t'\right)\right]$ is calculated as $\left[D_n\left(t'\right)\right] + \mathbf{Var}\left[C_\theta\left(t'\right)\right]$. The values of $\mathbf{E}\left[B_\theta\left(t'\right)\right]$ and $\mathbf{Var}\left[B_\theta\left(t'\right)\right]$ are presented as follows.

$$\begin{cases} \mathbf{E}\left[B_\theta\left(t'\right)\right] = d_n t' + \left[f\left(\pi_\theta\right) - g\left(\pi_\theta\right)\right]t' - \dfrac{\hat{\mu}}{\left(1-\hat{\phi}\right)}t' \\ \mathbf{Var}\left[B_\theta\left(t'\right)\right] = \left[f\left(\pi_\theta\right) - g\left(\pi_\theta\right)\right]t' + \dfrac{\hat{\sigma}^2}{(1-\hat{\phi}^2)}t' \end{cases} \tag{16}$$

According to the literature [22], the stochastic process $B_\theta\left(t'\right)$ can be described as Brownian motion with floating terms. The stochastic differential equation of $B_\theta\left(t'\right)$ is $\mathbf{d}B_\theta\left(t'\right) = \mu_{\theta,b}\mathbf{d}t' + \sigma_{\theta,b}\mathbf{d}W(t')$, where $W(t')$ is the standard Wiener process, $\mu_{\theta,b}$ is the drifting term of Brownian motion $B_\theta\left(t'\right)$, and $\sigma_{\theta,m}$ is the standard variance of Brownian motion $B_\theta\left(t'\right)$. Then, the drifting term and the standard variance of Brownian motion $B_\theta\left(t'\right)$ are as follows

$$\begin{cases} \mu_{n,\theta} = \left[f\left(\pi_\theta\right) - g\left(\pi_\theta\right)\right] - \frac{\hat{\mu}}{\left(1-\hat{\phi}\right)} + d_n \\ \sigma_{n,\theta} = \sqrt{\left[f\left(\pi_\theta\right) - g\left(\pi_\theta\right)\right] + \frac{\hat{\sigma}^2}{(1-\hat{\phi}^2)}} \end{cases} \tag{17}$$

Let $q_\theta\left(t'\right)$ be the amount of preempted instances (and to be resumed later) at the point of time t, has been proven to be a reflected Brownian motion given $B_\theta\left(t'\right)$, also known as a Brownian motion modified by a lower reflecting barrier at zero given $B_\theta\left(t'\right)$. The simple derivation is: $q_\theta\left(t'\right) = B_\theta\left(t'\right) - L_\theta\left(t'\right)$, where $L_\theta\left(t'\right) = \left[B_\theta(x)\right]^- = -\mathbf{min}\left[B_\theta\left(x\right), 0\right]^-$. If $q_\theta\left(t'\right) < 0$, no preemption occurs; If $q_\theta\left(t'\right) > 0$, preemption occurs. The higher of $q_\theta\left(t'\right)$, then the higher amount being preempted. Based on the Brownian motion theory in literature [22], the cumulative distribution function of $q_{n,\theta}$ is as follows.

$$\begin{aligned} \Pr\left[q_{n,\theta} \le q\right] = \Phi\left(\frac{q - \mu_{n,\theta}t'}{\sigma_{n,\theta}\sqrt{t'}}\right) \\ - e^{\frac{2\mu_{n,\theta} * q}{\sigma^2_{n,\theta}}} \Phi\left(\frac{-q - \mu_{n,\theta}t'}{\sigma_{n,\theta}\sqrt{t'}}\right) \end{aligned} \tag{18}$$

where $\Phi\left(\cdot\right)$ is the cumulative distribution function of a standard normal distribution. When the value t' is high, the probability distribution of the amount of preempted instances is equal to the following equation.

$$\Pr\left[q_{n,\theta}\left(t'\right) \le q\right] \to \begin{cases} 1 - e^{\frac{2\mu^q_{n,\theta}}{\left(\sigma_{n,\theta}\right)^2}}, \mu_{n,\theta} < 0 \\ 0, \mu_{n,\theta} \ge 0 \end{cases} \tag{19}$$

It can be seen from Equation (18) that when $\mu_{n,\theta} < 0$, $\Pr\left[q_{n,\theta}\left(t'\right) \le q\right]$ is an exponential distribution. If, on average, the total workload of the fault-tolerant jobs exceeds the surplus capacity (e.g., $\mu_{n,\theta} \ge 0$), then the amount of workload preempted goes to infinity in the long run. This is not a relevant case. We thus focus on the case in which $\mu_{n,\theta} < 0$. The average value of the amount of preempted instances per unit time at decision epoch n is shown respectively in Equation (20).

$$\mathbf{E}\left[q_{n,\theta}\right] = \frac{\left[f\left(\pi_\theta\right) - g\left(\pi_\theta\right)\right] + \frac{\hat{\sigma}^2}{(1-\hat{\phi}^2)}}{2\left\{\frac{\hat{\mu}}{(1-\hat{\phi})} - d_n - \left[f\left(\pi_\theta\right) - g\left(\pi_\theta\right)\right]\right\}} \tag{20}$$

The actual amount of preempted instances will be simulated by exponential distribution with *rate parameter* estimated in Equation (20). Therefore, the expected value of the actual preemption rate per unit time at epoch n, $\rho_{n.\theta}$, is estimated using the ratio $\frac{\mathbf{E}\left[q_{n,\theta}\right]}{\mathbf{E}\left[d_{n,\theta}\right]}$. The calculation of $\rho_{n.\theta}$ is as follows:

$$\begin{aligned} &\rho_{n.\theta} \\ &= \frac{\left[f\left(\pi_\theta\right) - g\left(\pi_\theta\right)\right] + \frac{\hat{\sigma}^2}{(1-\hat{\phi}^2)}}{2\left\{\frac{\hat{\mu}}{(1-\hat{\phi})} - d_n - \left[f\left(\pi_\theta\right) - g\left(\pi_\theta\right)\right]\right\}\left[d_n + f\left(\pi_\theta\right) - g\left(\pi_\theta\right)\right]} \end{aligned} \tag{21}$$

B. CNS-QL ALGORITHM FOR DPPCS

We present the details of the proposed CNSQL algorithm, which aims to find the optimal Lagrangian multiplier λ^*_θ and the corresponding optimal pricing policy $\pi^*_\theta(\lambda^*_\theta)$. π^*_θ refers to how to make price decisions in each pattern θ, which can maximize the revenue generated from excess capacity within maximum constraint φ in finite horizon T.

1) LAGRANGIAN MULTIPLIER UPDATING METHOD

A bisection method for updating Lagrangian multiplier is applied to find the optimal value λ^*_θ satisfying the constraint φ. This method is based on the theoretical deductions (see Theorem 1 in Appendix A that the actual preemption rate $\rho\left[\pi_\theta(\lambda_\theta)\right]$ is monotonic with Lagrangian multiplier λ_θ. That is, for each pattern θ, the higher the value of λ_θ, the lower the value of $\rho\left[\pi_\theta(\lambda_\theta)\right]$. We set the lower bound λ^l_θ and upper bound λ^u_θ of λ_θ. In the bisection method, the Lagrangian multiplier λ^m_θ is calculated by the middle point of λ^l_θ and λ^u_θ. If the expected value of the actual preemption rate, $\rho\left[\pi^*_\theta(\lambda^m_\theta)\right]$, is beyond the constraint condition. The Lagrangian multiplier will be updated as presented in Equation (22).

$$\lambda^m_\theta \leftarrow \begin{cases} \frac{\lambda^l_\theta + \lambda^m_\theta}{2}, \text{if } \rho\left[\pi^*_\theta(\lambda^m_\theta)\right] < \varphi - \zeta \\ \frac{\lambda^m_\theta + \lambda^u_\theta}{2}, \text{if } \rho\left[\pi^*_\theta(\lambda^m_\theta)\right] > \varphi \end{cases} \tag{22}$$

where $\varphi - \zeta$ denotes the lower bound of the preemption rate. When the expected preemption rate $\rho\left[\pi^*_\theta(\lambda^m_\theta)\right]$ satisfies the maximum constraint, the algorithm ends and outputs the optimal pricing policy set $\boldsymbol{\pi}^*$ and Lagrangian multiplier $\boldsymbol{\lambda}^*$.

