OSTOR: Online Scheduling Framework for Trading Continuous Queries

Jin Cheng[1, 2], Ningning Ding[3], John C.S. Lui[2], Jianwei Huang[1*]
[1] The Chinese University of Hong Kong, Shenzhen
[2] The Chinese University of Hong Kong
[3] Hong Kong University of Science and Technology (Guangzhou)

***Abstract*—Data trading significantly enhances data utility by enabling data sharing across diverse applications. Despite being crucial for real-time analytics and online machine learning, trading continuous queries with streaming data output remains largely unexplored. The inherent characteristics of trading continuous queries pose distinctive technical challenges in scheduling query execution. First, the streaming nature demands online scheduling under information uncertainty, where data utilities and execution costs vary unpredictably during query execution. Second, the intrinsic NP-hardness of the optimization problem, coupled with repeated invocation requirements, necessitates efficient algorithmic solutions to address computational complexity.**

We present *OSTOR*, the first online scheduling framework for trading continuous queries. *OSTOR* aims to maximize social welfare, defined as the difference between buyers' obtained utilities and sellers' execution costs, while achieving both theoretical guarantees and practical efficiency. To handle the information uncertainty, we present a primary-dual decomposition method that transforms the online scheduling problem into multiple one-round integer programming problems, enabling adaptive decision-making that only needs current system information. To address the computational complexity, we design an adaptive dual descent (*ADD*) algorithm that iteratively optimizes dual variables, achieving a bounded constant approximation ratio in polynomial time. We further enhance *OSTOR* through structure-aware greedy optimization strategies with provable performance guarantees. Extensive experiments demonstrate that *OSTOR* substantially improves social welfare and reduces query execution costs on both real-world and synthetic datasets, compared to existing data trading methods.

Jin Cheng is with Shenzhen Institute of Artificial Intelligence and Robotics for Society, the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen and the Department of Computer Science and Engineering, The Chinese University of Hong Kong (Email: jincheng2@link.cuhk.edu.cn). Ningning Ding is with the Data Science and Analytics Thrust, Information Hub, Hong Kong University of Science and Technology (Guangzhou) (Email: ningningding@hkust-gz.edu.cn). John C.S. Lui is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong (Email: cslui@cse.cuhk.edu.hk). Jianwei Huang is with the School of Science and Engineering, Shenzhen Institute of Artificial Intelligence and Robotics for Society, Shenzhen Key Laboratory of Crowd Intelligence Empowered Low-Carbon Energy Network, and CSIJRI Joint Research Centre on Smart Energy Storage, The Chinese University of Hong Kong, Shenzhen (Corresponding Author, Email: jianweihuang@cuhk.edu.cn).

This work is supported by the National Natural Science Foundation of China (Project 62271434), Shenzhen Key Lab of Crowd Intelligence Empowered Low-Carbon Energy Network (No. ZDSYS20220606100601002), the Shenzhen Stability Science Program 2023, the Shenzhen Institute of Artificial Intelligence and Robotics for Society, and Longgang District Shenzhen's "Ten Action Plan" for Supporting Innovation Projects (No. LGKCSDPT2024002). The work of John C.S. Lui is supported in part by the RGC GRF-14202923.

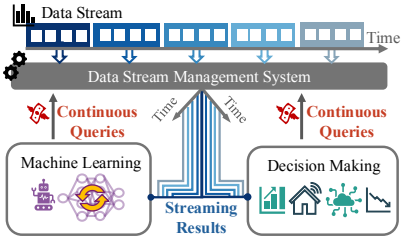


Fig. 1: Continuous queries for real-time applications.

I. Introduction

A. Background and Motivations

Data has emerged as an essential resource, playing a pivotal role in data-driven decision-making, such as machine learning [1]. The availability of diverse and extensive datasets through data trading significantly enhances the robustness of learning models by allowing more comprehensive training and validation, leading to more accurate and practical solutions. In 2022, the global data trading market attained a utility of $968 million, with a projected annual growth rate of 25% from 2023 to 2030 [2]. This burgeoning landscape has seen the emergence of numerous data trading platforms, such as AWS Data Exchange [3], Snowflake [4], and Xignite [5].

