Online Query-based Data Pricing with Time-discounting Valuations

Yicheng Fu*, Xiaoye Miao*⋆, Huanhuan Peng*, Chongning Na‡, Shuiguang Deng†, Jianwei Yin†
*Center for Data Science, Zhejiang University, Hangzhou, China
⋆The State Key Lab of Brain-Machine Intelligence, Zhejiang University, Hangzhou, China
‡FinTech Research Center, Zhejiang Lab, Hangzhou, China
†College of Computer Science, Zhejiang University, Hangzhou, China
Email: {fuycc, miaoxy, hhpeng}@zju.edu.cn; na@zhejianglab.com; dengsg@zju.edu.cn; zjuyjw@cs.zju.edu.cn

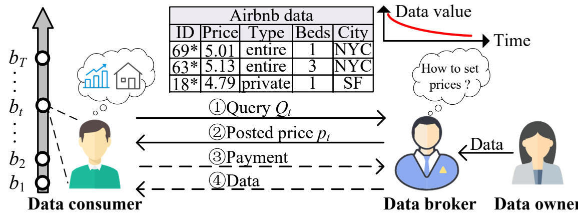
***Abstract*—Online data marketplaces emerge in diverse data-driven applications, where dynamically arriving consumers purchase the data at posted prices. The data value decays over time in many tasks, such as machine learning predictions and real-time systems. Existing query pricing methods do not consider the time-discounting data value. In this paper, we study *the query feature-based data pricing problem* with *unknown time-discounting data valuation*. We propose an effective online data pricing mechanism Pride to maximize the cumulative sales revenue. It leverages the powerful property of the *ellipsoid method* to efficiently solve online optimization via *exploration and exploitation*. Based on Thompson sampling, we present a *novel non-stationary* MAB algorithm Biased-TS to determine a suitable discount factor and attain the dynamic posted price. It is theoretically proved that, the regret upper bound order of Pride is dominated by the discretization error $O(\frac{T}{K})$, where K and T are the numbers of discount candidates and total trading rounds, respectively. Biased-TS gets a sub-linear regret upper bound $O(K^3\sqrt{T \ln T} + K \exp\{4\sqrt{\ln T}\})$. Extensive experiments using both synthetic and real datasets demonstrate that Pride yields around 90% of the optimal cumulative revenue, and it substantially outperforms the state-of-the-art methods.**

***Index Terms*—data pricing, online pricing, ellipsoid, Thompson sampling**

I. Introduction

Data have significant economic and social value in many application fields, such as financial markets, the Internet of Things (IoT) industries, real-time data analytics, and social networks. It innovates the emergence of online data markets [1] and many data trading platforms, e.g., GDBEX [2], SZDEx [3], Xignite [4], and Datacoup [5]. The query-based data trade [6]–[9] allows the data buyers to obtain desirable and fine-grained data via establishing query requests. This method of data trade in the online market can greatly improve data circulation and alleviate the problem of data isolation [5].

Fig. 1 depicts an online query-based data pricing scenario in *Airbnb* accommodation data [10]. As an example, a data buyer b_1 plans to train a machine learning (ML) model to forecast the rental prices. He wants to purchase the *Airbnb* accommodation data. Hence, the buyer b_1 arrives at this market and requests a query Q_1 to select the data that benefits the forecasting. The data broker, who collects the data from the data owner, posts a price p_1 based on the query Q_1. The buyer b_1 then will *either* pay for the query at the posted price p_1 *or* leave



Airbnb data				
ID	Price	Type	Beds	City
69*	5.01	entire	1	NYC
63*	5.13	entire	3	NYC
18*	4.79	private	1	SF

Fig. 1. Online data trading model

the market without payment, based on his/her *private* query valuation. Similarly, each of the other buyers later arrives and posts a different query relying on the demand. The data broker is faced with a critical problem: *how to price the fine-grained data in such an online market to maximize the cumulative revenue* (i.e., the sum of the buyers' payments).

Moreover, the data value usually *decays* over time in many data-driven applications, like ML predictions [11], [12], real-time systems [13], etc. Data that are *more up-to-date* can make *more contributions* to data-driven applications. For example, in the rental market, fresh data can capture current preferences and improve the accuracy of predicting rental prices. However, the market value of *Airbnb* accommodation data decreases over time as the current data becomes *stale* for the ever-changing rental market. It is important to consider this *time-discounting* data value when setting prices, as subsequent data buyers may not accept the price and result in much loss (a.k.a. regret) for the data broker. Nevertheless, the fact is that the *private* valuation of data buyers is always *unknown* to the data broker in real data markets, let alone the time-discounting one.

Therefore, we study *the revenue maximization problem of posted-pricing queries online under the unknown time-discounting data value setting* for the first time. There are two major challenges to solving the studied problem. The first one is how to effectively estimate the feature-based query valuation based on very *limited* feedback. In general, the query valuation of the buyer is always associated with a set of features, e.g., data privacy, data size, etc. The direct way for revenue maximization is to collect the query features and the corresponding valuations from buyers, learn the valuation function, and price any query based on the learned valuation. However, the real-world buyer is unwilling to expose his valuation, while the data broker can only know whether the buyer pays or not, i.e., whether the price is higher than the valuation.

Thus, it is hard to estimate the valuation from the limited binary feedback of buyers, especially for complex valuations. The second challenge is how to model the *unknown* time-discounting characteristic of the data value in the pricing issue for further maximizing the revenue. When data value is time-discounting, the historical knowledge of the data market value cannot provide an accurate estimation of future data value, and the previously posted price may be unacceptable, thus causing great regret. One possible solution is to use discrete values to approximate the consecutive discount trend of the data value as closely as possible. However, it is difficult to dynamically select a suitable discount factor per round. One should exploit historical knowledge and explore the future discount trend at the same time. To the best of our knowledge, none of the existing studies on data pricing and online pricing [14]–[16] can address the two challenges together.

In this paper, we propose an *effective* online query-based data pricing mechanism, named Pride, that dynamically prices customized queries from data buyers with time-discounting valuations to pursue the goal of cumulative revenue maximization. It models each query with a feature vector that can well distinguish different query demands from data buyers. To estimate the data market value, Pride utilizes a *knowledge set update* (KSU) module to progressively tighten the knowledge set with binary feedback. The *feature-based price calculation* (FPC) module of Pride leverages an efficient and effective ellipsoid method to solve the n-dimensional online optimization problem and obtain a *feature-based price* dynamically.

We further present a new non-stationary multi-armed bandit (MAB) algorithm Biased-TS (Biased Thompson sampling) as the *discount factor determination* (DFD) module of Pride. Biased-TS employs an effective *biased update* technique to dynamically select the suitable *discount factor*. It greatly facilitates tracking the discount trend of data value. Pride eventually multiplies the *feature-based price* and *discount factor* as the posted query price. We theoretically derive a sub-linear regret upper bound of Biased-TS. The cumulative regret upper bound of Pride is guaranteed. It is determined by feature-based pricing calculation, discount factor determination by the Biased-TS algorithm, and discretization error. The contributions of this paper are summarized as follows.

- We propose an effective online query pricing mechanism Pride that utilizes the ellipsoid method and non-stationary MAB to maximize the cumulative revenue. To our best knowledge, it is the first to study the *feature-based* query pricing with the time-discounting data value.
- We prove a discretization error-dominated upper bound $O(\frac{T}{K})$ of cumulative regret, where K and T are the numbers of discount candidates and total trading rounds, respectively. We present a *novel* non-stationary MAB algorithm Biased-TS to capture the time-discounting trend of data value. We derive its sub-linear regret upper bound of $O(K^3\sqrt{T\ln T} + K\exp\{4\sqrt{\ln T}\})$.
- Extensive experiments using synthetic and real datasets demonstrate that Pride yields about 90% optimal cumulative revenue, outperforming the state-of-the-art [14].

TABLE I
SYMBOLS AND DESCRIPTION

Notation	Description
$\mathbb{T}$	the set of T trade rounds, i.e., $\mathbb{T} = \{1, 2, \cdots, T\}$
b_t	the data buyer arrived at round t
$(Q_t, \mathbf{x}_t)$	the query Q_t made by buyer b_t with feature $\mathbf{x}_t \in \mathbb{R}^n$
v_t	the valuation on the query Q_t
p_t	the posted price for the query Q_t
$\boldsymbol{\theta}^*$	the original market value vector of the entire dataset $\mathscr{D}$
$d(t)$	the time-discounting function for the data market value
$R(T)$	the cumulative regret of trading the total T queries
$\mathcal{K}_t$	the knowledge set of data market value in round t
ε_1	the threshold for exploration and exploitation
p_t^o	the feature-based price in round t
K	the number of discount factor candidates
$\mathbb{F}$	the set of discount factors, i.e. $\mathbb{F} = \{\mu_0, \mu_1, \cdots, \mu_K\}$
d_t	the selected discount factor in round t
E_t	the ellipsoid of the knowledge set $\mathcal{K}_t$
f_t	the feedback from the data buyer b_t, i.e. $f_t \in \{0, 1\}$

The rest of this paper is organized as follows. Section II describes the related work. Section III formalizes the problem. Section IV overviews our pricing scheme Pride. Section V elaborates on the solution details. Section VI gives theoretical analysis. Experimental results and our findings are reported in Section VII. Finally, we conclude the paper in Section VIII.

II. Related work

Existing *query-based data pricing* studies explore various data products, e.g., typical SQL queries on relational database [6]–[8], [17]–[19], linear aggregate query on private data [14], [20], [21], query on private time-series [22] or correlated data [23], query on graph data [24], [25], and aggregate statistics on crowdsensing data [15]. These works all ignore the time-discounting characteristic of the data value.

Existing *online data pricing* studies include pricing queries with feature vectors [14], [21], pricing crowdsensing data distributions [15], and valuing the data during collection [26]. They neglect the time-discounting data value. In contrast, the pricing methods with the discounting data value either simply use known discount functions [16], study the incentive mechanisms in data collections [27]–[29] and data analytics service pricing [30], or adopt auction to support the decreasing buyer's utility [31]. Due to different problems, settings, or price calculations, these time-aware valuation based solutions cannot tackle feature-based query pricing.

In addition, there are some *online pricing* mechanisms considering time-discounting valuations but *not for data products*. A group of them [32], [33] calculates an explicit price for every buyer based on the given information. The other online pricing mechanisms [34], [35] utilize a non-stationary MAB strategy to make an exploration and exploitation trade-off to select the most proper price. Our time-discounting factor learning strategy belongs to the latter one. In particular, the discount function is given in [33], and it is bounded by an estimate function in [34]. By contrast, the discount function is completely unknown in [32], [35], which is the same as our setting. All these works cannot support the pricing of different queries and thus are infeasible to solve our problem.

III. Problem Formulation

In this section, we describe the online data marketplace and our query valuation model. We formalize our studied problem. Table I lists the frequently used symbols in the paper.

Online data marketplace. In this paper, we employ a general online data market model. It is composed of three entities: the data owner, the data broker, and the data consumer (or data buyer). The data buyer arrives at the market in an online fashion. Assume the total online time is divided into T rounds and collected in a set $\mathbb{T} = \{1, 2, \cdots, T\}$. In round $t \in \mathbb{T}$, a data buyer b_t arrives and makes a query Q_t with a private valuation v_t. There is no independence assumption among queries of different rounds. Based on the query Q_t, the data broker offers the data buyer a non-negative price p_t.