2) ALGORITHM DESIGN

The detailed steps of the proposed CNSQL algorithm for DPPCS are presented in Algorithm 2. The algorithm mainly includes outer and inner loops. The outer loop is to update the Lagrangian multiplier to find the optimal Lagrangian multiplier with the updating process presented in Equation (22). The inner loop is for learning the optimal pricing policy of each given Lagrangian multiplier, and evaluating the learned pricing policy.

As depicted in Algorithm 2, the initial step involves providing a time series dataset denoted as $\mathbf{c}_{1:T}$. The data points represent observations collected over a specific horizon T. The time series data has a pattern sequence $\theta_1 \rightarrow \theta_2 \rightarrow \ldots \rightarrow \theta_K$, where θ_k represents the parameter vector of the k'th pattern. Note that the RL agent doesn't know the

sequence pattern set, $\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots, \theta_k, \ldots, \theta_K\}$ and the changepoints $\boldsymbol{\Upsilon} = (t_{\theta_1}, t_{\theta_2}, \ldots, t_{\theta_k}, \ldots, t_{\theta_K})$. However, the RL agent can know the total number of pattern types, denoted as I. The agent needs to learn $\boldsymbol{\theta}$ and $\boldsymbol{\Upsilon}$ with newly obtained data. In addition, we input the demand information, $f(a_n)$ and $g(a_n)$. The initial Q-value $Q(\theta_i, s, a)$, as well as the lower and upper bounds $\left(\lambda^l_{\theta_i}, \lambda^u_{\theta_i}\right)$, are set for each pattern i ($i \in (1, 2, \ldots, I)$). i refers to the type of temporal pattern. When the algorithm ends, it outputs the optimal pricing policy $\boldsymbol{\pi}^*$ and optimal Lagrangian multiplier $\boldsymbol{\lambda}^*$.

The outer loop from Step 3 to Step 24 shows the Lagrangian multiplier updating process. Step 3 identifies the updated Lagrangian multiplier $\lambda^m_{\theta_i}$ calculated by Equation (21). When the expected value of the actual preemption rate $\rho\left[\pi^*_\theta(\lambda^m_\theta)\right]$, satisfies the maximum constraint, the algorithm ends and outputs the optimal pricing policy $\pi^*_{\theta_i}$ and Lagrangian multiplier $\lambda^*_{\theta_i}$ for each pattern θ_i ($i = 1, 2, \ldots, I$). If the current Lagrangian multiplier $\lambda^m_{\theta_i}$ is not within the constraint, the next move to update the Lagrangian multiplier is identified according to Equation (21). In this updating process, $\pi^*_\theta(\lambda^m_\theta)$ is learned from Steps 4 to 18, and $\rho\left[\pi^*_\theta(\lambda^m_\theta)\right]$ is estimated from Step 19-Step 24.

The policy learning part of the inner loop is from Step 4 to Step 18. It aims to learn the optimal pricing policy under each updated Lagrangian multiplier $\lambda^m_{\theta_i}$. The policy learning phase outputs the converged $Q^*\left(\theta_i, \lambda^m_{\theta_i}, s, a\right)$ or the pricing policy $\pi^*_{\theta_i}(\lambda^m_{\theta_i})$. Based on the QL algorithm (presented in Appendix B), the agent is combined with the detection method of pattern change (presented in Algorithm 1), the estimation method for the preemption rate (Equation (21)), and the updated Lagrangian multiplier λ_θ. The estimated Q-value with a specific λ_θ for each pattern, θ, is modified as Equation (23)

$$Q(\theta, \lambda_\theta, \pi_\theta) \leftarrow Q(\theta, \lambda_\theta, \pi_\theta) + \delta\left[R^{\lambda_\theta}(\pi_\theta) + \gamma Q(\theta, \lambda_\theta, \pi_\theta) - Q(\theta, \lambda_\theta, \pi_\theta)\right] \quad (23)$$

where π_θ is short of $\pi_\theta(a_n|s_n)$. Once the changepoint of the temporal pattern is detected by algorithm 1, the learning process of the Q-value will change into the next pattern. Then, the Q-value table of the changed pattern will be learned according to the experience (e. g. $R^{\lambda_\theta}(\pi_\theta)$) from the changed pattern. If the changed pattern has already appeared in the historical epochs, then the Q-value table will be updated based on the learned Q-value table in historical epochs without affecting the Q-values of other patterns.

The policy evaluation part of the inner loop is from Step 19 to Step 24. It aims to evaluate the expected value of the actual preemption rate, $\rho\left[\pi^*_\theta(\lambda^m_\theta)\right]$ calculated by Equation (24).

$$\rho\left[\pi^*_\theta(\lambda^m_\theta)\right] = \frac{\sum_{n \in N_\theta} \rho_n\left(\pi^*_\theta(\lambda^m_\theta)\right)}{N_\theta \times M_{max}} \quad (24)$$

If the $\rho\left[\pi^*_\theta(\lambda^m_\theta)\right]$ is within the maximum constraint, λ^m_θ will be the optimal value. If $\rho\left[\pi^*_\theta(\lambda^m_\theta)\right]$ is beyond the maximum constraint, λ^m_θ will be updated according to Equation (22). Then, a new Lagrangian multiplier is obtained, and repeat the policy learning process (Step 4-Step 18) until the agent finds the optimal Lagrangian multiplier λ^*_θ which can satisfy the maximum constraint.

Algorithm 2 CNSQL algorithm for DPPCS

1: **Input**: Time series data $\mathbf{c}_{1:T}$ with pattern sequence $\theta_1 \rightarrow \theta_2 \rightarrow \ldots \rightarrow \theta_K$, $f(a_n)$, $g(a_n)$, $Q(\theta_i, s, a) = 0$, $\lambda^l_{\theta_i}, \lambda^u_{\theta_i}$, $i = (1, 2, .., I)$, maximum preemption rate φ, epoch size N.
2:**Output**: Optimal pricing policy set $\boldsymbol{\pi}$ $\boldsymbol{\mathcal{D}}$ $\{\pi^*_{\theta_i}\left(\lambda^*_{\theta_i}\right)\}_{1 \le i \le I}$ and Lagrangian multiplier set $\boldsymbol{\lambda} = \{\lambda^*_{\theta_i}\}_{1 \le i \le I}$.
3:**Repeat:**Identify the updated Lagrangian multiplier according to Equation (22). *# Policy learning phase*
4: **for**episode = 1 to M_{max} **do**
5: Reset state s_0, $\boldsymbol{\theta}$, $\boldsymbol{\Upsilon}$, i, k
6: **for** n= 1 to N **do**
7: Select an action a_n according to epsilon-greedy.
Execute a_n
and obtain revenue R_n by Equation (6);
8: Estimate the ρ_{n,θ_i} by Equation (21), and calculate reward $R^{\lambda_{\theta_i}} = R_n - \lambda^m_{\theta_i} \times \rho_{n,\theta_i}$. Go into the next state s_{n+1};
9: Update the $Q\left(\theta_i, \lambda^m_{\theta_i}, s_n, a_n\right)$ according to Equation (23);
10: Identify $t_{\theta_{k+1}}$ and θ_{k+1} according to **Algorithm 1**.
11: **if**$t_{\theta_{k+1}}$ is not Null **then**
12: Increment θ_{k+1} into $\boldsymbol{\theta}$, and increment $n_{\theta_{k+1}}$ into $\boldsymbol{\Upsilon}$.
13: **if** θ_{k+1} is in $\boldsymbol{\theta}$**then**
14: $\theta_{i'} = \theta_{k+1}$;
15: $i = i'$.
16: **else**
17: $\theta_{i+1} = \theta_{k+1}$;
18: $i = i + 1$.
 # Policy evaluation phase
19: **for**episode = 1 to M_{max} **do**
20: Reset state s_0, n.
21: **for** $n = n_{\theta_k}$ to $n_{\theta_{k+1}}$ **do**
22: Choose action a_n from learned policy $\pi^*_{\theta_i}\left(\lambda^m_{\theta_i}\right)$ from *policy learning phase*;
23: Execute a_n and estimated the ρ_{n,θ_i} according to Equation (21).
24: Calculate $\rho\left[\pi^*_{\theta_i}(\lambda^m_{\theta_i})\right]$ of each pattern θ_i according to Equation (24), then go to Step 3.