Data from continuous queries (CQs) is valuable for real-time decision-making and online machine learning [6, 7], as illustrated in Fig. 1. This is because CQs continuously process data streams and provide timely streaming results for downstream applications. Specifically, Example 1, written in Continuous Query Language (CQL), demonstrates how CQs enable real-time decision-making [8] in the stock market. This CQ continuously analyzes the `StockTrades` stream over a sliding `5 MINUTES` window, computing two key results for each stock `Symbol`: average price (`AVG(Price)`) and price volatility (`STDDEV(Price)`). When volatility exceeds `200`, the CQ outputs these results to alert stock traders and guide trading decisions. For online learning [9–13], CQs provide streaming results that support continuous model updates with evolving data patterns, thereby maintaining model accuracy and adaptability under dynamic environments.

Example 1 (A Continuous Query in Stock Market)**.**

```
SELECT Symbol, AVG(Price), STDDEV(Price)
FROM StockTrades [RANGE 5 MINUTES]
GROUP BY Symbol
HAVING STDDEV(price) > 200
```

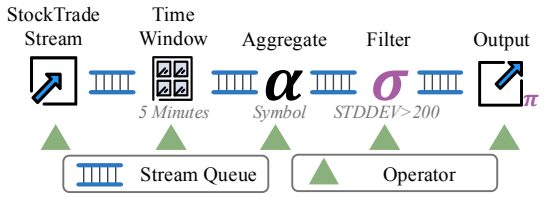


Fig. 2: A query plan for executing the CQ in Example 1.

Research on trading continuous queries remains largely unexplored despite its importance. Existing data trading studies (e.g., [14–19]) primarily focus on trading one-time queries over static database snapshots. These frameworks mainly address static properties such as privacy preservation [14–16] and truthfulness [17–19]. However, they are not suitable for trading data from continuous queries, as the streaming nature introduces distinctive technical challenges in scheduling query execution that existing methods cannot effectively address.

Trading continuous queries poses two key technical challenges arising from its inherent characteristics: information uncertainty and computational complexity. The system schedules CQ execution by constructing query plans with processing operators connected through stream queues, as shown in Fig. 2 of Example 1. While plan construction requires balancing data utilities against execution costs, the streaming nature introduces significant information uncertainty, necessitating online scheduling as both utilities and costs vary unpredictably during query execution. Specifically, execution costs vary due to dynamic factors [20, 21]: window memory costs vary with fluctuating data rates, aggregation processing costs change with varying symbol distributions, and filtering costs adapt to changing market conditions. Moreover, data utilities vary with market conditions and user preferences [22]. This leads to our first key question:

Key Question 1. *How to adaptively schedule continuous queries in an online manner to maximize social welfare, under information uncertainty in data trading?*

The second challenge stems from the intrinsic computational complexity of scheduling continuous queries, which presents an NP-hard combinatorial optimization problem [23]. Moreover, scheduling CQs requires continuous adjustments to accommodate dynamic stream variations, leading to repeated invocations of this complex optimization problem. Existing optimization approaches [24–27] are not suitable for scheduling CQs, as they do not consider the hierarchical coupling structure detailed in Section II. The structural properties of CQs require efficient algorithms that can handle these coupling relationships while ensuring computational efficiency and adaptivity. This leads to our second key question:

Key Question 2. *How to efficiently solve the NP-hard scheduling problem for trading continuous queries, under the requirement of repeated invocations?*

B. Key Contributions

We summarize our key contributions as follows:

- *Online Scheduling Framework for Trading Continuous Queries*: To the best of our knowledge, this is the first online scheduling framework for trading continuous queries. For Key Question 1, our framework adaptively addresses the information uncertainty through a customized primal-dual approach to maximize social welfare. The approach decomposes the online scheduling problem into multiple one-round integer programming problems, which enables scheduling decisions based solely on current system information.
- *Efficient Algorithm for Addressing Computational Complexity*: For Key Question 2, we develop an efficient Adaptive Dual Descent (*ADD*) algorithm to solve the NP-hard one-round problems by exploiting the problem structure. The *ADD* algorithm achieves a polynomial time complexity of $\mathcal{O}\left(MN\log(MN)\right)$ while guaranteeing a bounded constant approximation ratio, where M and N denote the numbers of query plans and continuous queries.
- *Greedy Strategies for Enhancing Performance*: We augment the algorithm with two structure-aware greedy strategies: Dynamic Reactivation Strategy (*DRS*) and Iterative Reassignment Strategy (*IRS*). *DRS* reactivates plans analytically, while *IRS* optimizes query assignments iteratively. These strategies are theoretically proven to optimize the approximation ratio in dynamic environments.
- *Experiments for Evaluating Effectiveness*: Through comprehensive experiments on both real-world and synthetic datasets, *OSTOR* demonstrates a 79.31% improvement in social welfare and achieves a 54.04% reduction in query execution costs through its adaptive scheduling mechanism, compared with classical data trading methods.

C. Related Works

We categorize related research into two areas: query-based data trading methods and online scheduling frameworks.

1) Query-based Data Trading: Research on data trading has primarily focused on static databases, emphasizing different mechanism properties such as privacy preservation [15, 28, 29], arbitrage-freeness [30–32], truthfulness [17, 33], and security [34–36]. Chen *et al.* [31] present GSHOP, a framework that introduces a pricing method for graph statistic queries through controlled noise injection while maintaining arbitrage-free guarantees, allowing buyers to balance cost and accuracy based on their requirements. Cai *et al.* [17] propose Cheap, a novel framework for trading high-dimensional correlated private data that models attribute correlations to optimize data perturbation while ensuring fair compensation through an auction-based mechanism. However, the growing demand for real-time analytics and online machine learning in domains such as stock market monitoring [37] and IoT applications [38] necessitates continuous query-based data trading, highlighting the limitations of these static approaches.

While existing mechanisms effectively address key challenges in static data trading, they cannot handle the distinctive technical challenges arising from continuous query scheduling, specifically the online optimization challenges requiring decisions without future information and the computational complexity due to query coupling. Privacy-preservation mechanisms [15, 28, 29] focus on protecting data during trading but

assume fixed data patterns. Arbitrage-free mechanisms [30–32] ensure consistent pricing across different query combinations but are limited to one-time queries. Although truthfulness mechanisms [17, 33, 39] and security frameworks [34–36] provide important trading properties such as accurate query pricing and secure transaction processing, they are designed for static datasets with fixed query patterns.

2) Online Scheduling Framework: Existing online scheduling frameworks have been developed for various domains, including distributed machine learning [40, 41], edge and cloud computing [24–27], and transportation networks [42, 43]. Mohan *et al.* [26] propose Synergy, a resource-sensitive scheduler that infers the sensitivity of deep neural networks to different resources using optimistic profiling and allocates workloads accordingly to improve the average job completion time on multi-tenant GPU clusters. Zhou *et al.* [40] design DPS, a dynamic pricing and scheduling mechanism for distributed machine learning jobs using multi-armed bandit techniques to make online decisions. Cui *et al.* [27] propose a container scheduling algorithm for edge cluster upgrading that employs self-attention and reinforcement learning to optimize task latency. Lee *et al.* [24] propose a time-dependent pricing and scheduling algorithm TD-PnS for cloud object storage services that jointly optimize pricing, resource scheduling, and energy management to maximize service provider profit through Lyapunov optimization.

While these scheduling frameworks effectively address scheduling problems in their domains, they are unsuitable for continuous queries. This is primarily because continuous query scheduling exhibits hierarchical coupling relationships between plans and queries, as detailed in Section II. These inherent coupling structures make the scheduling problem NP-hard and require repeated optimization invocations.