The data buyer. The buyer accepts the price p_t if the price is no larger than the buyer's valuation, i.e., $p_t \leq v_t$, and the buyer's utility is $(v_t - p_t)$. Otherwise, the buyer rejects it and gains no utility. It is assumed that, the data buyer is *rational* and *selfish* [16], [33], who plays its *dominant strategy* [36]. A dominant strategy is one that maximizes the buyer's individual utility regardless of what other buyers do [33], [37].

The posted price mechanism. We utilize the posted price mechanism to trade queries. Referring to game theory, the posted price mechanism is *dominant-strategy incentive-compatible* (DSIC). It means that, the buyer cannot acquire a utility better than acting *truthfully* and never leads to negative utility [37]–[40]. In this paper, the data broker first posts a price, and then the data buyer decides whether to pay for the query. Since the posted price is irrelevant to the valuation of the current buyer, acting any other strategy cannot increase the utility of the buyer. Hence, the posted price mechanism is DSIC and thus *strategy-proof*. It can prevent the presence of *rational* strategic buyers. The buyer can achieve *individual utility maximization* under the posted price mechanism. Note that, some *irrational* strategic buyers may try to bias the data broker by rejecting the acceptable prices, despite harming their individual utilities. That is worthy of future research.

The goal of the data broker is to maximize its cumulative revenue in the total T rounds. In round t, if the posted price is accepted ($p_t \leq v_t$), the revenue is equal to the value of p_t, otherwise 0. Ideally, setting the posted price as the buyer's valuation in each round can get the maximum overall revenue. However, the data broker does not know the actual query valuation of the buyer in the real situation.

Feature-based query representation. We use the famous *hedonic pricing* [41]–[43] in marketing to model the valuation of data buyers' queries. It states that the valuation of a product is described by a deterministic function of its *features*. In this paper, the product is the query, and the deterministic function describes the query valuation from the buyer. The vector $\mathbf{x}_t \in \mathbb{R}^n$ is used to denote the *n*-dimensional feature of a query Q_t.

It is worthwhile to mention that, this feature-based query representation can be widely applied to different real-life scenarios. For example, the feature vector can represent the *usage volume* of a SQL query over a database $\mathscr{D}$ in the market. Namely, each element of the feature vector $\mathbf{x}_t$ can correspond to an attribute of the using database $\mathscr{D}$ in the market, and the value of this element indicates the proportion of the queried data amount to the total data. Besides, the feature vector can also represent the *buyer's selection* of data tuples, where each element of the vector indicates whether the corresponding tuple is chosen. Moreover, when it comes to the personal data market, the feature can be regarded as the *privacy compensation* for the data owners, e.g., measuring the leakage degree of owers' privacy [14], [21].

The time-discounting market value of the data. In this paper, for a dataset $\mathscr{D}$, the market value is described by a *n*-dimensional vector. In particular, the original market value vector of $\mathscr{D}$ is denoted by $\boldsymbol{\theta}^* \in \mathbb{R}^n$, which is unknown for the data broker. It reflects the unique market value of the dataset. Each element of $\boldsymbol{\theta}^*$ measures the value of each dimension of the data. Typically the data value changes over time. Many data-driven applications require *fresh* data [13], [29], such as cloud computing, real-time monitoring, etc. Empirical studies on face verification [30] and NLP [11] also emphasize that the data quality *decays* over time, harming the model performance. The decay period of data value may span from seconds to decades [30], [44]. In other cases, the data value may *increase* over time such as the time-series data [45], or *fluctuate* for market uncertainty [14]. In this paper, we focus mainly on the time-discounting data value. Our solution can also be generalized to deal with more complex changes in data value.

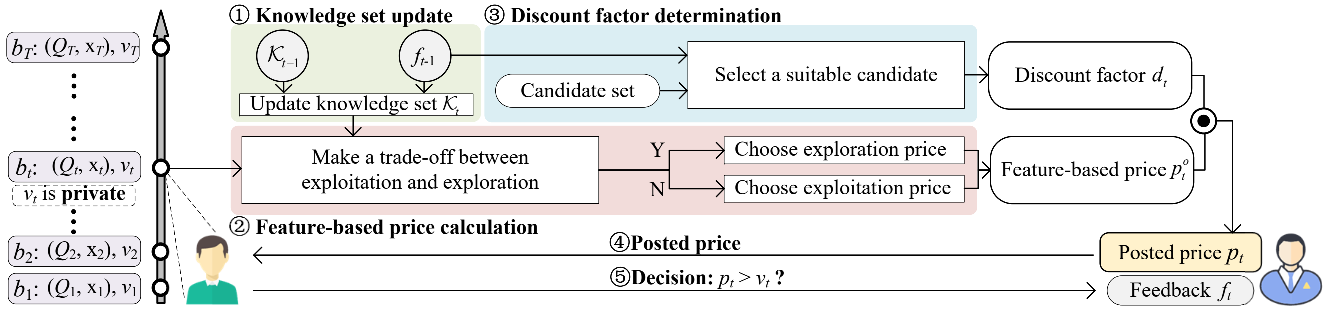
To represent the time-discounting market value of the dataset $\mathscr{D}$, we add a *time-discounting function* $d(t)$ to the original market value in the form of $d(t) \cdot \boldsymbol{\theta}^*$. Namely, the data market value decays following a time-discounting function $d(t)$ over time *t*. Hence, we can get the valuation model of the query Q_t via combining the query feature $\mathbf{x}_t$ and the time-discounting data market value $d(t) \cdot \boldsymbol{\theta}^*$ of the dataset $\mathscr{D}$.

Definition 1: (**The query valuation model**). Given a dataset $\mathscr{D}$, let $\boldsymbol{\theta}^* \in \mathbb{R}^n$ denote the original data market value, and $d(t)$ be the time-discounting function. For a query Q_t with its *n*-dimensional feature vector $\mathbf{x}_t \in \mathbb{R}^n$, the valuation of the query Q_t, denoted by $v(Q_t, \mathscr{D})$, abbreviated as v_t, is written as

$$v_t = d(t) \cdot \mathbf{x}_t^T \cdot \boldsymbol{\theta}^* \tag{1}$$

This linear model is widely used in *contextual pricing* [41], [46]–[48]. Adding a time-discounting characteristic to the market value through multiplication form is a common practice in research [32], [34], [35]. For clarity, we prefer to call the value of the dataset $d(t) \cdot \boldsymbol{\theta}^*$ as the data *market value*, and the buyer's estimated value of the query v_t as the query *valuation*.

Obviously, the query valuation in Definition 1 contains three components: i) the query feature $\mathbf{x}_t$ that well distinguish customized queries, ii) the original market value of the dataset that reflects the intrinsic value when data is fresh, and iii) the time-discounting function that reflects the non-increasing trend of the data value. Moreover, this query valuation model can be generalized, where our solution is still feasible. For example, we can consider the *market noise* to represent the fluctuation of the market value. Let a random variable δ_t



① Knowledge set update
③ Discount factor determination
b_T: (Q_T, x_T), v_T
$\mathcal{K}_{t-1}$
f_{t-1}
Update knowledge set $\mathcal{K}_t$
Candidate set
Select a suitable candidate
Discount factor d_t
b_t: (Q_t, x_t), v_t
v_t is private
Make a trade-off between exploitation and exploration
Y
N
Choose exploration price
Choose exploitation price
Feature-based price p_t^o
b_2: (Q_2, x_2), v_2
b_1: (Q_1, x_1), v_1
② Feature-based price calculation
④Posted price
⑤Decision: $p_t > v_t$?
Posted price p_t
Feedback f_t

Fig. 2. The framework of online query pricing mechanism Pride

denote the noise in round t, which is independent of the feature $\mathbf{x}_t$. The *noisy* valuation model of the buyer b_t can be written as $d(t) \cdot (\mathbf{x}_t^T \boldsymbol{\theta}^* + \delta_t)$. The noise may increase the regret of the data broker, as we empirically validated in Section VII.

As discussed earlier, the buyer's query valuation can be regarded as an optimal value of the posted price, when pursuing the cumulative revenue maximization. The gap between the actual posted price and the optimal one (i.e., query valuation) is what we called *regret*. Thus, the original maximization problem of the cumulative revenue can be equivalently converted to the minimization problem of the *cumulative regret*.

Definition 2: (**The cumulative regret**). Suppose there are a total of T rounds of trade made in an online query transaction. Let r_t denote the regret in a single round t, p_t is the posted price, and v_t is the buyer's query valuation. The cumulative regret, denoted by $R(T)$, can be represented as

$$R(T) = \sum_{t=1}^{T} r_t = \sum_{t=1}^{T} (v_t - p_t \cdot \mathbb{1}\{p_t \leq v_t\}) \tag{2}$$

In particular, if the posted price is no larger than the valuation, it makes a regret equal to the difference between the two. Otherwise, it incurs even more regret for no payment.

Definition 3: (**Problem statement**). Suppose there is a sequence of different query requests $Q_1, Q_2, \cdots, Q_T$ (with feature vectors $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T$) posted on the online market to retrieve a dataset $\mathscr{D}$ that has an unknown time-discounting market value. The problem studied in this paper is to dynamically calculate a proper posted price p_t for the query Q_t, $t \in \mathbb{T}(= \{1, 2, \cdots, T\})$, such that the cumulative regret defined in Eq. 2 is minimized.

Intuitively, to solve this problem, we should attain an accurate query valuation as the posted price for query trading. Based on Eq. 1, the query valuation is composed of three parts: the discount function $d(t)$, the original data market value $\boldsymbol{\theta}^*$, and the query representation $\mathbf{x}$. In particular, give a query Q, we can directly obtain the query feature $\mathbf{x}$. However, how to dynamically derive the proper data market value and its discount level at a certain round is rather challenging.

IV. Solution Overview

In this section, we overview our proposed online pricing mechanism, named Pride, for PRIcing the time-Discounting value of quEries. It aims to minimize the cumulative regret of online selling queries, with a regret upper bound guarantee.

Fig. 2 illustrates the architecture of Pride, where the data buyers arrive online. The t-th arriving data buyer b_t establishes a query Q_t with the feature vector $\mathbf{x}_t$. The buyer b_t also has a private valuation v_t on the query Q_t. The input of Pride includes a sequence of customized queries in the form of $Q_1, Q_2, \cdots, Q_T$ from data buyers that freely come to the online data market. Pride is responsible for posting the query prices in the form of $p_1, p_2, \cdots, p_T$ for the query requests, and getting the binary trade feedback (i.e., buy it or not).

Pride incorporates *three* major functionality modules to derive a proper posted price for each query trade: i) *knowledge set update* (KSU) module, that uses the only *binary feedback* from the data buyer to dynamically tighten the *knowledge set* of the market value; ii) *feature-based price calculation* (FPC) module, that borrows an *efficient and effective ellipsoid technique* to shape the knowledge set and thus derives a *feature-based price* to initially estimate the query valuation; iii) *discount factor determination* (DFD) module, that leverages a *biased Thompson sampling method* to select an optimal discretized discount factor to *further track the time-discounting trend* of the market value and thus refine the query valuation estimation. It has a sub-linear regret upper bound in *non-stationary MAB* environment. The query price is derived by the multiplication of the feature-based price and discount factor.