According to the proposed CNSQL algorithm in a non-stationary environment, the optimal dynamic pricing π^*_θ and optimal Lagrangian multiplier λ^*_θ of each pattern, θ, are learned. The learned optimal policies π^*_θ are stored in a pattern-policy library. In the real-world pricing environment, the agent can access the learned pricing policy π^*_θ stored in the pattern-policy library without re-learn the policy when the temporal pattern changes.

V. EXPERIMENTS

In this section, we mainly conduct experiments to verify the performance of our proposed approach. We mainly study three questions: (1) Does the proposed detection method of temporal pattern change in our research can recognize the pattern change online and estimate the changed model accurately? (2) How does the proposed CNSQL algorithm

perform on the expected total revenue and actual preemption rate in comparison with other baseline algorithms? (3) How does the proposed CNSQL algorithm in different temporal patterns affect the expected total revenue and actual preemption rate? To answer these questions, a set of experiments were conducted on both synthetic and real-world non-stationary scenarios.

A. ENVIRONMENT SETUP

To simulate the demand process for the preemptible services, we adopt properties assumed in Section III. The demand functions are assumed to be known by the decision-maker. Without loss of generality, we set the arrival rate $f(a_n)$ to be $f(a_n) = \eta(1 - a_n^{\kappa})^{\omega}$, and the departure rate $g(a_n)$ to be $g(a_n) = \vartheta - \vartheta(1 - a_n^{\kappa})^{\omega}$. η represents the potential maximum value of arrival rate per time unit. ϑ represents the maximum value of the departure rate per time unit. Note that when $a_n = 0$, the lowest price, then $f(0) = \eta$ and $g(0) = 0$. $f(0) = \eta$ represents that all of the fault-tolerant tasks in the preemptible market will use preemptible services with no pay. At the same time, the departure rate $g(0) = 0$ theoretically shows that there is no incentive for consumers to leave preemptible services. But when $a_n = 1$, the highest price, the $f(1) = 0$ and $g(1) = \vartheta$. $f(1) = 0$ and $g(1) = \vartheta$ represent that when the price of preemptible service is equal to the price of the high-priority service, no tasks are running on the preemptible service. We set discrete action set A to be $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$, that is, the decision at each epoch n, denoted as a_n, can take on one of ten possible values.

We simulate the resource supply model by Gaussian AR assumed in Section III. We set the total capacity to 100. Two types of temporal pattern models are simulated: one is $c_t = 30 + 0.1c_{t-1} + \varepsilon_1$, where $0 \leq t \leq 34$ and $81 < t \leq 100$. The Gaussian noise term ε_1 follows $N(0, 10^2)$; The other one is $c_t = 80 + 0.1c_{t-1} + \varepsilon_2$, where $34 < t \leq 81$. The Gaussian noise term ε_2 follows $N(0, 5^2)$. In the first pattern, the mean and variance of c_t are as follows: $\mathrm{E}_1(c_t) = 77.7$ and $\mathrm{VAR}_1(c_t) = 10.01$. In the second pattern, the mean and variance of c_t are as follows: $\mathrm{E}_2(c_t) = 22.4$ and $\mathrm{VAR}_2(c_t) = 5.05$. The values of the demand parameters are simulated as follows: $\eta = 50$, $\omega = 3$, $\kappa = 0.5$, $\vartheta = 50$. Additionally, the number of decision epochs, N, is equal to 20.

Furthermore, we evaluate the practicability of our proposed approach by adopting real-world data from the widely-used Google CPU usage Trace dataset.[10] The reasons for using the cluster-usage dataset are as follows: (1) The dataset contains a time series for the resource usage of the high-priority service, spanning 29 days within a cluster. This dataset can be utilized to calculate the excess capacity after satisfying the workload of the high-priority service; (2) CPU utilization is the key significant metric for the workload on instances and it exhibits extreme non-stationarity [3]. Then, the excess capacity of the CPU is calculated as $c_t = \mathrm{R} - w_t$, which is described in Section III. The values of the demand parameters are simulated as follows: $\eta = 2000$, $\omega = 5$, $\kappa = 2.5$, $\vartheta = 2000$. The number of decision epochs, N, is equal to 15.

[10]Google, 2016. Google cluster dataset. URL: https://github.com/google/cluster-data.

B. BASELINES AND EVALUATION METRICS

To investigate the proposed approach, several alternatives are used to demonstrate the advantages and disadvantages of different approaches, which are briefly introduced as follows:

(1) Baseline non-stationary QL algorithms without considering changepoint detection method. We compare our proposed algorithm with two state-of-the-art algorithms, QL with eligibility traces [15], and Repeated Updated QL (RUQL) [37]. QL with eligibility traces has been validated that it has a better performance than classical QL in a non-stationary environment. Repeated Updated QL (RUQL) is a variant of the QL algorithm, which essentially repeats the updates to the Q-values of a state-action pair inversely proportional to the rate of choosing that action. RUQL has also been confirmed from experiment and theoretical perspectives that RUQL outperforms classical QL algorithms in noisy non-stationary environments. To ensure a fair comparison, we add preemption rate constraints into these baseline non-stationary methods respectively.

(2) Baseline QL algorithms without considering maximum constraint. These experiments are to examine the effectiveness and necessity of our proposed bisection method on the constraint. We compare our proposed algorithm with the baseline algorithms, classical QL and NSQL. Classical QL is a basic Q-value learning process without a changepoint detection method and maximum constraint. NSQL is a non-stationary QL algorithm that incorporates the changepoint method but does not consider the maximum constraint.

To evaluate our method, we need a performance metric. In the classical case of stationary MDPs, a stationary optimal policy is learned by the RL algorithm. The RL algorithm will be evaluated based on the average cumulative reward (the sum of all of the rewards received in the finite horizon) and the learned policy yield when it is used to control the stationary MDP. The higher the average cumulative reward, the better performance of the RL algorithm.

The DPPCS problem in our research has a big difference from the traditional MDPs. To measure the performance of our proposed approach, we not only need to consider the expected total revenue garnered by the proposed CNSQL algorithm, but also need to consider the preemption rate resulting from the learned optimal policy. In addition, because of the fluctuation of the non-stationary environment, we also consider their standard variance respectively, which represents the robustness of the proposed CNSQL algorithm in the non-stationary environment. In our paper, the performance of the proposed CNSQL algorithm can be considered better if it achieves higher expected total revenue with lower standard variance within a maximum preemption rate constraint.

C. EXPERIMENTS RESULTS

1) DETECTION RESULTS ON MULTI-TEMPORAL PATTERN CHANGE

To answer question (1), we analyze the results of the detected changepoints as shown in Figure 5. This figure presents the outcomes of the changepoint detection for both simulated and real-world non-stationary traces. In Figure 5, the black line represents the time-varying excess capacity, which indicates the available capacity for preemptible services at continuous-time index t. The red line indicates the probability of a specific time point being identified as a changepoint. The X-axis represents the time variable, denoted as t. The left Y-axis represents the excess capacity. The right Y-axis represents the probability of each time point being a changepoint. By examining this visualization, we can gain insights into the identified change points and their corresponding probabilities over the given horizon $T = 100$ and $T = 150$.