3) Key Distinctions from Existing Works: Our work differs from existing data trading frameworks in several fundamental aspects. While previous works focus on static databases with one-time queries, our framework specifically addresses the challenges inherent in continuous data streams where query patterns and data characteristics evolve dynamically over time. Furthermore, existing scheduling frameworks, though effective in resource allocation, are inadequate for handling the unique hierarchical coupling relationships between execution plans and continuous queries in data trading scenarios. This intrinsic coupling nature, combined with the requirement for online optimization without future knowledge, fundamentally distinguishes our problem from traditional scheduling tasks.

To the best of our knowledge, *OSTOR* is the first online scheduling framework designed for trading continuous queries. Our optimization objectives address the distinctive characteristics of trading continuous queries (i.e., information uncertainty and computational complexity), enabling seamless integration with existing data trading frameworks [20].

The paper is organized as follows. Section II presents the system model. Section III formulates the problem and solution overview. Sections IV and V detail one-round and online scheduling algorithms. Section VI presents experimental results. Section VII provides discussion. Section VIII concludes.

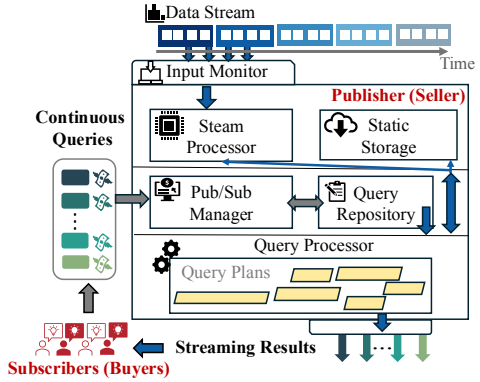


Fig. 3: Data trading system.

II. System Model

This section presents the system model. We first introduce continuous queries and query plans in Section II-A, then describe the data trading system with buyer and seller modelings in Section II-B.

A. Continuous Query and Query Plan

We introduce continuous queries and query plans based on CQL [44], a foundational stream processing language integrated into modern systems like Apache Spark Structured Streaming and Apache Flink [45]. A continuous query is a persistent query that processes streaming data based on windows (time-based or count-based) and continuously produces updated results as new data arrives. Formally, it can be expressed as $Q = (I, P, O)$, where I represents input data streams, P defines the data processing logic (e.g., selection, projection, join, and aggregation), and O specifies output specifications including update intervals and delivery methods.

Query plans provide scheduling strategies containing physical implementation details for query execution. As illustrated in Fig. 2, a query plan is a directed graph that processes one or more continuous queries, comprising three key components: operators for data processing, stream queues between operators, and synopses that maintain memory for data processing. The plan executes continuously to process incoming data streams and produce corresponding results.

For notational simplicity, we use "query" and "plan" to refer to continuous query and query plan, respectively.

B. Data Trading System

We consider a data stream management system [8] for trading continuous queries through a widely adopted publish-subscribe pattern [46, 47], as illustrated in Fig. 3. The system (i.e., the publisher or seller) comprises an Input Monitor for data stream ingestion, a Stream Processor for data processing, a Static Storage for memory management, a Subscription Manager coordinating buyers' subscriptions with the Query Repository, and a Query Processor for query execution scheduling. The trading process operates in cycles, where each cycle

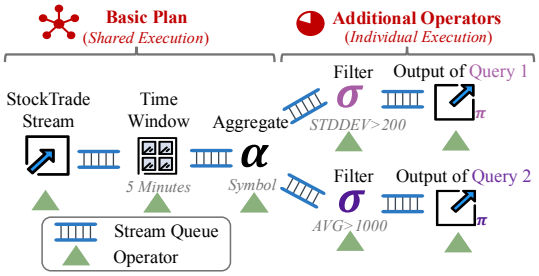


Fig. 4: Basic plan and additional operators for query execution.

consists of discrete time slots $\mathcal{T} = \{1, 2, \ldots, T\}$. At the beginning of each cycle, buyers subscribe to queries based on their analytical needs. After collecting all subscriptions, the seller constructs plans to schedule query execution and delivers real-time results to the corresponding buyers [48]. We next present the modeling of the data buyers and sellers.