It is worth noting that, the regret of Pride results from the FPC module, DFD module (w.r.t. Biased-TS algorithm), and discretization error. The regret of Pride is bounded by $O(\frac{T}{K})$. The worst-case regret of Biased-TS is $O(K^3\sqrt{T \ln T} + K \exp\{4\sqrt{\ln T}\})$, growing sublinearly with T.

As shown in Fig. 2, Pride first adopts the *knowledge set update* (KSU) module to progressively and roughly learn the data market value. The initial knowledge set $\mathcal{K}_1$ is set as the full set, i.e., the maximal set, (and thus definitely containing the real data market value). Pride updates the *knowledge set* $\mathcal{K}_t$ by using the previous knowledge set $\mathcal{K}_{t-1}$ with the feedback f_{t-1} (to be detailed in Section V-A). There is a new linear inequality about the market value space information added to the knowledge set per round. In this way, the knowledge set goes shrinking over time. That is, the estimation of the market value becomes more and more precise.

The *feature-based price calculation* (FPC) module of Pride introduces an *exploration-exploitation* strategy to offer a *feature-based price* dynamically (based on the knowledge set), to make a trade-off between the immediate revenue and long-term benefit. It roughly estimates the query valuation. In particular, if the knowledge of the time-discounting market value is not adequate, it prefers to choose an *exploration price* as the feature-based price to explore the range information of the market value for *long-term benefit*. Otherwise, an *exploitation price* is adopted to fully utilize the current knowledge for *immediate revenue*. The exploration price and exploitation price are derived based on an efficient ellipsoid technique. It avoids solving two time-consuming linear programs with the constraint of multiple n-dimensional linear inequalities. The FPC module will be described in Section V-B.

The *discount factor determination* (DFD) module of Pride utilizes a novel Biased-TS algorithm to select the time-discounting factors for capturing the discount trend of the market value, so as to refine the feature-based price. Biased-TS approximates a continuous trend with discrete values, as the unknown continuous discount trend is difficult to learn. In other words, it selects the most suitable value (from a set of discrete candidates) as the discount factor d_t in round t. As to be elaborated in Section V-C, Biased-TS employs a new *biased update* strategy to explore the lower discount factor in the subsequent rounds. It adopts *combinatorial probability* as the factor selection criteria to make an exploration-exploitation trade-off. Biased-TS also uses feedback from the data buyer to update its parameters for pursuing better future performance.

Pride finally posts the query price $p_t = p_t^o \cdot d_t$ and collects the buyer's feedback on purchasing the query at the price p_t or not. This feedback will be used to refine the knowledge set and to find the proper discount factor in the next round. The whole procedure of Pride will be introduced in Section V-D.

It is worthwhile to mention that, Pride can provide hindsight for solving the data pricing with time-sensitive value, not just discounting. For example, the data broker can learn the time-increasing trend of the data value by dynamically selecting candidate values (that are greater than 1).

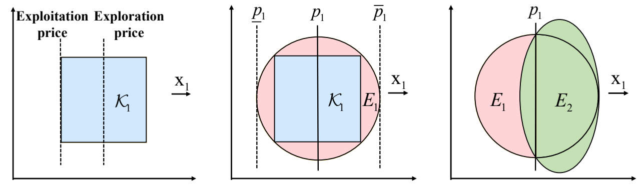
V. Online Query Pricing Mechanism

In this section, we elaborate on the details of Pride with three functionality modules.

A. Knowledge Set Update

The query valuation defined in Eq. 1 can be expressed as $v_t = d(t) \cdot \mathbf{x}_t^T \cdot \boldsymbol{\theta}^*$. Both the original market value $\boldsymbol{\theta}^*$ and discount function $d(t)$ are unknown to the data broker.

The *knowledge set update* (KSU) module tries to gain a good *knowledge* of the time-discounting market value $d(t) \cdot \boldsymbol{\theta}^*$, to derive a proper posted price for revenue maximization. It collects the decisions (or feedback) from data buyers to refine the knowledge of the market value in an online fashion. Let $\mathcal{K}_t$ denote the knowledge set of the time-discounting market value in round t. $\mathcal{K}_1$ represents the initial knowledge set containing the real $\boldsymbol{\theta}^*$. Supposed that each element $\theta_i^* \in \boldsymbol{\theta}^*$ initially belongs to an interval $[l_i^*, u_i^*]$ at the first round, i.e., $l_i^* \leq \theta_i^* \leq u_i^*$, $i = 1, 2, \cdots, n$. Then, we have $\mathcal{K}_1 = \{\theta_i^* \in \mathbb{R} \mid l_i^* \leq \theta_i^* \leq u_i^*,\ i = 1, 2, \cdots, n\}$.



(a) Prices (b) Before cutting (c) After cutting

Fig. 3. Pricing based on the ellipsoid

For each collected decision, a new linear inequality will be added to the knowledge set. Specifically, if posted price p_t is rejected (resp. accepted), the knowledge set $\mathcal{K}_t$ will be used to derive/update $\mathcal{K}_{t+1} = \mathcal{K}_t \cap \{\boldsymbol{\theta} \in \mathbb{R}^n \mid p_t \geq \mathbf{x}_t^T \boldsymbol{\theta}\}$ (resp. $\mathcal{K}_{t+1} = \mathcal{K}_t \cap \{\boldsymbol{\theta} \in \mathbb{R}^n \mid p_t \leq \mathbf{x}_t^T \boldsymbol{\theta}\}$). It means that the knowledge set/space is narrowing down, and thus the feasible space of the market value turns smaller, indicating a more precise market value estimation. As further detailed in the next subsection, Pride adopts an ellipsoid to replace the n-dimensional knowledge set $\mathcal{K}_t$ for its feasibility and efficiency.

B. Feature-based Price Calculation

For a query Q_t with the feature vector $\mathbf{x}_t$, the *feature-based price calculation* (FPC) module is to get a price (for further getting the posted price) based on the knowledge set $\mathcal{K}_t$. This price is called *feature-based* price. The FPC module employs a novel *exploration-exploitation* strategy, in which there is an *exploitation price* and an *exploration price*, and one of them is strategically set as the feature-based price. It also employs an *ellipsoid* technique to accelerate the price calculation.

One intuitive solution for posting a proper price p_t is to derive an interval with a lower bound and an upper bound to roughly estimate the query valuation. Specifically, for a query Q_t and the current knowledge set $\mathcal{K}_t$, one can derive a lower bound valuation (or price) $\underline{p}_t = \min_{\boldsymbol{\theta} \in \mathcal{K}_t} \mathbf{x}_t^T \boldsymbol{\theta}$ and an upper bound valuation (or price) $\overline{p}_t = \max_{\boldsymbol{\theta} \in \mathcal{K}_t} \mathbf{x}_t^T \boldsymbol{\theta}$ on estimating the query valuation v_t. The exploitation price is exactly the lower bound price $\underline{p}_t$. It will be accepted by the data buyer with the highest probability. In this case, the attained revenue may be very limited and the knowledge set remains unchanged. It thus does not benefit future trade. By contrast, the *exploration price* is set as $\frac{\underline{p}_t + \overline{p}_t}{2}$. Fig. 3(a) illustrates a schematic diagram of exploitation and exploration prices in a 2D situation. If the exploration price is chosen as the feature-based price, the space of the knowledge set will be narrowed down by half (indicating a more precise estimation of the market value that benefits the subsequent query trade). While the exploration price brings more revenue, it also suffers a high probability of being rejected by the data buyer.

To this end, the exploration-exploitation strategy in the FPC module uses a threshold ε_1 to judge the "size" of the knowledge set and make a trade-off between exploitation and exploration. That is if $\overline{p}_t - \underline{p}_t \leq \varepsilon_1$, the feature-based price is set as the exploitation price, otherwise it is set as the exploration price. So far, how to get the feature-based price seems clear, other than the lower and upper bound prices derivation within n-dimensional space. Actually, it has to solve two linear programs with the constraint of n-dimensional linear inequality set (i.e., knowledge set). It is quite time-consuming, especially when n is large.

As a result, we leverage an *efficient ellipsoid* method [49] to address this issue. The key idea of the method is to replace the raw knowledge set $\mathcal{K}_t$, viewed as a polytope in geometry, with the ellipsoid E_t (as defined in Definition 4) of the minimum volume that contains $\mathcal{K}_t$. E_t is called the Löwer-John ellipsoid of the convex body $\mathcal{K}_t$. Fig. 3(b) and Fig. 3(c) illustrate how the ellipsoid works in the 2D situation. Initially, we use the ball E_1 with the radius $R = \sqrt{\sum_{i=1}^{n} \max(l_i^{*2}, u_i^{*2})}$ and the center $\mathbf{c}_1$ to contain the initial knowledge set $\mathcal{K}_1$. Given the feature vector $\mathbf{x}_1$, we can get the exploration price p_1 (i.e., the solid line in Fig. 3(b)) based on ellipsoid E_1. If the price is accepted, E_1 will be cut off by the line $\{\boldsymbol{\theta} \in \mathbb{R}^n \mid \mathbf{x}_1^T \boldsymbol{\theta} = p_1\}$. In Fig. 3(c), the ellipsoid E_2 is then updated, containing the intersection between E_1 and half-space $\{\boldsymbol{\theta} \in \mathbb{R}^n \mid \mathbf{x}_1^T \boldsymbol{\theta} \geq p_1\}$.

Definition 4: (**Ellipsoid**). $E \subseteq \mathbb{R}^n$ is an ellipsoid, if there exists a vector $\mathbf{c} \in \mathbb{R}^n$ and a positive definite matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, such that

$$E = E(\mathbf{A}, \mathbf{c}) = \{\boldsymbol{\theta} \in \mathbb{R}^n \mid (\boldsymbol{\theta} - \mathbf{c})^T \mathbf{A}^{-1} (\boldsymbol{\theta} - \mathbf{c}) \leq 1\} \quad (3)$$

where the vector $\mathbf{c}$ is the central vector of the ellipsoid, and matrix $\mathbf{A}$ contains the shape characteristics of the ellipsoid.

Using the property of the ellipsoid, the upper and lower bound prices can be efficiently calculated. Formally, let $\mathbf{A}_t$ and $\mathbf{c}_t$ be the shape matrix and center vector in round t, respectively. The maximum signed direction from the ellipsoid center to the boundary, denoted by $\mathbf{h}_t \in \mathbb{R}^n$, can be calculated by $\mathbf{h}_t = \mathbf{A}_t \mathbf{x}_t / \sqrt{\mathbf{x}_t^T \mathbf{A}_t \mathbf{x}_t}$. Given a query feature vector $\mathbf{x}_t$, the upper and lower bound prices can be derived by

$$\underline{p}_t = \mathbf{x}_t^T (\mathbf{c}_t - \mathbf{h}_t) \quad \text{and} \quad \overline{p}_t = \mathbf{x}_t^T (\mathbf{c}_t + \mathbf{h}_t). \quad (4)$$

Recall that, the exploration price is adopted if $\overline{p}_t - \underline{p}_t > \varepsilon_1$. It means that, when the ellipsoid (or knowledge set) is not narrow enough, the central vector $\mathbf{c}_t$ of the ellipsoid is used to estimate the time-discounting market value vector $d(t) \cdot \boldsymbol{\theta}^*$. Otherwise, the vector $(\mathbf{c}_t - \mathbf{h}_t)$ that represents the lower boundary vector is used to conservatively estimate the market value.