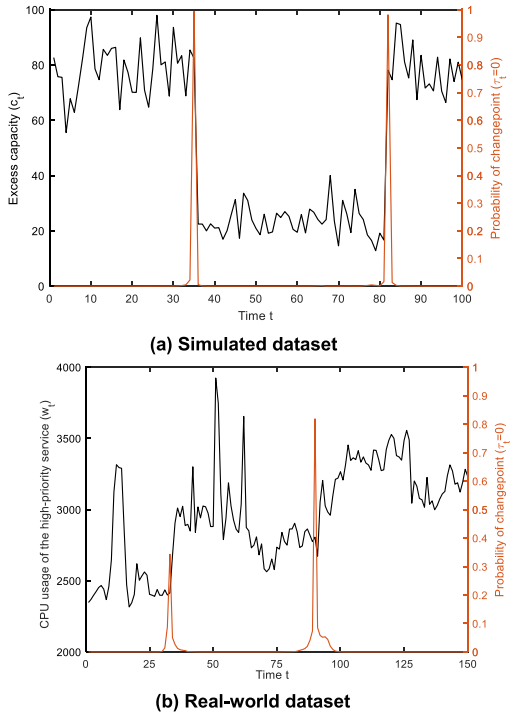

FIGURE 3. The non-stationary traces and changepoints detection results.

In Figure 3-a, the sample size of the delay time (L) is set to 2 time units. Two changepoints are detected: $t_{\theta_1} = 34$ and $t_{\theta_2} = 81$. The probability of these detected changepoints is represented by the red line. By analyzing these probabilities, we can observe that the first changepoint is identified with a probability of $\Pr(\tau_{35} = 0) = 0.99$, while the second change point has a probability of $\Pr(\tau_{82} = 0) = 0.98$. This demonstrates the high predictive capability of the Bayesian changepoint detection method in non-stationary contexts. The change points are accurately detected between two adjacent patterns when the sample size of the delay time is $L = 2$.

In Figure 3-b, we observe the non-stationary traces in a real-world scenario, which exhibit more irregular and unpredictable workload patterns. To handle the irregular and volatile nature of the real-world scenario, a threshold of 0.3 is set to identify changepoints. Specifically, when the probability $\Pr(\tau_t = 0) > 0.3$, time t is considered as a changepoint. The delay time (L) is set to 5-time units in this case. Based on the experimental results, two estimated changepoints and three estimated pattern models are identified. The first pattern model is estimated as $w_t = 2500 + 0.32w_{t-1} + \varepsilon_1$, where $\varepsilon_1 \sim N(0, 3.9^2)$. The first changepoint is located at $t_{\theta_1} = 33$ with probability $\Pr(\tau_{33} = 0) = 0.31$. The second pattern is estimated as $w_t = 2700 + 0.48w_{t-1} + \varepsilon_2$, where $\varepsilon_2 \sim N(0, 7.5^2)$. The second changepoint is located at $t_{\theta_2} = 90$ with a probability $\Pr(\tau_{90} = 0) = 0.82$. The third pattern model is estimated as $w_t = 3100 + 0.21w_{t-1} + \varepsilon_3$, where $\varepsilon_3 \sim N(0, 2.8^2)$.

From the detection results shown in Figure 3, significant pattern shifts rather than instantaneous changes can be detected quickly and accurately with low delay time. The results demonstrate the effectiveness of the detection method for the non-stationary trace in cloud workload.

2) COMPARISON RESULTS WITH BASELINES

To answer question (2), we compare the proposed algorithm with state-of-the-art algorithms. First, we compare with two non-stationary algorithms: QL-eligibility traces [15], and RUQL [37]. To ensure a fair comparison, all three algorithms are subjected to the same constraints. As presented in Figure 4, the experimental results demonstrate the average revenue and its standard variance, as well as the average preemption rate and its standard variance, as the number of learning steps increases. The solid lines in Figure 4 represent the average values of the results over five runs, while the dashed lines represent the corresponding standard deviations. The experiment results are conducted with different maximum constraints, $\varphi = 5\%$ and 20%.

As is shown in Figure 4, it can be observed that the average preemption rate for CNSQL converges to the maximum constraint of 5% with lower error compared to the CQL-eligibility traces and CRUQL algorithms. This is because of the algorithm's ability to perceive changes and update the Lagrangian multiplier λ_{θ_1} and λ_{θ_2} respectively. The optimal Lagrange multipliers vary depending on the different patterns, with $\lambda^*_{\theta_1}$ =244, and $\lambda^*_{\theta_2}$ =219. By adjusting these Lagrange multipliers, CNSQL can make the actual preemption rate satisfy the maximum constraint on each pattern θ_k (k = 1,2), thereby satisfying the maximum constraint over the entire time horizon. Although QL-eligibility traces achieve higher average revenue than CNSQL, they exhibit a higher standard deviation of average revenue. The higher standard deviation arises from the non-stationarity for the excess capacity.

As is shown in Figure 5, the average preemption rate caused by CNSQL converges to around 20% with a standard deviation of 0.04. The optimal Lagrangian multipliers for each temporal pattern are $\lambda^*_{\theta_1}$ =162, and $\lambda^*_{\theta_2}$ =121. Comparing these values to the optimal Lagrangian multipliers under

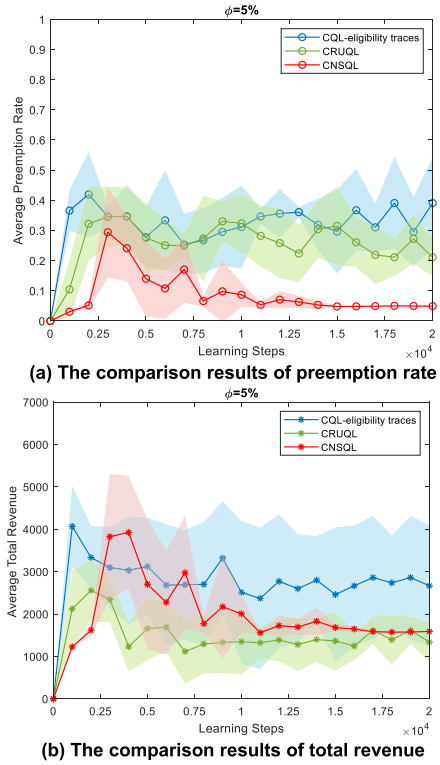

FIGURE 4. Comparison results of CNSQL, CQL-eligibility traces, and CRUQL when φ is 5%.

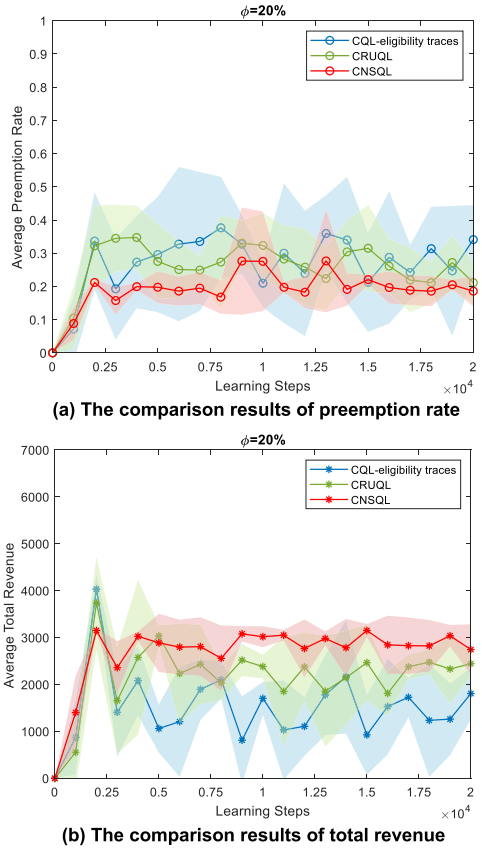

FIGURE 5. Comparison results of CNSQL, CQL-eligibility traces, and CRUQL when φ is 20%.

a maximum constraint of $\varphi = 5\%$, it can be observed that as the maximum constraint φ increases, the optimal Lagrangian multipliers $\lambda^*_{\theta_k}$ decreases. Furthermore, the average revenue obtained with a maximum constraint of $\varphi = 20\%$ is significantly higher than the average revenue obtained with a maximum constraint of $\varphi = 5\%$. Additionally, the average value of the total revenue obtained by CNSQL converges to around 3000, which is the highest compared to QL-eligibility traces and RUQL. This difference in revenue can be attributed to the fact that the CNSQL algorithm can effectively utilize a larger amount of excess capacity to serve preemptible services, particularly when operating under a more relaxed preemption rate constraint.