*1) **Data Buyer Modeling**:* Each buyer $j \in \mathcal{N}$ subscribes to one query with a subscription budget S_j, where $\mathcal{N} = \{1, 2, \cdots, N\}$ denotes the set of all subscribed queries.

The utility[1] $u_j^{(t)}$ of query j varies over time $t \in \mathcal{T}$, reflecting the dynamic nature of data value in real-time analytics. This time-varying utility is influenced by multiple factors, including market dynamics, emergency events, and user preference shifts [22]. For instance, in Example 1, the utility of stock data increases during market volatility periods and after major economic announcements, when real-time analytical insights become crucial for decision-making.

*2) **Data Seller Modeling**:* The seller schedules query execution by constructing plans from a set of predefined basic plans $\mathcal{M} = \{1, 2, \cdots, M\}$ that handle common execution operations. Based on scheduling decisions that determine the activation of basic plans and their query assignments, each activated plan incorporates additional operators to meet its assigned queries' individual requirements. Computation results of common operators in the basic plan are shared among queries assigned to the same plan, which is a well-established query execution pattern [20, 50–52], significantly improving efficiency. We illustrate this process in the following example.

Example 2 (A Basic Plan and Additional Operators)**.** *Consider a query plan executing two queries as shown in Fig. 4, where the first query is from Example 1 and the second is:*

```
SELECT Symbol, AVG(Price), AVG(Volume)
FROM StockTrades [RANGE 5 MINUTES]
GROUP BY Symbol
HAVING AVG(Volume) > 1000.
```

The common operators from the basic plan enable execution sharing, while distinct filters require individual execution.

There are two types of execution costs in each plan: activation cost $\alpha_i^{(t)}$ for shared execution in plan i, and assignment cost[2] $\beta_{ij}^{(t)}$ for individual execution when query j is executed in plan i. Both costs vary over time $t \in \mathcal{T}$ due to dynamic execution environment factors [21]. For example, window memory costs vary with data rate fluctuations, aggregation processing costs change with symbol distribution variations, and filtering costs adapt to dynamic market conditions.

The seller determines two types of binary scheduling variables at each time slot $t \in \mathcal{T}$: plan activation variables $\boldsymbol{y}^{(t)} = \left\{ y_i^{(t)} \in \{0,1\} \mid i \in \mathcal{M} \right\}$ and query assignment variables $\boldsymbol{x}^{(t)} = \left\{ x_{ij}^{(t)} \in \{0,1\} \mid i \in \mathcal{M}, j \in \mathcal{N} \right\}$. The activation variable $y_i^{(t)}$ determines whether plan i is activated with cost $\alpha_i^{(t)}$, defining the set of activated plans as $\mathcal{M}_a^{(t)} = \{i \in \mathcal{M} \mid y_i^{(t)} = 1\}$. When $x_{ij}^{(t)} = 1$, query j is assigned to plan i with cost $\beta_{ij}^{(t)}$. These variables are coupled by $x_{ij}^{(t)} \le y_i^{(t)}, \forall i \in \mathcal{M}, j \in \mathcal{N}, t \in \mathcal{T}$, ensuring queries are only assigned to activated plans, which forms a hierarchical coupling structure.

This section has introduced the concepts of continuous queries, query plans, and the data trading system with seller and buyer modelings. Based on these concepts, we present the problem formulation and solution overview in the next section.

III. Problem Formulation and Solution Overview

In this section, we define the online and one-round scheduling problems for trading continuous queries in Sections III-A and III-B. Section III-C then provides a solution overview.