C. Discount Factor Determination

Although the knowledge set (or the ellipsoid) of the time-discounting market value has been well modeled and updated by the KSU and FPC modules, it may still fail in learning the time-discounting trend of the market value for two reasons.

First, the ellipsoid is shrinking fast due to the *central cut* [50]. It thus cannot contain the time-discounting market value for a long time. In Fig. 4(a), the ellipsoid E_t and the market value $\boldsymbol{\theta}_t^* = d(t) \cdot \boldsymbol{\theta}^*$ in round t are represented by circles and small dots, respectively. The filled shapes are used to highlight the notations for the “current” round. The arrows show the discount trend of the market value. As depicted in the figure, the exploration price (i.e., the dash-line crossing the center of the ellipsoid) always cuts the half-volume of the ellipsoid. It is called the *central cut*, which means that half of the ellipsoid’s volume will be cut each time. After the price is accepted, the left half of the ellipsoid E_1 is cut down, and the

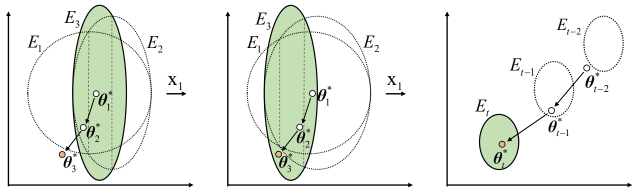

E_3
E_1
E_2
$\mathbf{x}_1$
$\boldsymbol{\theta}_1^*$
$\boldsymbol{\theta}_2^*$
$\boldsymbol{\theta}_3^*$
E_{t-2}
E_{t-1}
E_t
$\boldsymbol{\theta}_{t-2}^*$
$\boldsymbol{\theta}_{t-1}^*$
$\boldsymbol{\theta}_t^*$
(a) Central cut (b) Shallow cut (c) Trend tracking

Fig. 4. Different situations on the ellipsoid

remaining part of E_1 is updated to the ellipsoid E_2. Similarly, when the price is rejected, the process is carried out from E_2 to E_3. Central cuts are effective for narrowing down the ellipsoid in an *unchanging* environment. However, in a time-discounting scenario, central cuts may cause the ellipsoid to not contain the actual market value fast, as the $\boldsymbol{\theta}_3^*$ is out of the ellipsoid E_3. This leads to a useless knowledge set.

Second, the updates made to the knowledge set do not align with the discount trend of the data market value. As depicted in Fig. 4(a), the ellipsoid updates along or opposite the direction of the query feature $\mathbf{x}_1$, instead of the changing direction of the data market value. In addition, once $\overline{p}_t - \underline{p}_t \leq \varepsilon_1$, for any given feature $\mathbf{x}_t$, the FPC module will keep offering a fixed *exorbitant* exploitation price in the following rounds, while the real market value is ever decreasing. This *exorbitant* posted price makes the data unsellable, incurring much regret.

In light of this, we arm Pride with the module of *discount factor determination* (DFD) to dynamically select the discount factor $d_t \in [0, 1]$ for well capturing the time-discounting trend of the market value in online pricing. The multiplication of the feature-based price and discount factor (respectively derived by FPC and DFD modules) forms the posted price of Pride.

From a theoretical standpoint, the discount factor helps to carefully modify the ellipsoid. As shown in Fig. 4(b), when the price is accepted, a *shallow cut* [50] is made to reduce less than half of the ellipsoid volume, allowing the knowledge space to contain the market value for longer and refine the knowledge set further. Conversely, when the price is rejected, more than half of the volume will be cut down by a *deep cut* [50], as shown in the process from E_2 to E_3. It greatly narrows the feasible space. The discount factors also increase the acceptance probabilities of prices, thus reducing the regret.

In addition, dynamically setting the discount factor can also track the discount trend of the market value, when the ellipsoid cannot contain the market value vector. As depicted in Fig. 4(c), the ellipsoid E_{t-2} cannot contain the data market value $\boldsymbol{\theta}_{t-2}^*$ in round $(t-2)$. After properly setting the discount factor, the ellipsoid can continue to follow the discount trend of the market value in the following rounds.

However, it is difficult to dynamically set a proper discount factor per round in the time-discounting setting. On the one hand, a lower discount factor is more expected for future trade due to the discount trend of the market value. On the other hand, if it is too low, it will generate unnecessary regret.

To this end, we model the problem as the *non-stationary MAB optimization problem*. We propose a novel non-stationary MAB algorithm Biased-TS (Biased Thompson Sampling) in the DFD module to dynamically select the optimal discount

Algorithm 1: Biased-TS algorithm

```
Input: Candidate set $\mathbb{F} = \{\mu_0, \mu_1, \mu_2, \cdots, \mu_K\}$
Output: Selected candidate set $\mathbb{F}^* = \{\mu_1^*, \mu_2^*, \cdots, \mu_T^*\}$
for $t = 1, 2, \cdots, T$ do
    if $t \leq K + 1$ then
        $I_t = t - 1$                      // initialization stage
    else
        for $k = 0, 1, 2, \cdots, K$ do
            $P_k = w_k / \sum_i w_i$; $\vartheta_k \sim Beta(\alpha_k, \beta_k)$
            $\tilde{\vartheta}_k = (1-\gamma) \cdot P_k + \gamma \cdot \vartheta_k$    // combinat. prob.
        $I_t = \arg\max(\tilde{\theta}_k)$
    if $\mu_{I_t}$ is accepted then
        for $i = 0, 1, \cdots, I_t$ do
            $\alpha_i \leftarrow \alpha_i + U_i$         // biased update strategy
        $w_{I_t} \leftarrow w_{I_t} + 1$
    else
        for $i = I_t, I_t + 1, \cdots, K$ do
            $\beta_i \leftarrow \beta_i + U_i$         // biased update strategy
    $\mu_t^* \leftarrow \mu_{I_t}$
return $\mathbb{F}^*$
```

factor from candidates. It approximates the continuous time-discounting trend by the discretization values.

Specifically, Biased-TS is developed based on the celebrated Thompson sampling (TS) method [51]. TS is famous for its simple structure, guaranteed performance, and wide applications [52]–[54]. However, the traditional TS cannot handle the time-discounting setting, as it may be stubborn to one "best" choice, instead of changing with the environment. Thus, in contrast to the basic TS, there are three improvements in Biased-TS. It introduces an initialization stage to better understand the time-discounting level at the very beginning. It utilizes a *biased update* strategy to incline to select the small discount factor for future trade. It selects the discount factor via comparing *combinatorial probabilities* (instead of sampling values) to balance the exploration and exploitation.

Algorithm 1 gives the pseudo-code of Biased-TS. Similar to TS, Biased-TS uses a beta distribution $Beta(\alpha, \beta)$ as a prior distribution to describe the probability of the candidate being selected. The *candidate set* $\mathbb{F} = \{\mu_0, \mu_1, \mu_2, ..., \mu_K\}$ collects $(K+1)$-candidates of time-discounting factors, where each candidate $\mu_k \in [0,1]$, $k = 0, 1, 2, \cdots, K$. K is the number of discount factor candidates, controlling the discretization degree. Each candidate is assigned a probability distribution $Beta(\alpha_k, \beta_k)$, $k = 0, 1, 2, ..., K$, and I_t is used to represent the identity of the chosen candidate in round t. The *initialization stage* of Biased-TS takes place in the first $(K+1)$ rounds, during which each candidate tries to gain *initial belief* of the time-discounting level (lines 2-3). This stage filters out some exorbitant factors, which can reduce the regret due to offering too large values. Note that, transferring initial belief from one dataset to another may improve performance if prior knowledge of the time-discounting trend exists.

In the following round t, Biased-TS turns to calculate the *combinatorial probability* of each candidate. It selects the discount factor with the highest combinatorial probability (lines 5-8). w_k denotes the chosen times of the k-th candidate. P_k weights the historical performance of candidates, representing exploitation of the current best choice (line 6). Biased-TS gets the weight ratio P_k and draws a sampling value ϑ_k from the beta distribution of each candidate μ_k $(k = 0, 1, 2, \cdots, K)$. It calculates the combinatorial probability $\tilde{\vartheta}_k$ by

$$\tilde{\vartheta}_k = (1-\gamma) \cdot P_k + \gamma \cdot \vartheta_k, \tag{5}$$

where the balance parameter $\gamma \in [0,1]$ is used to balance the exploration and exploitation (line 7). In particular, the sampling value tends to favor the exploration of small discount factors over time, due to the *biased update* mentioned later. This can cause Biased-TS to be too "conservative" in choosing small discount factors, resulting in less revenue. To address this, the weight ratio is used to give more weight to the current optimal choice and balance exploration and exploitation.

Since the smaller discount factors are expected to be selected in the future, Biased-TS captures this property through a *biased* update strategy. Specifically, after receiving the feedback $f_t \in \{0, 1\}$ (i.e., accept or not), it updates its parameters as follows, and $U_k = \lambda^k$ $(\lambda \geq 1)$ is the step size of the update.

$$Beta(\alpha_k + f_t \cdot U_k, \beta_k + (1 - f_t) \cdot U_k) \tag{6}$$

If the currently selected factor is accepted, the beta distribution parameters of all candidates less than or equal to the chosen factor will be updated (lines 10-11). Besides, the weight of the selected arm μ_{I_t} will be updated by $w_{I_t} = w_{I_t} + 1$ (line 12). Otherwise, if the arm is rejected, the beta distribution parameters of all candidates greater than or equal to the selected one are updated (lines 14-15). The intuition behind this is that, if a price is accepted, all prices less than that price will be accepted by the buyer out of personal rationality. If the price is rejected, the buyers will also reject the higher price. In addition, the candidate with a larger value will get a larger step size of the update. It helps to realize that, the higher price is better when the query can be sold (line 11). Conversely, if the price cannot be accepted, it means that the algorithm is too "greedy" and it will be punished for this action (line 15).

D. Online Query Pricing Mechanism

Based on the three functionality modules, we propose our PRIcing mechanism for the time-Discounting quEry Pride.