We compare CNSQL with two baselines, NSQL and QL, which aim to demonstrate the effect of using changepoint detection and maximum constraints on the total revenue and preemption rate. The results in Table 1 show the mean value and standard deviation of the total revenue and actual preemption rate based on five independent runs.

As is shown in Table 1, the baselines (NSQL and QL) without maximum constraints result in an extremely high preemption rate, reaching close to 100%. This high preemption rate makes these algorithms impractical for real-world applications, despite achieving higher average revenue compared to CNSQL. In the QL algorithm, the standard deviation of the total revenue accrued by QL is high. This stems from its inability to detect the changes in patterns and update the pricing policy accordingly. QL only relies on the current observed state taking actions and updates a single Q-value table using experience from all patterns. The NSQL algorithm exhibits a higher revenue compared to QL. NSQL can detect pattern changes and learn the optimal pricing policy for each pattern model. This pattern awareness contributes to the higher average revenue achieved by NSQL. However, NSQL has a high preemption rate due to without considering the maximum constraint.

In contrast, CNSQL leverages changepoint detection and adapts to the pricing policy based on the detected patterns. This approach allows for more accurate modeling of the environment and dynamically adjusting the pricing strategy

TABLE 1. Comparison results of CNSQL, NSQL, and QL when φ is 5% and 20%.

Algorithms	Maximum Constraint φ			
	$\varphi = 5\%$		$\varphi = 20\%$	
	Total Revenue (Mean ± SD)	Preemption Rate (Mean ± SD)	Total Revenue (Mean ± SD)	Preemption Rate (Mean ± SD)
CNSQL	2114.02 (±15.2)	0.04 (±0.012)	3084.49 (±13.8)	0.19 (±0.030)
NSQL	13051.14 (±89.1)	0.69 (±0.511)	24189.35 (±66.1)	0.47 (±0.359)
QL	11839.09 (±151.9)	0.81 (±0.541)	16961.47 (±169.7)	0.84 (±0.530)

accordingly with maximum constraint. As a result, our proposed algorithm achieves a balance between revenue maximization and preemption rate, making it more practical and effective for real-world applications.

Then, we conduct experiments on the real-world trace to verify the practicality of our framework. We adopt the dataset in Figure 5(b) which is much more complicated with irregular change patterns. The comparison results of the total revenue and actual preemption rate with different maximum constraints of $\varphi = 5\%$ and 20% are shown in Table. 2.

TABLE 2. Comparison results of the total revenue and actual preemption rate when φ is 5% and 20%.

Algorithms	Maximum Constraint φ			
	$\varphi = 5\%$		$\varphi = 20\%$	
	Total Revenue (Mean ± SD)	Preemption Rate (Mean ± SD)	Total Revenue (Mean ± SD)	Preemption Rate (Mean ± SD)
CNSQL	20104 (±15.2)	0.045 (±0.001)	50084 (±13.8)	0.195 (±0.04)
CRUQL	15050 (±93.3)	0.051 (±0.22)	37112 (±84.2)	0.213 (±0.20)
CQL with eligibility traces	14056 (±115.1)	0.060 (±0.20)	36506 (±105.1)	0.181 (±0.25)
CQL	17903 (±151.6)	0.314 (±0.56)	20061 (±161.7)	0.911 (±0.61)

Table.2 shows the values of the mean and standard variance for the total revenue in the real-world non-stationary environment. It is evident that the baselines, CQL with eligibility traces, CQL, and CRUQL, fail to achieve higher revenue compared to CNSQL. The main reason is that they fail to make dynamic adjustments to adapt to different pattern changes. Moreover, the preemption rate of baseline algorithms (CQL, CRUQL, and CQL with eligibility trace) exceeds the maximum constraint, indicating that their performance is not well-suited for a non-stationary environment. The updating strategy of the Lagrangian multiplier λ_θ is not effective under non-stationary conditions without change-point detection, and the bisection search struggles to obtain the optimal Lagrange multiplier. CQL has the worst performance because it doesn't consider dynamic pattern change and maximum constraint. The results in Table.2 also show that the average total revenue for a maximum constraint of 20% is higher than the average revenue compared to 5%. We can see that CNSQL exhibits a more remarkable revenue-generating capability when the maximum constraint is relaxed to a higher value.

From the above results presented in Figure 4 and Figure 5, Table 1, and Table 2, we can conclude that setting an appropriate maximum constraint (φ) has a significant impact on the revenue. It is observed that a higher maximum value of φ leads to higher average revenue. However, it is essential to consider that a higher maximum value also corresponds to a higher probability of being preempted, which may require a higher level of fault tolerance for service interruption from consumers. Operators need to carefully investigate the sensitivity of consumers to service interruption. If consumers are more sensitive to service price than service interruption, it may be beneficial to set a higher maximum value of φ to obtain higher revenue. On the other hand, if consumers are more sensitive to service interruption than service price, it would be advisable to set a lower maximum value of φ to attract more consumers and then maintain the potential demand for preemptible services.

3) ABLATION STUDY

To answer question (3), we investigate how the values of the Lagrangian multipliers for different pattern models affect the total revenue and preemption rate achieved by CNSQL algorithm. In this experiment, we set $\lambda_{\theta_1} = 130$ and vary the value of λ_{θ_2} from 0 to 1000. Similarly, we set $\lambda_{\theta_2} = 130$ and vary the value of λ_{θ_1} from 0 to 1000. The expected value of total revenue and preemption rate are presented in Figure 6.

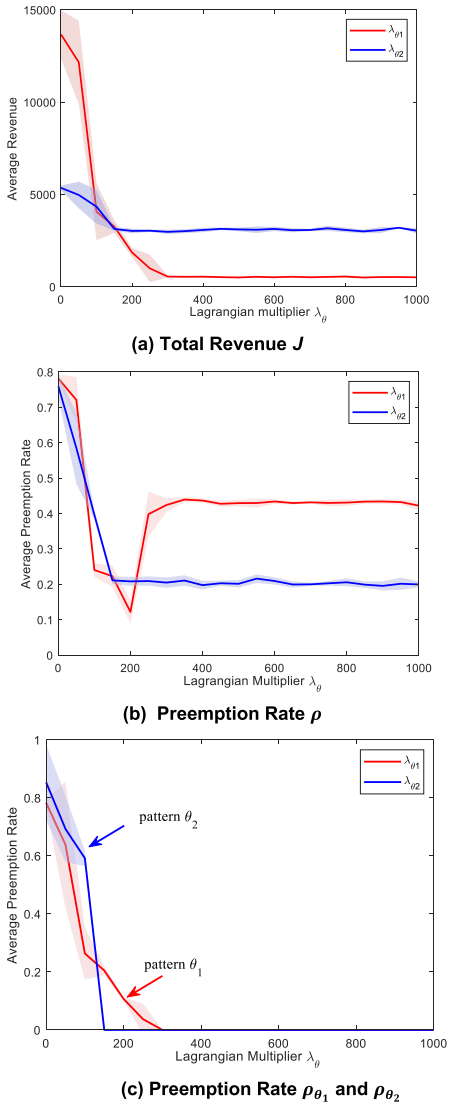

FIGURE 6. Effect of λ_θ on the average revenue and preemption rate accrued by our proposed approach.