A. Online Scheduling Problem

This paper addresses the online scheduling problem of trading continuous queries, where the seller aims to maximize social welfare, defined as the total economic value generated by the trading system, representing the difference between buyers' utilities from query results and sellers' costs during query execution in a dynamic environment. We first present an offline problem $\mathbb{P}_1$, which provides "ideal" scheduling strategies by assuming complete information (i.e., $\boldsymbol{u}^{(t)}$, $\boldsymbol{\alpha}^{(t)}$, and $\boldsymbol{\beta}^{(t)}$ across time span $\mathcal{T}$).

$$\max_{\boldsymbol{x},\boldsymbol{y}} \quad \sum_{t\in\mathcal{T}} \left\{ \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{N}} x_{ij}^{(t)} (u_j^{(t)} - \beta_{ij}^{(t)}) - \sum_{i\in\mathcal{M}} y_i^{(t)} \alpha_i^{(t)} \right\} \tag{$\mathbb{P}_1$}$$

$$\text{s.t.} \quad x_{ij}^{(t)} \le y_i^{(t)}, \ \forall\, i \in \mathcal{M},\ j \in \mathcal{N},\ t \in \mathcal{T}, \tag{1a}$$

$$\sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{M}} x_{ij}^{(t)} u_j^{(t)} \le S_j, \ \forall\, j \in \mathcal{N}, \tag{1b}$$

$$\sum_{i\in\mathcal{M}} x_{ij}^{(t)} \le 1, \ \forall\, j \in \mathcal{N},\ t \in \mathcal{T}, \tag{1c}$$

$$x_{ij}^{(t)},\ y_i^{(t)} \in \{0,1\}, \quad \forall\, i \in \mathcal{M},\ j \in \mathcal{N},\ t \in \mathcal{T}. \tag{1d}$$

[1]Following [20], we use monetary units per time slot (e.g., cents/minute) as the basic measurement unit, where utilities and execution costs are calculated for each time slot. This aligns with cloud computing services [49], where resources are charged based on fixed time intervals.

[2]A query can be executed by multiple plans in various cases, such as (1) identical filters with different temporal parameters [20], and (2) different filters with identical temporal requirements [53]. When plan i cannot execute query j, we set $\beta_{ij}^{(t)} = \infty$ to prohibit such assignments.

Constraint (1a) ensures that queries are assigned only to activated plans. Constraint (1b) enforces each query's subscription budget constraint across the time span. Constraint (1c) restricts each query to be assigned to at most one plan.

By introducing dual variable vectors $\boldsymbol{\lambda}$, $\boldsymbol{w}$, and $\boldsymbol{\phi}$ for constraints (1a), (1b), and (1c), respectively, and relaxing the binary constraints (1d), we obtain the dual of Problem $\mathbb{P}_1$:

$$\min_{\boldsymbol{\lambda},\boldsymbol{w},\boldsymbol{\phi}} \quad \sum_{j\in\mathcal{N}} S_j w_j + \sum_{t\in\mathcal{T}}\sum_{j\in\mathcal{N}} \phi_j^{(t)} \tag{$\mathbb{P}_1^d$}$$

$$\text{s.t.} \quad \lambda_{ij}^{(t)} + w_j u_j^{(t)} + \phi_j^{(t)} \geq u_j^{(t)} - \beta_{ij}^{(t)}, \quad \forall\, i\in\mathcal{M},\ j\in\mathcal{N},\ t\in\mathcal{T}, \tag{2a}$$

$$\sum_{j\in\mathcal{N}} \lambda_{ij}^{(t)} \leq \alpha_i^{(t)},\ \forall\, i\in\mathcal{M},\ t\in\mathcal{T}, \tag{2b}$$

$$\boldsymbol{\lambda},\ \boldsymbol{w},\ \boldsymbol{\phi} \geq 0. \tag{2c}$$

The online scheduling problem addresses $\mathbb{P}_1$ under uncertain information (i.e., unpredictable $\boldsymbol{u}^{(t)}$, $\boldsymbol{\alpha}^{(t)}$, and $\boldsymbol{\beta}^{(t)}$ across time span $\mathcal{T}$). Since complete future information is impractical in dynamic execution environments, the seller has to make decisions at time t based on currently available system information (i.e., information up to time t).