The pseudo-code of Pride is given in Algorithm 2. When a data buyer b_t arrives in round t, Pride first derives a feature-based price p_t^o. If $\overline{p}_t - \underline{p}_t > \varepsilon_1$, then the exploration price $\frac{\underline{p}_t + \overline{p}_t}{2}$ is adopted, otherwise the exploitation price $\underline{p}_t$ is chosen (lines 2-8). If $\frac{t}{T} > \varepsilon_2$, Pride sets a discount factor d_t by invoking the DFD module (i.e. Biased-TS) and posts the price $p_t = p_t^o \cdot d_t$ (lines 9-10), otherwise the feature-based price will be posted directly (lines 11-12). Note that, to better integrate the FPC module with the DFD module, we do not select the discount factor from the very beginning. It is motivated by an interesting observation that, in the early stage of online pricing, the knowledge set space is broad and the data broker only has a very coarse estimation of the market value. Hence,

Algorithm 2: Query pricing mechanism Pride

```
Input: $\mathbf{A}_1 = R^2\mathbb{I}^{n\times n}$, $\mathbf{c}_1 = \mathbf{0}^{n\times 1}$, discount factors $\mathbb{F}$
Output: The posted price $p_t$ in each round $t \in \mathbb{T}$
1  for $t = 1, 2, ..., T$ do
2      $E_t = \{\boldsymbol{\theta} \in \mathbb{R}^n | (\boldsymbol{\theta} - \mathbf{c}_t)^T \mathbf{A}_t^{-1} (\boldsymbol{\theta} - \mathbf{c}_t) \leq 1\}$
3      receive a query feature $\mathbf{x}_t \in \mathbb{R}^n$
4      $\mathbf{h}_t = \frac{\mathbf{A}_t \mathbf{x}_t}{\sqrt{\mathbf{x}_t^T \mathbf{A}_t \mathbf{x}_t}}$; $\underline{p}_t = \mathbf{x}_t^T(\mathbf{c}_t - \mathbf{h}_t)$; $\overline{p}_t = \mathbf{x}_t^T(\mathbf{c}_t + \mathbf{h}_t)$
5      if $\overline{p}_t - \underline{p}_t > \varepsilon_1$ then
6          $p_t^o = \max\{0, \frac{\underline{p}_t + \overline{p}_t}{2}\}$
7      else
8          $p_t^o = \max\{0, \underline{p}_t\}$
9      if $\frac{t}{T} > \varepsilon_2$ then
10         invoke the DFD module to get $d_t$; $p_t = p_t^o \cdot d_t$
11     else
12         $p_t = p_t^o$
13     if the exploration price is chosen then
14         $D_t = \frac{\mathbf{x}_t^T \mathbf{c}_t - p_t}{\sqrt{\mathbf{x}_t^T \mathbf{A}_t \mathbf{x}_t}}$
15         if $p_t$ is rejected and $-\frac{1}{n} \leq D_t \leq 1$ then
16             $\mathbf{A}_{t+1} = \frac{n^2(1-D_t^2)}{n^2-1}(\mathbf{A}_t - \frac{2(1+nD_t)}{(n+1)(1+D_t)}\mathbf{h}_t\mathbf{h}_t^T)$
                $\mathbf{c}_{t+1} = \mathbf{c}_t - \frac{1+nD_t}{n+1}\mathbf{h}_t$
17         else if $p_t$ is accepted and $-\frac{1}{n} \leq -D_t \leq 1$ then
18             $\mathbf{A}_{t+1} = \frac{n^2(1-D_t^2)}{n^2-1}(\mathbf{A}_t - \frac{2(1-nD_t)}{(n+1)(1-D_t)}\mathbf{h}_t\mathbf{h}_t^T)$
                $\mathbf{c}_{t+1} = \mathbf{c}_t + \frac{1-nD_t}{n+1}\mathbf{h}_t$
19         else
20             $\mathbf{A}_{t+1} = \mathbf{A}_t$; $\mathbf{c}_{t+1} = \mathbf{c}_t$
21     else
22         $\mathbf{A}_{t+1} = \mathbf{A}_t$; $\mathbf{c}_{t+1} = \mathbf{c}_t$
23     update the parameters of the DFD module
```

the threshold parameter ε_2 is used to control a certain "learning phase" to effectively shrink the market value estimation.

Pride then updates the parameters of the ellipsoid and Biased-TS, after receiving the decision from the data buyer (lines 13-23). To be more specific, it first calculates a parameter D_t to locate the cutting hyperplane (or cutting line in 2D) to cut the current ellipsoid. This parameter denotes the signed distance from the ellipsoid center to the cutting hyperplane in the space $\mathbb{R}^n$ endowed with the norm $\|\cdot\|_{A_t^{-1}}$ (line 14).

When the posted price is rejected, Pride retains the intersecting piece between the current ellipsoid E_t and the half-space $\{\boldsymbol{\theta} \in \mathbb{R}^n \mid \mathbf{x}_t^T\boldsymbol{\theta} \leq p_t\}$ if the inequality $-\frac{1}{n} \leq D_t \leq 1$ is satisfied. It computes the Löwer-John ellipsoid E_{t+1}, according to Grötschel's work [50] (lines 15-16). By the symmetry of the ellipsoid, the formulas in the acceptance branch can be obtained (lines 17-18). Otherwise, the new ellipsoid E_{t+1} in the next round has the unchanged center vector $\mathbf{c}_{t+1}$ and shape matrix $\mathbf{A}_{t+1}$ (line 20). If the exploitation price is adopted, the ellipsoid will remain unchanged (line 22). Moreover, Pride adopts the *biased update* strategy (described in Biased-TS) to update the related parameters for selecting a suitable discount factor in the subsequent rounds (line 23).

Compared to the typical ellipsoid-based pricing method [14], [41], Pride does not incur extra time and space complexity overhead (even though it considers the discounting data value). The time and space complexities of Pride are dominated by the FPC module. Both are $O(n^2)$ per round. In particular, the computation overhead in a single round mainly comes from the matrix-vector and vector-vector multiplications. The space complexity is mainly caused by the shape and the center parameters of the ellipsoid.

VI. Theoretical analysis

In this section, we analyze the regret upper bound of our pricing mechanism Pride.

The main target of Pride is to maximize the cumulative revenue which equals the problem of cumulative regret minimization. The cumulative regret of the pricing mechanism Pride mainly arises from i) the regret of discount factor determination (i.e., Biased-TS) derived in Corollary 1, ii) the regret of feature-based price calculation, and iii) the discretization error [34]. The discretization error is $O(\frac{1}{K})$, as Biased-TS incurs a regret of $O(\frac{1}{K})$ per round due to the discretization (despite it can choose the best discount factor). We prove that, the cumulative regret of Pride is bounded by $O((1-\frac{1}{K})n^2 \ln\frac{T}{n} + K^3\sqrt{T\ln T} + K\exp\{4\sqrt{\ln T}\} + \frac{T}{K})$, as stated in Corollary 2.

A. Regret Upper Bound of Discount Factor Determination

The regret analysis of discount factor selection by Biased-TS can be transferred to the problem of estimating the number of rounds where the sub-optimal arms (i.e., candidates) are chosen. To facilitate the analysis, we first prove the confidence interval of the variable from the beta distribution in Theorem 1 to estimate the sampling value from the beta distribution.

We decompose the regret analysis of Biased-TS into two situations: a) an *underestimated* case and b) a *well-estimated* case. For Case a), we try to bound the number of sub-optimal rounds with a worst-case assumption. For Case b), we directly utilize the upper and lower confidence bounds from Theorem 1 to estimate the rounds that generate regret. We derive the regret upper bound of Biased-TS in Corollary 1.

Biased-TS adopts the combinatorial probability $\tilde{\vartheta}$ to select the "best choice", which consists of a sampling value ϑ from beta distribution and a weight ratio P that can be explicitly calculated. However, it is difficult to figure out the exact sampling value in every round due to the uncertainty of sampling. Hence, we try to construct the confidence interval by combining the lower and upper confidence bound, to better estimate the sampling value for regret analysis.

Theorem 1: (**The confidence interval of beta variable**). Consider a random variable $Beta$ with beta distribution $Beta(s+1, t-s+1)$, where $s = \sum_{i=1}^{t} X_i$ is the sum of $t \in \mathbb{N}$ Bernoulli trails $X_i \sim Be(\phi)$ with parameter $\phi \in [0,1]$. For a finite integer $T \in \mathbb{N}$, $T > t$, let $u_t = \frac{s}{t} + \sqrt{\frac{\ln T}{t}}$ denote the upper confidence bound, and $l_t = \frac{s}{t} - \sqrt{\frac{\ln T}{t}}$ denote the lower confidence bound in round t. It can derive that,

$$Pr(l_t \leq Beta \leq u_t) \geq 1 - \frac{1}{T^2}(1 + \exp\{4\sqrt{\frac{\ln T}{t}}\}) \quad (7)$$

Proof. From the upper confidence bound of the beta variable in [55], we have $Pr(Beta \leq u_t) \geq 1 - \frac{1}{T^2}$. We can derive the lower confidence bound of the beta variable in a similar way. That is, $Pr(Beta \leq l_t) \leq \exp\{4\sqrt{\frac{\ln T}{t}}\}\frac{1}{T^2}$. □

We employ this *certain* confidence interval to estimate the *uncertain* sampling value from the beta distribution, and thus quantify the variables of each candidate. It can greatly facilitate the analysis of the candidate-choosing process.

We analyze the situation that may raise regret from the perspective of the non-stationary MAB problem. The possible occurrence of regret in Biased-TS can be roughly divided into two situations: i) the optimal arm is underestimated, ii) the optimal arm is well-estimated, but a sub-optimal arm is chosen. We can derive the regret upper bound of Biased-TS.

Corollary 1: (**Regret upper bound for Biased-TS**). Given the exponential update-policy $U_k = \lambda^k$ ($\lambda \geq 1$), Biased-TS has a regret upper bound of $O(K^3\sqrt{T\ln T}+K\exp\{4\sqrt{\ln T}\})$.

Proof. Given $(K+1)$ candidates $\mathbb{F} = \{\mu_0, \mu_1, \cdots, \mu_K\}$ of discount factors. For representation, the whole time horizon T is divided into $(K+1)$ segments $\mathbb{S} = \{S_0, S_1, ..., S_K\}$, where S_i is the set of rounds in which μ_i is the optimal discount factor. Formally, $S_i = \{t \in T \mid \mu_i \leq d(t) \leq \mu_{i+1}\}$.

We define the lower bound and upper bound of the combinatorial probability of candidate k in round t: $\tilde{l}_{t,k} = (1-\gamma)\cdot P_{t,k} + \gamma \cdot l_{t,k}$ and $\tilde{u}_{t,k} = (1-\gamma)\cdot P_{t,k} + \gamma\cdot u_{t,k}$. Let t_k^s and t_k^e denote the start and end points of segment S_k, respectively. $i(t)$ and $i^*(t)$ denote the selected arm and the optimal arm in round t, respectively. For the first $(K+1)$ rounds, the generated regret can be bounded by $R_K = K+1$. For the rest rounds, we let R_r denote this part regret and we have:

$$R_r(T) \leq \sum_{t=1}^{T} E[\mathbb{1}\{i(t) \neq i^*(t)\}] = \sum_{k=0}^{K}\sum_{t=t_k^s}^{t_k^e} E[\mathbb{1}\{i(t) \neq k\}]$$

To determine whether the optimal arm is underestimated, we assume that if $\tilde{l}_{tk} > \tilde{u}_{ti}$ ($i \in [0,K]$, $i \neq k$), the optimal arm is considered to be well-estimated. This assumption puts the analysis into a worst-case for the optimal arm can be chosen only when its lower bound of the confidence interval is relatively high. Thus, in any segment S_k, we have

$$\sum_{t=t_k^s}^{t_k^e} E[\mathbb{1}\{i(t)\neq k\}] = \sum_{t=t_k^s}^{t_k^e} Pr(\tilde{\vartheta}_{ti(t)} \geq \tilde{\vartheta}_{tk}, \tilde{u}_{ti(t)} \geq \tilde{l}_{tk}) + \sum_{t=t_k^s}^{t_k^e} Pr(\tilde{\vartheta}_{ti(t)} \geq \tilde{\vartheta}_{tk}, \tilde{u}_{ti(t)} < \tilde{l}_{tk})$$

Let LHS and RHS denote the first and second parts of the right side of the above equality. t_k is the moment of $\tilde{u}_{t,k+1} = \tilde{l}_{tk}$. As the μ_{k+1} and μ_k is always accepted in the first $(B-k-1)$ segments due to our biased update, we have $LHS \leq K(\Delta_k + KR(t_k^s))$, where Δ_k indicates the rejected rounds of μ_{k+1} in segment S_k, and $R(t_k^s)$ is the number of rounds that generate regret before S_k. Moreover, for $\tilde{u}_{t_k,k+1} = \tilde{l}_{t_k k}$, we can derive that $\Delta_k \leq \frac{1}{g}(\frac{1-\gamma}{\gamma}C_K + \sqrt{T\ln T})$, where g is a constant. Especially, we assume that constant C_K can bound the number of rounds of the k-th arm being chosen.