In Figure 6-a, we observe the effect of varying Lagrangian multipliers on the expected value of the total revenue in the

entire horizon. When the values of λ_{θ_1} and λ_{θ_2} are at their minimum, the revenue is maximized. It means the operator prioritizes the revenue generated from excess resources over the risk of preemption. As λ_{θ_1} or λ_{θ_2} increase, the operator becomes more concerned about the preemption risk, resulting in a decrease in revenue generated from excess capacity, which eventually converges to a stable value. However, there is a significant difference in the convergence of revenue for different values of λ_{θ_1} and λ_{θ_2}. For example, as λ_{θ_1} increases and reaches $\lambda_{\theta_1} = 239$, the expected value of revenue J converges to around 349.1. But, as λ_{θ_2} increases and reaches $\lambda_{\theta_2} = 173$, J converges to around 3230.8. This difference arises from the disparity in the mean and variance of the excess capacity. The pattern θ_1 with higher mean and lower variance leads to significantly higher revenue compared to pattern θ_2 with lower mean and higher variance. Specifically, as the values of λ_{θ_1} or λ_{θ_2} increase, pattern θ_1 or θ_2 becomes no longer suitable for preemptible services, and as a result, the converged revenue obtained from pattern θ_2 is lower than the revenue obtained from pattern θ_1.

In Figure 6-b, we observe the effect of varying Lagrangian multipliers on the expected value of preemption rate ρ resulting over the entire horizon. When λ_{θ_1} is fixed and λ_{θ_2} increases, the value of ρ decreases and ultimately converges to 0.21. But, when λ_{θ_2} is fixed and λ_{θ_1} increases, the value of ρ initially decreases and then increases eventually converging to 0.42. Indeed, the reasons for the difference in the observed trends are consistent with the difference in the total revenue presented in Figure 6-a. The sensitivity of the Lagrangian multipliers on the preemption rate is much higher in excess capacity patterns with low mean and high variance compared to patterns with high mean and low variance. Specifically, when λ_{θ_1} is fixed, ρ_{θ_1} is fixed which is significantly lower than ρ_{θ_2} caused in pattern θ_2 at the initial value of λ_{θ_2}. As λ_{θ_2} increases, ρ_{θ_2} decreases significantly and quickly converges to zero. Consequently, the value of ρ exhibits a decreasing trend and converges to ρ_{θ_1}. On the contrary, when λ_{θ_2} is fixed, ρ_{θ_2} is relatively higher than $\rho_{\theta_1}(\lambda_{\theta_1} = 130)$. As the value of λ_{θ_1} increases, the value of ρ_{θ_1} decreases. This decrease causes ρ_{θ_1} to be gradually lower than ρ_{θ_2}, resulting in an overall decrease in the preemption rate. However, when λ_{θ_1} increases to 239, the excessively high value of λ_{θ_1} makes pattern θ_1 no longer suitable for adopting preemptible services. As a result, the preemption rate increases and returns to ρ_{θ_2}.

Figure 6-c displays the effect of the Lagrangian multipliers on the expected value of the preemption rate, ρ_{θ_1} and ρ_{θ_2}, which is obtained over pattern θ_1 and θ_2 respectively. The results validate Lemma 1 in Appendix A, which suggests that a higher value of the Lagrangian multiplier λ_{θ_1} or λ_{θ_2} leads to a lower preemption rate ρ_{θ_1} or ρ_{θ_2}. It helps strike a balance the trade-off between maximizing revenue and minimizing the preemption risk, resulting in an appropriate preemption rate for each pattern. This result also demonstrates that the bisection method can assist the agent in finding the optimal Lagrangian multiplier with low computational complexity and fewer iterations.

VI. CONCLUSION

Our research focuses on studying how to dynamically price preemptible services to maximize the revenue generated from excess capacity while satisfying a maximum preemption rate constraint. To solve the DPPCS, we first formulate the DPPCS problem as CNSMDP. Then, we transform the CNSMDP as a piecewise Lagrangian dual problem to overcome the constraint. This transformation allows the constrained preemption rate to be converted into the reward function within the RL framework. To solve the Lagrangian dual problem, we propose a novel dynamic pricing framework that is capable of learning a piecewise optimal pricing policy and the corresponding optimal Lagrangian multiplier. We first propose the estimation methods of the unknown environment parameters, including a detection method for identifying temporal pattern changes, and a diffusion approximation method for estimating the actual preemption rate. It enables us to continuously estimate the unknown change-points among muti-temporal patterns, and the unknown model of changed temporal patterns. The proposed diffusion approximation method can approximate the complex preemption process with a priority-queueing system as a reflected Brownian motion process. Building on the above estimation methods, we then develop a CNSQL algorithm with a bisection method for updating the Lagrangian multiplier. This algorithm can perceive pattern changes in the non-stationary environment and automatically adjusts the learning process of the QL algorithm in response to these changes. If a changed pattern has been encountered in the past, the updated learning process can improve upon the policy learned before.

Experiment results demonstrate the proposed dynamic pricing approach outperforms the classic QL algorithm and other non-stationary RL algorithms. It performs well in maximizing the revenue generated from excess capacity while satisfying the maximum constraint. In addition, we recommend the operators focus on reducing users' sensitivity to service interruption by improving their resilience to such interruptions. If users exhibit high sensitivity to service interruption, operators should consider lowering the maximum constraint on the preemption rate. It is important to strike a balance in setting preemption constraints, as overly strict constraints can negatively impact revenue from preemptible services and may even render them unsuitable for adoption by operators.

Although the proposed approach in this paper effectively ensures revenue maximization within the given constraint, there is still considerable room for designing an efficient algorithm for adapting to the non-stationary cloud environment. For example, instead of real-time pattern recognition, leveraging deep learning (e.g., Long Short-Term Memory, LSTM), which can memorize temporal patterns in time series data, could be advantageous to identify non-stationary temporal patterns in cloud computing. This would enable the approach to adapt to more complex non-stationary structures.

APPENDIX A

In the following, we first prove the existence of a solution for the Lagrangian dual problem in Equation (9) in Section III. Secondly, we derive the expected value of the preemption rate for each pattern θ, $\rho_{n,\theta}$, is monotonic to the Lagrangian multiplier λ_θ, which proves a bisection method is a faster and more efficient method to determine the optimal Lagrangian multiplier λ_θ^*. Before proceeding into the proof of the existence of the Lagrangian dual problem in Equation (9), we need to give basic definitions and assumptions that our proof relies on.

Definition 1: $\pi_\theta^*(\lambda_l) = \text{argmax}_a \rho_\theta(s,a)$: The lowest-preemption policy always chooses higher prices at every state s. $\pi_\theta^*(\lambda_h) = \text{argmax}_a \rho(s,a)$: The highest-preemption policy always chooses lower prices in every state s.

Assumption 1: The maximum constraint under the lowest-preemption policy $\pi_\theta^*(\lambda_l)$ is strictly feasible: $\rho\left[\pi_\theta^*(\lambda_l)\right] < \varphi$.

Assumption 2: The actual preemption rate exceeds the maximum constraint under the highest-preemption policy $\pi_\theta^*(\lambda_h)$: $\rho\left[\pi_\theta^*(\lambda_h)\right] > \varphi$.

The above assumptions are reasonable, and we give the following example to explain the rationality of the assumptions. When the value of λ_h is close to zero, it indicates that the operator has little constraint on the preemption rate. In this case, the optimal policy $\pi_\theta^*(\lambda_h)$ places greater emphasis on maximizing revenue and less emphasis on preemption risk. As a result, the preemption rate exceeds the constraint value φ. On the contrary, when the value of λ_l approaches infinity, the optimal pricing policy $\pi_\theta^*(\lambda_l)$ places greater emphasis on reducing the preemption rate and less emphasis on maximizing revenue. Then, the actual preemption rate $\rho\left[\pi_\theta^*(\lambda_h)\right]$ will tend to zero, then the actual preemption rate $\rho\left[\pi_\theta^*(\lambda_h)\right]$ will be lower than the constraint value φ.