B. One-round Scheduling Problem

To handle information uncertainty, we decompose $\mathbb{P}_1$ into one-round subproblems $\mathbb{P}_2$ by relaxing constraint (1b) and scaling $u_j^{(t)}$ to $d_j^{(t)}$ based on remaining budget S_j. We formulate the one-round problem at time $t\in\mathcal{T}$ as follows:

$$\max_{\boldsymbol{x}^{(t)},\boldsymbol{y}^{(t)}} \quad \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{N}} (d_j^{(t)} - \beta_{ij}^{(t)}) x_{ij}^{(t)} - \sum_{i\in\mathcal{M}} \alpha_i^{(t)} y_i^{(t)} \tag{$\mathbb{P}_2$}$$

$$\text{s.t.} \quad x_{ij}^{(t)} \leq y_i^{(t)},\ \forall\, i\in\mathcal{M},\ j\in\mathcal{N}, \tag{3a}$$

$$\sum_{i\in\mathcal{M}} x_{ij}^{(t)} \leq 1,\ \forall\, j\in\mathcal{N} \tag{3b}$$

$$x_{ij}^{(t)}, y_i^{(t)} \in \{0,1\},\ \forall\, i\in\mathcal{M},\ j\in\mathcal{N}. \tag{3c}$$

We derive the corresponding dual problem $\mathbb{P}_2^d$ as follows:

$$\min_{\boldsymbol{\lambda}^{(t)},\boldsymbol{\phi}^{(t)}} \quad \sum_{j\in\mathcal{N}} \phi_j^{(t)} \tag{$\mathbb{P}_2^d$}$$

$$\text{s.t.} \quad \lambda_{ij}^{(t)} + \phi_j^{(t)} \geq d_j^{(t)} - \beta_{ij}^{(t)},\ \forall\, i\in\mathcal{M},\ j\in\mathcal{N} \tag{4a}$$

$$\sum_{j\in\mathcal{N}} \lambda_{ij}^{(t)} \leq \alpha_i^{(t)},\ \forall i\in\mathcal{M} \tag{4b}$$

$$\lambda_{ij}^{(t)}, \phi_j^{(t)} \geq 0,\ \forall i\in\mathcal{M},\ j\in\mathcal{N}. \tag{4c}$$

The one-round scheduling problem $\mathbb{P}_2$ jointly optimizes binary variables $(y_i^{(t)}, x_{ij}^{(t)})$ for plan activation and query assignment to maximize social welfare. Despite its simplified form compared to $\mathbb{P}_1$, the computational complexity of $\mathbb{P}_2$ remains challenging, as shown in the following proposition.

Proposition 1. *Problem $\mathbb{P}_2$ is strongly NP-hard.*

Proof. We prove this by a polynomial-time reduction from the strongly NP-hard set cover problem [54] to the one-round problem $\mathbb{P}_2$, as detailed in our online technical report [55]. □

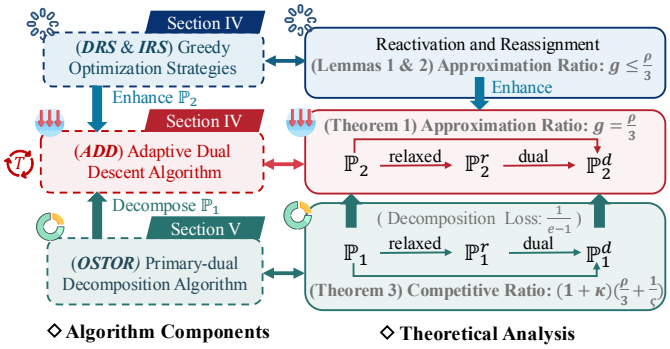


Fig. 5: Online scheduling framework.