For RHS, due to the condition of $\tilde{u}_{ti(t)} < \tilde{l}_{tk}$, a sub-optimal arm being chosen will only happen in two cases: i) the sampling value of $i(t)$-th arm is larger than its upper bound, and ii) the sampling value of k-th arm is lower than its lower bound. Thus, using Theorem 1, we have

$$RHS \leq \sum_{i=0,i\neq k}^{K}\sum_{t=t_{k1}}^{t_{k2}} \frac{1}{(tU_i)^2} + \exp\{4\sqrt{\ln T}\}\sum_{t=t_{k1}}^{t_{k2}}\frac{1}{t^2}$$

By combining the upper bounds of LHS and RHS, it can derive that

$$R_r(T) \leq (\frac{1-\gamma}{\gamma})\frac{K(K+1)^2C_K}{g} + \frac{K(K+1)^2}{g}\sqrt{T\ln T} + \frac{\pi^2}{6}(K+1)(K+\exp\{4\sqrt{\ln T}\})$$
□

Collorary 1 indicates that, the worst-case regret of the Biased-TS grows sublinearly with T. In other words, the regret per round of Biased-TS converges to zero as transactions happen. Besides, too many candidates may bring the burden of learning the best choice and thus generate more regret.

B. Regret Upper Bound of Pride

The regret of our pricing mechanism mainly results from the discount factor determination, feature-based price calculation, and the discretization strategy. The regret of feature-based price calculation actually generates from the exploration rounds. The following regret analysis is thus to identify an upper bound number of rounds of the exploration period T_e.

Lemma 2: Let R denote the radius of the initial ellipsoid and S denote the maximum norm of feature x_t, the feature-based price calculation (FPC) module chooses the exploration prices in at most $20n^2 \cdot \ln(20R\cdot S^2\cdot(n+1)/\varepsilon_1)$ rounds.

Proof. Let ζ be the maximum value of $(1-d_t)$ in T_e. Assume that $\zeta \leq \frac{\sqrt{\mathbf{x}_t^T\mathbf{A}_t\mathbf{x}_t}}{2n\mathbf{x}_t^T\mathbf{c}_t}$, then we have $D_t \in \left[0, \frac{1}{2n}\right]$. We construct an upper bound [50] and a lower bound [14] on the final volume of the ellipsoid by well-studied lemmas to derive the upper bound number of exploration rounds. □

Finally, combining the regret bound of the exploration rounds and Biased-TS, and the discretization error, we can derive the regret upper bound of Pride.

Corollary 2: (**Regret upper bound for Pride**) The worst cumulative regret of Pride can be written as

$$O((1-\frac{1}{K})n^2\ln\frac{T}{n} + K^3\sqrt{T\ln T} + K\exp\{4\sqrt{\ln T}\} + \frac{T}{K}) \quad (8)$$

Proof. Let the thresholds $\varepsilon_1 = O(\frac{n^2}{T})$ [14] and $\varepsilon_2 \leq 20n^2\ln(20RS^2(n+1)/\varepsilon_1)/T$. The cumulative regret of exploration period T_e can be easily bounded by Lemma 2. The discount factor determination regret R_{TD} is derived in Corollary 1, and the regret generated from discretization error R_D is with $O(\frac{1}{K})$ per round. Thus, we can bound the cumulative regret by the following inequality.

$$\begin{aligned} R(T) &= R_{T_e} + R_{TD} + R_D \\ &\leq R\cdot S\cdot T_e + R_{TD} + \frac{R\cdot S}{K}(T-T_e) \end{aligned} \quad (9)$$

where R denotes the radius of the initial ellipsoid and S denotes the maximum norm of feature $\mathbf{x}_t$. □

TABLE II
EXPERIMENTAL PARAMETER TABLE

Parameter	**Value range**	**Default**
# Dimension n	20, 40, 55, 60, 80, 100, $\cdots$, 180	20
# Candidates K	20, 30, 40, $\cdots$, 100	50
x in Base $\beta = 0.99^x$	0.001, 0.005, 0.01, 0.05, 0.5, 1	0.01
Balance parm. γ	0, 0.1, 0.2, $\cdots$, 1	0.8
Threshold ε_2	0.5, 0.1, 0.01, 0.001, 10^{-4}, 10^{-5}	0.01
Update parm. λ	1, 1.1, 1.2, $\cdots$, 2.5	1.6
S.D. σ	0, 0.005, 0.01, 0.015, 0.02, 0.025	0
Probability of rejection p_r	0, 0.002, 0.004, 0.006, 0.008, 0.010	0

According to Corollary 2, we can conclude that, the regret upper bound of Pride is dominated by the linear order $O(\frac{T}{K})$.

VII. Experiments

In this section, we conduct extensive experimental evaluations to verify the performance of our online query pricing mechanism. All experiments were implemented with Python and conducted on an Intel Core 2.80 GHz Server with 192GB RAM, running Ubuntu 18.04 system.

A. Experimental Setup

In the experiments, we use a query feature $\mathbf{x}_t$ with a valuation $v_t = d(t) \cdot \mathbf{x}_t^T \cdot \boldsymbol{\theta}^*$ to simulate data buyer b_t. In round t, the buyer makes a query Q_t with a corresponding feature vector $\mathbf{x}_t$, and accepts the price if $p_t \leq v_t$.

Query feature and market value generation. Two different trading datasets, i.e., noisy linear query (*NLQ*) [14] and *Airbnb* [10], are used to simulate the situations of the personal data and SQL query trading, respectively. (i) *NLQ* is used to generate query feature vectors that represent the privacy compensation for data buyers. Following the related work [14], the elements of the query feature vector $\mathbf{x}_t$ and the market value vector $\boldsymbol{\theta}^*$ are randomly drawn from the given distributions. (ii) The real-world *Airbnb* dataset is used to simulate the SQL query pricing. There are online data buyers who want to buy the *Airbnb* dataset for training the ML model to forecast the rental prices, as exemplified in Section I. Here, the query feature $\mathbf{x}_t$ denotes the usage volume of the SQL query Q_t. We pre-process the data and the final dimension n is 55. To get the query feature, we first generate n-dimensional 0-1 vector representing the attribute selection in the buyer's query, and then multiply it by a digit in $[0, 1]$ to denote the proportion of the data queried by the buyer to the total data volume. To get the market value, the linear regression is conducted to obtain the regression coefficients [14]. As the higher absolute value of coefficient means more influence on regression tasks, we take the absolute value of coefficients (denoting different dimension values) as the market value $\boldsymbol{\theta}^*$.

Time-discounting functions. Pride can support multiple time-discounting cases. We utilize two time-discounting functions, i.e., the *exponential* one $d(t) = \beta^t$, $\beta \in (0, 1)$ [27], [34] and the *linear* one $d(t) = 1 - \frac{t}{T}$ [34], [35]. By default, $\beta = 0.99^{0.01}$ on *NLQ* and $\beta = 0.99^{0.005}$ on *Airbnb*. A new mixed discount setting is also used, in which each buyer has an exponential or linear discount with a probability of 50%.

Baselines. In the experiments, we adopt the ellipsoid-based pricing algorithm (EP) [14] as the baseline to compare with our solution Pride. To further demonstrate the flexibility of Pride and the superiority of Biased-TS, we design to implement the DFD module of Pride with one of three other non-stationary MAB algorithms (for ablation study): SW-TS [55], D-UCB [56], and Biased-UCB [35].

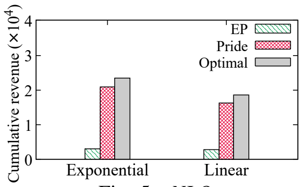


Fig. 5. *NLQ*

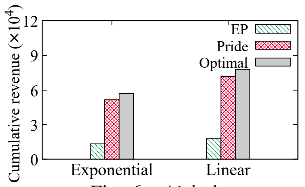


Fig. 6. *Airbnb*

Metrics in the evaluation are cumulative revenue and regret ratio. Cumulative revenue is the total revenue generated from each round. The higher cumulative revenue indicates better performance. The regret ratio is the ratio of cumulative regret to cumulative query valuation, i.e., $\sum_{t=1}^{T} r_t / \sum_{t=1}^{T} v_t$. The results averaged over 100 runs are reported.

Implementation details. We conduct a total of 10,000 and 50,000 rounds of trade on *NLQ* and *Airbnb*, respectively. We set $(K + 1)$ discount factor candidates $\mu_k = \frac{k}{K}$ $(k = 0, 1, \cdots, K)$, and $\varepsilon_1 = \frac{n^2}{T}$ [14]. The candidates' number K is 50 on *NLQ* and 70 on *Airbnb*. The initial radius R is $\sqrt{2n}$ on *NLQ* and 6 on *Airbnb*. The window size τ is $\sqrt{T}$ [55] for SW-TS, the exploration-exploitation control parameter λ_1 is 0.9 [35], and the attenuation factor a is 2 [57] for D-UCB and Biased-UCB. More parameter settings are shown in Table II.

B. Overall Performance Study

The overall performance of Pride. The first set of experiments is to evaluate the pricing performance of Pride and EP, compared with the optimal revenue (i.e., cumulative query valuation). The cumulative revenue of them under the exponential and linear discount functions is reported in Fig. 5 and Fig. 6. Compared to EP, the cumulative revenue of Pride improves around 6 times on *NLQ* and nearly 4 times on *Airbnb*. Pride obtains more than 90% of the optimal revenue. It indicates that, considering the discount factor is necessary.

Fig. 7 depicts the execution time of Pride in 10,000 rounds with changing dimensionality n on *NLQ*. The results indicate that Pride is quite time-efficient. In fact, Pride and EP share the same time complexity. As n increases, the time climbs up as the execution of the matrix-vector and vector-vector multiplications increases, consistent with previous analysis.