Proposition 1: (Existence of Solutions). When constraint satisfies assumptions 1 and 2, then the optimal value of Lagrangian multiplier λ_θ^* and optimal pricing policy $\pi_\theta(\lambda_\theta^*)$ for each pattern θ always exists that satisfies the constraint $\rho\left[\pi_\theta(\lambda_\theta^*)\right] < \varphi$, where $\lambda_\theta^* > 0$.

Proof. Under assumption 1, a policy that doesn't exceed the constrained preemption rate exists, and therefore the pricing policy $\pi_\theta^*(\lambda_\theta)$ and Lagrangian multiplier λ_θ always have a strictly feasible solution. Under assumption 2, it's obvious that there always exists a positive optimal Lagrangian multiplier λ_θ which penalizes the preemption. If the average preemption rate of high-preemption policy doesn't exceed the constraint φ, then the CNSMDP problem degrades into MDP without constraints, and the optimal Lagrangian multiplier λ_θ is always zero. In conclusion, based on the two natural assumptions which hold in practical problems, we have the above proposition.

Theorem 1: $\rho\left[\pi_\theta^*(\lambda_\theta)\right]$ is monotonically non-increased with the increase of λ_θ. *e.g.*, if $\lambda_l < \lambda_h$, then $\rho\left[\pi_\theta^*(\lambda_l)\right] \geq \rho\left[\pi_\theta^*(\lambda_h)\right]$, where $\pi_\theta^*(\lambda_l) = \text{argmax}_{\boldsymbol{\pi}} L(\pi_\theta, \lambda_l)$, $\pi_\theta^*(\lambda_h) = \text{argmax}_{\boldsymbol{\pi}} L(\pi_\theta, \lambda_h)$.

Proof. According to the definition of $\pi_\theta^*(\lambda_\theta)$, we have:

$$V\left[\pi_\theta^*(\lambda_l)\right] - \lambda_l\left\{\rho\left[\pi_\theta^*(\lambda_l)\right] - \varphi\right\} \geq V\left[\pi_\theta^*(\lambda_h)\right] - \lambda_l\left\{\rho\left[\pi_\theta^*(\lambda_h)\right] - \varphi\right\} \tag{25}$$

$$V\left[\pi_\theta^*(\lambda_h)\right] - \lambda_h\left\{\rho\left[\pi_\theta^*(\lambda_h)\right] - \varphi\right\} \geq V\left[\pi_\theta^*(\lambda_l)\right] - \lambda_h\left\{\rho\left[\pi_\theta^*(\lambda_l)\right] - \varphi\right\} \tag{26}$$

By adding the two sides of the two inequalities, Equation (25) and Equation (26), then

$$\begin{aligned} &V\left[\pi_\theta^*(\lambda_l)\right] - \lambda_l\left\{\rho\left[\pi_\theta^*(\lambda_l)\right] - \varphi\right\} + V\left[\pi_\theta^*(\lambda_h)\right] \\ &- \lambda_l\left\{\rho\left[\pi_\theta^*(\lambda_h)\right] - \varphi\right\} \\ &\geq V\left[\pi_\theta^*(\lambda_h)\right] - \lambda_h\left\{\rho\left[\pi_\theta^*(\lambda_h)\right] - \varphi\right\} + V\left[\pi_\theta^*(\lambda_l)\right] \\ &- \lambda_h\left\{\rho\left[\pi_\theta^*(\lambda_l)\right] - \varphi\right\} \end{aligned} \tag{27}$$

Then, after eliminating the same terms on both sides of Equation (27), we have:

$$(\lambda_l - \lambda_h)\left\{\rho\left[\pi_\theta^*(\lambda_h)\right] - \rho\left[\pi_\theta^*(\lambda_l)\right]\right\} \geq 0 \tag{28}$$

because $\lambda_a < \lambda_b$, which leads to the following conclusion:

$$\rho\left[\pi_\theta^*(\lambda_l)\right] \geq \rho\left[\pi_\theta^*(\lambda_h)\right] \tag{29}$$

Proof finishes.

APPENDIX B

Q-learning is a classical algorithm for solving RL problems by learning the optimal policy modeled as stationary MDP. The basis of this method is to calculate the quality in state s using action a by trial-and-error by defining a state-action-value function $Q(s,a)$:

$$Q(s,a) = \mathbf{E}_{\pi\sim(s,a)}\left[\sum_{i=0}^{N-n} \gamma^i R_{n+i}(\pi) \,|(a_n = a|s_n = s)\right] \tag{30}$$

where π is short of $\pi(a|s)$, which is the probability distribution for choosing action a in state s. $R_{n+i}(\pi)$ is one-step revenue at the $n+i$ th period. γ is the discount factor. According to Equation (30), the higher the value of $Q(s,a)$ is, the more beneficial it is for long-term revenue to adopt a pricing policy π. Then, the optimal pricing policy π^* learned means that in state s, according to policy π^*, adopting price a can obtain the highest long-term expected revenue and satisfies $Q(\pi^*) = Q(\pi)$. For stationary MDP, the state-value function $V(s)$ can be precisely predicted by the repeatedly learning of agent through Equation (30), and the relationship between the two is: $V^*(s) = Q^*(\pi(a|s))$. It has been theoretically proved that when the agent interacts with a stationary environment according to Equation (31), the optimal policy $\pi^*(a|s)$ for the finite horizon, T, can be obtained.

$$Q(s,a) \leftarrow Q(s,a) + \delta\left[R(s,a) + \gamma Q\left(s',a'\right) - Q(s,a)\right] \tag{31}$$

where δ is the learning rate.

REFERENCES

[1] S. Chen, K. Moinzadeh, and Y. Tan, "Discount schemes for the preemptible service of a cloud platform with unutilized capacity," *Inf. Syst. Res.*, vol. 32, no. 3, pp. 967–986, Sep. 2021, doi: 10.1287/isre.2021.1011.

[2] S. Yuan, S. Das, R. Ramesh, and C. Qiao, "Service agreement trifecta: Backup resources, price and penalty in the availability-aware cloud," *Inf. Syst. Res.*, vol. 29, no. 4, pp. 947–964, Dec. 2018, doi: 10.1287/isre.2017.0755.

[3] J. Maghakian, J. Comden, and Z. Liu, "Online optimization in the non-stationary cloud: Change point detection for resource provisioning (invited paper)," in *Proc. 53rd Annu. Conf. Inf. Sci. Syst. (CISS)*, Baltimore, MD, USA, Mar. 2019, pp. 1–6, doi: 10.1109/CISS.2019.8692890.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010, doi: 10.1145/1721654.1721672.

[5] J. Barr. (Jun. 5, 2015). *Cloud Computing, Server Utilization, & the Environment| AWS News Blog*. Accessed: Nov. 11, 2022. [Online]. Available: https://aws.amazon.com/blogs/aws/cloud-computing-server-utilization-the-environment/

[6] A. N. Avramidis and A. V. D. Boer, "Dynamic pricing with finite price sets: A non-parametric approach," *Math. Methods Oper. Res.*, vol. 94, no. 1, pp. 1–34, Aug. 2021, doi: 10.1007/s00186-021-00744-y.

[7] L. Lin, L. Pan, and S. Liu, "Methods for improving the availability of spot instances: A survey," *Comput. Ind.*, vol. 141, pp. 1–18, Oct. 2022, doi: 10.1016/j.compind.2022.103718.

[8] G. George, R. Wolski, C. Krintz, and J. Brevik, "Analyzing AWS spot instance pricing," in *Proc. IEEE Int. Conf. Cloud Eng. (IC2E)*, Prague, Czech Republic, Jun. 2019, pp. 222–228, doi: 10.1109/IC2E.2019.00036.

[9] O. A. Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, "Deconstructing Amazon EC2 spot instance pricing," *ACM Trans. Econ. Comput.*, vol. 1, no. 3, pp. 1–20, Sep. 2013, doi: 10.1145/2509413.2509416.

[10] H. Xu and B. Li, "Dynamic cloud pricing for revenue maximization," *IEEE Trans. Cloud Comput.*, vol. 1, no. 2, pp. 158–171, Jul. 2013, doi: 10.1109/TCC.2013.15.

[11] F. Alzhouri, A. Agarwal, and Y. Liu, "Maximizing cloud revenue using dynamic pricing of multiple class virtual machines," *IEEE Trans. Cloud Comput.*, vol. 9, no. 2, pp. 682–695, Apr. 2021, doi: 10.1109/TCC.2018.2878023.

[12] D. M. Davidow, "Analyzing Alibaba cloud's preemptible instance pricing," M.S. dissertation, Dept. Comput. Sci., Univ. Haifa, Haifa, Israel, 2021.

[13] E. Lecarpentier and E. Rachelson, "Non-stationary Markov decision processes, a worst-case approach using model-based reinforcement learning, extended version," Jan. 2020, *arXiv:1904.10090*. Accessed: May 18, 2022.

[14] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2017.

[15] R. Rana and F. S. Oliveira, "Real-time dynamic pricing in a non-stationary environment using model-free reinforcement learning," *Omega*, vol. 47, pp. 116–126, Sep. 2014, doi: 10.1016/j.omega.2013.10.004.

[16] R. Rana and F. S. Oliveira, "Dynamic pricing policies for interdependent perishable products or services using reinforcement learning," *Expert Syst. Appl.*, vol. 42, no. 1, pp. 426–436, Jan. 2015, doi: 10.1016/j.eswa.2014.07.007.

[17] R. Lu, S. H. Hong, and X. Zhang, "A dynamic pricing demand response algorithm for smart grid: Reinforcement learning approach," *Appl. Energy*, vol. 220, pp. 220–230, Jun. 2018, doi: 10.1016/j.apenergy.2018.03.072.

[18] L. Zhang, Y. Gao, H. Zhu, and L. Tao, "A distributed real-time pricing strategy based on reinforcement learning approach for smart grid," *Expert Syst. Appl.*, vol. 191, pp. 1–12, Apr. 2022, doi: 10.1016/j.eswa.2021.116285.

[19] D. Kumar, G. Baranwal, Z. Raza, and D. P. Vidyarthi, "A survey on spot pricing in cloud computing," *J. Netw. Syst. Manage.*, vol. 26, no. 4, pp. 809–856, Oct. 2018, doi: 10.1007/s10922-017-9444-x.

[20] A. Deldari and A. Salehan, "A survey on preemptible IaaS cloud instances: Challenges, issues, opportunities, and advantages," *Iran J. Comput. Sci.*, vol. 4, no. 3, pp. 1–24, Sep. 2021, doi: 10.1007/s42044-020-00071-1.

[21] H. Peng, Y. Cheng, and X. Li, "Real-time pricing method for spot cloud services with non-stationary excess capacity," *Sustainability*, vol. 15, no. 4, p. 3363, Feb. 2023, doi: 10.3390/su15043363.

[22] J. M. Harrison, *Brownian Models of Performance and Control*. Cambridge, U.K.: Cambridge Univ. Press, 2013.

[23] D. Zhang and L. Weatherford, "Dynamic pricing for network revenue management: A new approach and application in the hotel industry," *INFORMS J. Comput.*, vol. 29, no. 1, pp. 18–35, Jan. 2017, doi: 10.1287/ijoc.2016.0713.

[24] A. V. D. Boer, "Dynamic pricing and learning: Historical origins, current research, and new directions," *Surv. Oper. Res. Manage. Sci.*, vol. 20, no. 1, pp. 1–18, Jun. 2015, doi: 10.1016/j.sorms.2015.03.001.

[25] A. V. D. Boer, "Tracking the market: Dynamic pricing and learning in a changing environment," *Eur. J. Oper. Res.*, vol. 247, no. 3, pp. 914–927, Dec. 2015, doi: 10.1016/j.ejor.2015.06.059.

[26] F. Alzhouri, A. Agarwal, Y. Liu, and A. S. Bataineh, "Dynamic pricing for maximizing cloud revenue: A column generation approach," in *Proc. 18th Int. Conf. Distrib. Comput. Netw.*, Hyderabad, India, Jan. 2017, pp. 1–9, doi: 10.1145/3007748.3007756.

[27] C. V. L. Raju, Y. Narahari, and K. Ravikumar, "Learning dynamic prices in electronic retail markets with customer segmentation," *Ann. Oper. Res.*, vol. 143, no. 1, pp. 59–75, Mar. 2006, doi: 10.1007/s10479-006-7372-3.

[28] Y. Cheng, "Dynamic packaging in e-retailing with stochastic demand over finite horizons: A Q-learning approach," *Expert Syst. Appl.*, vol. 36, no. 1, pp. 472–480, Jan. 2009, doi: 10.1016/j.eswa.2007.09.050.

[29] B.-G. Kim, Y. Zhang, M. van der Schaar, and J.-W. Lee, "Dynamic pricing and energy consumption scheduling with reinforcement learning," *IEEE Trans. Smart Grid*, vol. 7, no. 5, pp. 2187–2198, Sep. 2016, doi: 10.1109/TSG.2015.2495145.

[30] K. Siddesha, G. V. Jayaramaiah, and C. Singh, "A novel deep reinforcement learning scheme for task scheduling in cloud computing," *Cluster Comput.*, vol. 25, no. 6, pp. 4171–4188, Dec. 2022, doi: 10.1007/s10586-022-03630-2.

[31] A. Alsarhan, A. Itradat, A. Y. Al-Dubai, A. Y. Zomaya, and G. Min, "Adaptive resource allocation and provisioning in multi-service cloud environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 1, pp. 31–42, Jan. 2018, doi: 10.1109/TPDS.2017.2748578.

[32] 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 Trans. Parallel Distrib. Syst.*, vol. 29, no. 12, pp. 2742–2756, Dec. 2018, doi: 10.1109/TPDS.2018.2843343.

[33] R. Salles, K. Belloze, F. Porto, P. H. Gonzalez, and E. Ogasawara, "Nonstationary time series transformation methods: An experimental review," *Knowl.-Based Syst.*, vol. 164, pp. 274–291, Jan. 2019, doi: 10.1016/j.knosys.2018.10.041.

[34] Y. Zhang, B. Tang, and Q. Yang, "BCORLE(λ): An offline reinforcement learning and evaluation framework for coupons allocation in e-commerce market," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 20410–20422.

[35] R. P. Adams and D. J. C. MacKay, "Bayesian online changepoint detection," Oct. 2007, *arXiv:0710.3742*. Accessed: Jun. 30, 2022.

[36] X. Liu, "Diffusion approximations for double-ended queues with reneging in heavy traffic," *Queueing Syst.*, vol. 91, nos. 1–2, pp. 49–87, Feb. 2019, doi: 10.1007/s11134-018-9589-7.

[37] S. Abdallah and M. Kaisers, "Addressing environment non-stationarity by repeating Q-learning updates," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1582–1612, 2016.

HUIJIE PENG was born in Shandong, China, in 1990. She received the B.S. degree in financial management from Zao Zhuang University, Shandong, in 2013, and the M.S. degree in corporate management from the Shanghai University of Engineering and Technology, Shanghai, in 2017. She is currently pursuing the Ph.D. degree in management science and engineering. Her research interests include cloud computing, dynamic pricing, and reinforcement learning.

YAN CHENG received the B.S. degree in management science from the Nanjing University of Science and Technology, Nanjing, China, in 1990, and the M.S. degree in engineering and Ph.D. degree in management science from the Harbin Institute of Technology, Harbin, China, in 1997 and 2001, respectively. He is a Professor with the Department of Management Science and Engineering, East China University of Science and Technology, Shanghai. He has authored or coauthored about 40 scientific articles. His research interests include information management and information systems, data science, and revenue management.