To explore the algorithms' performance in different discount environments, Fig. 8 plots the regret ratio of EP and Pride on *Airbnb*. The final regret ratio of Pride in exponential discount and linear discount is about 0.095 and 0.081. Pride outperforms EP in each case. In the first 500 rounds, the regret ratio of Pride exhibits a decreasing trend, similar to EP. This is because that, Pride is still in the initial "learning phase", as analyzed in Section V-D. After a sharp decrease, both algorithms exhibit an increase in the regret ratio. In EP, this is caused by a lack of accounting for the discount trend of data value, which further validates our strategy's effectiveness. For Pride, it is due to the discretization errors discussed in Section

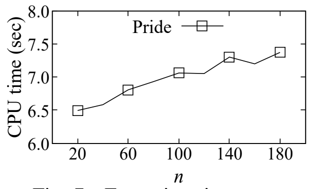


Fig. 7. Execution time *vs.* n

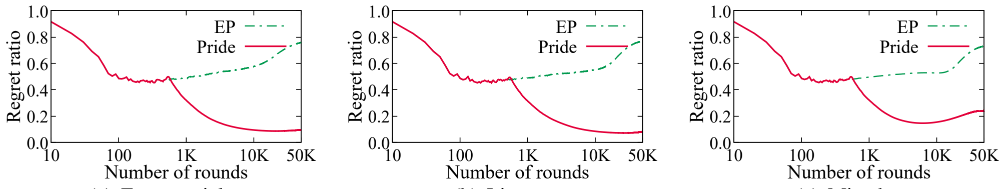


Fig. 8. Regret ratio on *Airbnb* data pricing

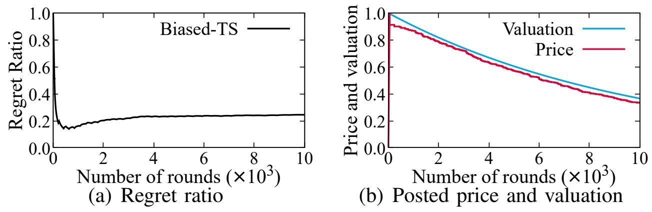


Fig. 9. The performance of Biased-TS

VI. As depicted in Fig. 8(c), the regret ratio in the mixed situation is much higher than former two cases, as the mixed discount functions make the change of value not continuous, increasing the difficulty of learning the time-discounting trend.

Fig. 10 depicts the query valuation derived by Eq. 1 and the posted price per round. The figure clearly shows that, the posted price closely matches the actual query valuation per round. As trades continue, the gap between the query valuation and the posted price decreases. It confirms the effectiveness of Pride. Further, Fig. 11 plots the selected discount factor identity per round, where there are 50 (and 70) candidate discount factors ranking in ascending order for *NLQ* (and *Airbnb*). One interesting observation in Fig. 11(a) is that, after the initialization phase, Pride sticks to choosing one fixed discount factor and lasts for the end. It is because that, the chosen factor is so suitable to make the posted price always accepted. The shallow cut slows down the shrinkage of the knowledge set. Thus, Pride does not need to change the choice of the factor. In contrast, the situation in Fig.11(b) is different. Pride sticks to one candidate in the first 30,000 rounds. Next, it begins to change the choice, as the ellipsoid cannot follow the discount change anymore. The slightly descending trend in a step-like manner is consistent with our theoretical analysis.

The performance of Biased-TS. To verify the discount trend learning effect of Biased-TS, Fig. 9(a) displays the regret ratio of Biased-TS to show its convergence process. Here, the original valuation $\mathbf{x}_t^T\boldsymbol{\theta}^* = 1$ for every buyer, and other settings are the same as the experiments on *NLQ*. After the initial stage, the regret ratio drops to 0.510. During the subsequent rounds, Biased-TS adjusts its choice of discount factor based on the feedback to follow the discount trend. Changes in discount factor selection cause curve fluctuations. Both the sublinear regret and discretization error drive the regret ratio to slowly increase. Starting around the 1,800-th round, the regret ratio change becomes less than 0.0001, and the regret ratio gradually converges to 0.245. Fig. 9(b) depicts the posted price and the valuation. It further confirms that, Biased-TS can learn the discount trend of the data value.

The impact of the market noise and irrational buyers. We study the market noise that is denoted by a variable δ_t from $\mathcal{N}(0, \sigma^2)$ to evaluate its impact on Pride, where the standard deviation (S.D.) σ ranges from 0 to 0.025. As depicted in Fig. 12, Pride performs well when the market fluctuation is small. With the increasing S.D., the regret ratio turns large. The reason is that the greater the fluctuation in market value, the less conducive it is to learn about market value.

To study the influence of buyers' irrational actions on Pride, we let the strategic buyer have a probability p_r to reject the posted price p_t deliberately even when $v_t \geq p_t$, where p_r ranges from 0.002 to 0.01. Fig. 13 shows that, when p_r is low, Pride still achieves some revenue. However, when this probability increases, the regret ratio also climbs up. Thus, how to handle these strategic buyers is our future research.

C. Parameter Study

This set of experiments is to evaluate the impact of different parameters on the performance of Pride using *NLQ* dataset.

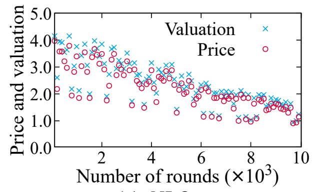
Effect of update parameter λ. Fig. 14 plots the regret ratio of Pride with the changing value of λ. As shown, the regret ratio first quickly drops and then gradually increases with the increase of λ. When λ is too low (i.e., $\lambda < 1.3$), Pride cannot well explore the small discount factor for the future trade; and as it increases, more rounds are required to adapt the change of the optimal discount factor, suffering more regret.

Effect of balance parameter γ. Fig. 15 depicts the regret ratio with the varying γ values. When $\gamma = 0$, Biased-TS selects the optimal arm only based on the historical performance of candidates, with a regret ratio of almost 1. On the contrary, it selects the optimal factor only based on the sampling values when $\gamma = 1$, leading to a higher regret ratio than the cases of $\gamma \in (0, 1)$. It confirms that the combinatorial probability can reduce regret by the exploration and exploitation trade-off.

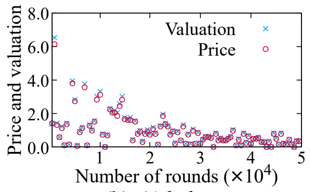
Effect of threshold parameter ε_2. The regret ratio under different ε_2 values on *NLQ* dataset is shown in Fig. 16. The parameter ε_2 controls the time when Biased-TS begins to learn the discount factor. If it starts learning the discount factor too late, Pride will suffer more regret. Because it does not capture the time-discounting characteristic in time. But a certain amount of learning period can improve the algorithm performance, as discussed in Section V.

Effect of the number of candidates K. As shown in Fig. 17, the regret ratio increases when $K < 40$ or $K > 70$. More discretization error is generated when the value of K is small. More regret is generated for selecting the optimal discount factor when the value of K is too large. It is consistent with our theoretical analysis results in Section VI.

Effect of dimensionality n. As the value of n increases in Fig. 18, the regret ratio is also increasing. It is because, the

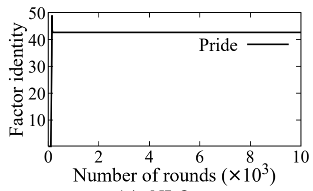


(a) *NLQ*

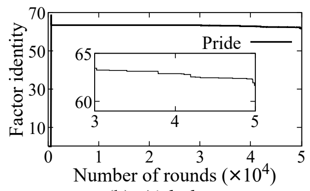


(b) *Airbnb*

Fig. 10. Posted price and query valuation per round



(a) *NLQ*



(b) *Airbnb*

Fig. 11. Selected discount factor per round

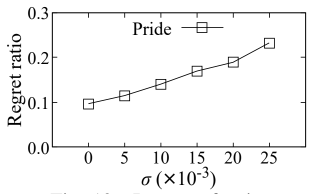


Fig. 12. Impact of noise

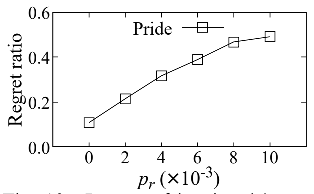


Fig. 13. Impact of irrational buyers

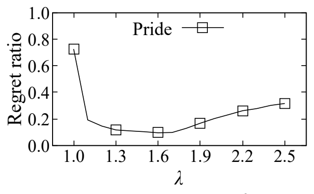


Fig. 14. Impact of λ

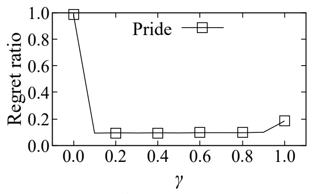


Fig. 15. Impact of γ

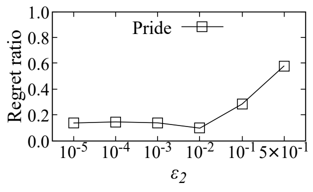


Fig. 16. Impact of ε_2

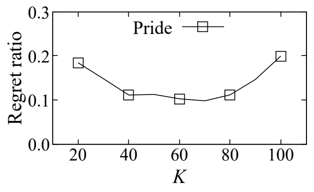


Fig. 17. Impact of K

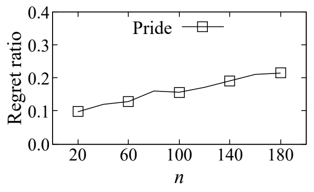


Fig. 18. Impact of n

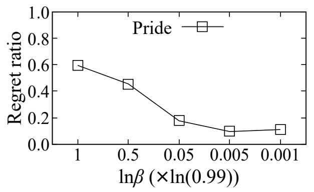


Fig. 19. Impact of β

feature-based price calculation module of Pride needs more rounds for the exploration price when n is large, suffering a higher risk of being rejected than the exploitation price.

Effect of different exponential discounts. Fig. 19 illustrates the regret ratio of Pride in different exponential time-discounting environments. When the discount tends to be too fast (w.r.t. the small value of β), Pride does not have enough rounds to learn the valuations, thus generating more regret.

D. Ablation Study

Our pricing mechanism Pride is quite flexible to consider the time-discounting property of data value (implemented by the DFD module). In this set of experiments, we further investigate the effect of the DFD module on the performance of Pride through ablation study. We test all variants in default exponential-discount settings.

We implement the DFD module with SW-TS [55], D-UCB [56], and Biased-UCB [35], denoted by w/ SW-TS, w/ D-UCB, and w/ Biased-UCB, respectively. Moreover, we generate the three other variants based on Biased-TS but without i) initialization stage (w/o IS), ii) combinatorial probability (w/o CP), and iii) biased update (w/o BU). The corresponding experimental results (i.e., the regret ratio) of the pricing algorithms are reported in Table III.

We can conclude that the performance of the DFD module significantly influences the final pricing performance. Pride using Biased-TS that considers the discount factor is the best in learning the time-discounting value of the data, and each of the components in Biased-TS does make contributions to online query pricing. The good performance of Biased-TS can be attributed to the skillful exploitation-exploration balance.

Furthermore, the biased update strategy used in Biased-TS makes the most obvious contributions, as the regret ratio of w/o BU version increases by up to 4 times over Pride. It

TABLE III
REGRET RATIO FOR ABLATION STUDY

Method	w/ SW-TS	w/ D-UCB	w/ Biased-UCB	w/o IS	w/o CP	w/o BU	Pride
NLQ	0.532	0.198	0.119	0.312	0.208	0.419	**0.109**
Airbnb	0.540	0.289	0.160	0.382	0.247	0.481	**0.095**

demonstrates that the biased update strategy is quite essential and effective to select the suitable discount factor in a discount environment. The initialization stage reduces the regret by more than 2 times. Hence, it is necessary and useful to obtain the rough space of the discount trend at the beginning of online pricing. The adoption of combinatorial probability can improve pricing performance, although it is not as effective as the other two innovations. Besides, compared to the w/ Biased-UCB version with the most similar performance to Pride, our algorithm is still able to reduce the cumulative regret value by 9.2 % and 40.6% on *NLQ* and *Airbnb*, respectively.

VIII. Conclusion

In this paper, we propose an effective online data pricing mechanism Pride of dynamically posting query prices with the time-discounting data value to maximize the cumulative revenue. We leverage an efficient ellipsoid method to derive the feature-based query price via exploration and exploitation. We present a *novel* non-stationary MAB algorithm Biased-TS with a regret-bound guarantee to strategically capture the time-discounting data value trend. We theoretically analyze the regret upper bound of Pride. Extensive experiments using both synthetic and real datasets demonstrate the superior performance of Pride, compared to the state-of-the-art approaches.

Acknowledgments. This work is supported by NSFC (Grants No. 62372404, No. 61825205), Fundamental Research Funds for Central Universities (Grant No. 2021FZZX001-25), and Zhejiang Provincial NSF (Grant No. LR21F020005). Xiaoye Miao is the corresponding author of the work.

References

[1] S. A. Azcoitia, C. Iordanou, and N. Laoutaris, "Understanding the price of data in commercial data marketplaces," in *ICDE*, 2023, pp. 3718–3728.

[2] GBDEx. (2023) Guiyang global big data exchange. [Online]. Available: https://www.gzdex.com.cn/

[3] SZDEx. (2023) Shen zhen data exchange. [Online]. Available: https://www.szdex.com/

[4] Xignite. (2023) Xignite. [Online]. Available: https://www.xignite.com/

[5] Datacoup. (2023) Datacoup. [Online]. Available: https://www.datacoup.com/

[6] P. Koutris, P. Upadhyaya, M. Balazinska, B. Howe, and D. Suciu, "Query-based data pricing," *J. ACM*, vol. 62, no. 5, pp. 1–44, 2015.

[7] S. Deep, P. Koutris, and Y. Bidasaria, "QIRANA demonstration: Real time scalable query pricing," *Proceedings of the VLDB Endowment*, vol. 10, no. 12, pp. 1949–1952, 2017.

[8] X. Miao, Y. Gao, L. Chen, H. Peng, J. Yin, and Q. Li, "Towards query pricing on incomplete data," *IEEE Trans. Knowl. and Data Eng.*, vol. 34, no. 8, pp. 4024–4036, 2022.

[9] X. Miao, H. Peng, X. Huang, L. Chen, Y. Gao, and J. Yin, "Modern data pricing models: Taxonomy and comprehensive survey," *arXiv preprint arXiv:2306.04945*, 2023.

[10] Airbnb. (2018) Airbnb listings in major us cities. [Online]. Available: https://www.kaggle.com/datasets/rudymizrahi/airbnb-listings-in-major-us-cities-deloitte-ml

[11] E. Valavi, J. Hestness, N. Ardalani, and M. Iansiti, "Time and the value of data," *arXiv preprint arXiv:2203.09118*, 2022.

[12] H. Peng, X. Miao, L. Chen, Y. Gao, and J. Yi, "Pricing prediction services for profit maximization with incomplete information," in *ICDE*, 2023, pp. 1353–1365.

[13] M. Zhang, A. Arafa, J. Huang, and H. V. Poor, "Pricing fresh data," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1211–1225, 2021.

[14] C. Niu, Z. Zheng, F. Wu, S. Tang, and G. Chen, "Online pricing with reserve price constraint for personal data markets," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 4, pp. 1928–1943, 2020.

[15] Z. Zheng, Y. Peng, F. Wu, S. Tang, and G. Chen, "An online pricing mechanism for mobile crowdsensing data markets," in *Mobihoc*, 2017, pp. 1–10.

[16] S. Xue, H. Ding, L. Zhang, H. Tan, and X.-Y. Li, "Online competitive posted-pricing mechanism for trading time-sensitive valued data," in *BigCom*, 2022, pp. 44–53.

[17] S. Chawla, S. Deep, P. Koutris, and Y. Teng, "Revenue maximization for query pricing," *Proceedings of the VLDB Endowment*, vol. 13, no. 1, pp. 1–14, 2019.

[18] C. Li and G. Miklau, "Pricing aggregate queries in a data marketplace." in *WebDB*, 2012, pp. 19–24.

[19] B.-R. Lin and D. Kifer, "On arbitrage-free pricing for general data queries," *Proceedings of the VLDB Endowment*, vol. 7, no. 9, pp. 757–768, 2014.

[20] C. Li, D. Y. Li, G. Miklau, and D. Suciu, "A theory of pricing private data," *ACM Trans. on Database Syst.*, vol. 39, no. 4, pp. 1–28, 2014.

[21] H. Cai, F. Ye, Y. Yang, Y. Zhu, J. Li, and F. Xiao, "Online pricing and trading of private data in correlated queries," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 569–585, 2021.

[22] C. Niu, Z. Zheng, S. Tang, X. Gao, and F. Wu, "Making big money from small sensors: Trading time-series data under pufferfish privacy," in *INFOCOM*, 2019, pp. 568–576.

[23] C. Niu, Z. Zheng, F. Wu, S. Tang, X. Gao, and G. Chen, "Unlocking the value of privacy: Trading aggregate statistics over private correlated data," in *SIGKDD*, 2018, pp. 2031–2040.

[24] C. Chen, Y. Yuan, Z. Went, G. Wang, and A. Li, "GQP: A framework for scalable and effective graph query-based pricing," in *ICDE*, 2022, pp. 1573–1585.

[25] H. Hou, L. Qiao, Y. Yuan, C. Chen, and G. Wang, "A scalable query pricing framework for incomplete graph data," in *DASFAA*, 2023, pp. 97–113.

[26] A. Xu, Z. Zheng, F. Wu, and G. Chen, "Online data valuation and pricing for machine learning tasks in mobile health," in *INFOCOM*, 2022, pp. 850–859.

[27] Z. Zheng, S. Yang, J. Xie, F. Wu, X. Gao, and G. Chen, "On designing strategy-proof budget feasible online mechanisms for mobile crowdsensing with time-discounting values," *IEEE Trans. Mob. Comput.*, vol. 21, no. 6, pp. 2088–2102, 2020.

[28] L. Xu, C. Jiang, Y. Qian, Y. Zhao, J. Li, and Y. Ren, "Dynamic privacy pricing: A multi-armed bandit approach with time-variant rewards," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 2, pp. 271–285, 2016.

[29] Y. Zhan, Y. Xia, Y. Liu, F. Li, and Y. Wang, "Incentive-aware time-sensitive data collection in mobile opportunistic crowdsensing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 9, pp. 7849–7861, 2017.

[30] Y. Jiao, P. Wang, S. Feng, and D. Niyato, "Profit maximization mechanism and data management for data analytics services," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2001–2014, 2018.

[31] R. C. Fernandez, "Protecting data markets from strategic buyers," in *SIGMOD*, 2023, pp. 1755–1769.

[32] G. Romano, G. Tartaglia, A. Marchesi, and N. Gatti, "Online posted pricing with unknown time-discounted valuations," in *AAAI*, vol. 35, no. 6, 2021, pp. 5682–5689.

[33] F. Wu, J. Liu, Z. Zheng, and G. Chen, "A strategy-proof online auction with time discounting values," in *AAAI*, vol. 28, no. 1, 2014, pp. 812–818.

[34] S. Li, L. Zhang, and X.-Y. Li, "Online pricing with limited supply and time-sensitive valuations," in *INFOCOM*, 2022, pp. 860–869.

[35] W. Mao, Z. Zheng, F. Wu, and G. Chen, "Online pricing for revenue maximization with unknown time discounting valuations." in *IJCAI*, 2018, pp. 440–446.

[36] T. Roughgarden, *Twenty lectures on algorithmic game theory*. Cambridge University Press, 2016.

[37] Y. Gao, X. Wei, X. Jing, Y. Shi, X. Gao, and G. Chen, "Online shipping container pricing strategy achieving vanishing regret with limited inventory," in *ICDE*, 2023, pp. 1719–1731.

[38] M. Feldman, A. Fiat, and A. Roytman, "Makespan minimization via posted prices," in *EC*, 2017, pp. 405–422.

[39] S. Chawla, J. D. Hartline, D. L. Malec, and B. Sivan, "Multi-parameter mechanism design and sequential posted pricing," in *STOC*, 2010, pp. 311–320.

[40] Y. Emek, R. Lavi, R. Niazadeh, and Y. Shi, "Stateful posted pricing with vanishing regret via dynamic deterministic markov decision processes," *NeurIPS*, vol. 33, pp. 2970–2982, 2020.

[41] M. C. Cohen, I. Lobel, and R. Paes Leme, "Feature-based dynamic pricing," *Manage. Sci.*, vol. 66, no. 11, pp. 4921–4943, 2020.

[42] P. Ye, J. Qian, J. Chen, C.-h. Wu, Y. Zhou, S. De Mars, F. Yang, and L. Zhang, "Customized regression model for airbnb dynamic pricing," in *SIGKDD*, 2018, pp. 932–940.

[43] S. Malpezzi *et al.*, "Hedonic pricing models: A selective and applied review," *Housing Economics and Public Policy*, vol. 1, pp. 67–89, 2003.

[44] J. Needham, *Disruptive possibilities: How big data changes everything*. O'Reilly Media, Inc., 2013.

[45] S. Xue and X.-Y. Li, "Competitive online truthful time-sensitive-valued data auction," *arXiv preprint arXiv:2210.10945*, 2022.

[46] K. Amin, A. Rostamizadeh, and U. Syed, "Repeated contextual auctions with strategic buyers," in *NIPS*, vol. 27, 2014, pp. 622–630.

[47] A. Javanmard, "Perishability of data: dynamic pricing under varying-coefficient models," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 1714–1744, 2017.

[48] A. Javanmard and H. Nazerzadeh, "Dynamic pricing in high-dimensions," *J. Mach. Learn. Res.*, vol. 20, no. 1, pp. 315–363, 2019.

[49] L. G. Khachiyan, "A polynomial algorithm in linear programming," *Dokl. Akad. Nauk SSSR*, vol. 244, pp. 1093–1096, 1979.

[50] M. Grötschel, L. Lovász, A. Schrijver, M. Grötschel, L. Lovász, and A. Schrijver, "The ellipsoid method," in *Geometric Algorithms and Combinatorial Optimization*. Springer, 1993, pp. 64–101.

[51] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3-4, pp. 285–294, 1933.

[52] O. Chapelle and L. Li, "An empirical evaluation of thompson sampling," in *NIPS*, vol. 24, 2011, pp. 2249–2257.

[53] S. Agrawal and N. Goyal, "Near-optimal regret bounds for thompson sampling," *J. ACM*, vol. 64, no. 5, pp. 1–24, 2017.

[54] D. Russo and B. Van Roy, "An information-theoretic analysis of thompson sampling," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2442–2471, 2016.

[55] F. Trovo, S. Paladino, M. Restelli, and N. Gatti, "Sliding-window thompson sampling for non-stationary settings," *J. Artif. Intell. Res.*, vol. 68, pp. 311–364, 2020.

[56] A. Garivier and E. Moulines, "On upper-confidence bound policies for non-stationary bandit problems," *arXiv preprint arXiv:0805.3415*, 2008.

[57] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, pp. 235–256, 2002.