Therefore, solving the scheduling problem faces two challenges: information uncertainty in Problem $\mathbb{P}_1$ and computational complexity of NP-hard Problem $\mathbb{P}_2$ requiring repeated invocations. In the next subsection, we present the solution overview, with detailed algorithms in Sections IV and V.

C. Solution Overview

We propose an online scheduling framework that integrates temporal decomposition and efficient approximation algorithms to address these challenges. As illustrated in Fig. 5, the framework consists of the following components:

- *Primary-dual Decomposition Algorithm for Online Scheduling*: The framework decomposes $\mathbb{P}_1$ into a sequence of one-round problems $\mathbb{P}_2$ through primal-dual algorithm *OSTOR*. Using carefully designed weights $w_j^{(t)}$, it enables efficient scheduling decisions solely based on current information. Section V presents the details.
- *Adaptive Dual Descent Algorithm for One-round Scheduling*: The framework incorporates an efficient approximation algorithm *ADD* to solve Problem $\mathbb{P}_2$ through adaptive dual descent. The dual variables $\lambda_{ij}^{(t)}$ and $\phi_j^{(t)}$ act as economic prices to guide plan activation and query assignment. Section IV-A provides the details.
- *Greedy Optimization Strategies for Enhancement*: To optimize system performance, the framework employs two greedy strategies with theoretical guarantees: Dynamic Reactivation Strategy (*DRS*) and Iterative Reassignment Strategy (*IRS*). *DRS* iteratively activates superior query plans following designed principles until convergence, while *IRS* optimizes social welfare through structured query-to-plan reassignment. Section IV-B provides detailed analysis.

This section has formulated the online and one-round scheduling problems for trading continuous queries. In Section IV, we present the algorithm design for the one-round scheduling problem, followed by Section V, where we develop the complete online scheduling framework.

IV. One-Round Scheduling Algorithm Design

In this section, we propose an efficient approximation algorithm for the one-round problem $\mathbb{P}_2$ through a series of algorithm designs. We present the basic Adaptive Dual Descent (*ADD*) algorithm in Section IV-A, providing polynomial time complexity with a constant approximation ratio.

Then Section IV-B introduces two greedy strategies to enhance performance. Due to page limit, all proofs are in our technical report [55]. Since all parameters and variables are in time slot t, we omit the superscript (t) in this section for clarity.

A. Basic Adaptive Dual Descent Algorithm

We develop an efficient approximation approach using adaptive dual descent (*ADD*) as the basic algorithm for Problem $\mathbb{P}_2$. The algorithm presentation follows three steps: key ideas in Section IV-A1, detailed algorithm design in Section IV-A2, and theoretical analysis of approximation ratio and time complexity in Section IV-A3.

*1) **Algorithm Idea**: ADD* iteratively constructs a solution by adaptively adjusting dual variables, which naturally guide both plan activation decisions and query assignment strategies. Since the core mechanism of our algorithm relies on this dual adjustment process, understanding the dual solution is fundamental: We first conduct a comprehensive analysis of the dual solution in part (a), and then demonstrate how to systematically derive a feasible primal solution while effectively addressing the problem-specific constraints in part (b).

(a) Analysis of Dual Solutions: Our analysis of dual solutions, focusing on their structure and economic interpretations, reveals the core scheduling mechanism. For any optimal dual solution $(\boldsymbol{\phi}^*, \boldsymbol{\lambda}^*)$ to Problem $\mathbb{P}_2^d$, given $\boldsymbol{\lambda}^*$, the optimal ϕ_j^* that minimizes the objective function takes the following form:

$$\phi_j^* = \max_{i \in \mathcal{M}}(0, d_j - \beta_{ij} - \lambda_{ij}^*), \tag{5}$$

since ϕ_j^* needs to satisfy constraint (4a) while minimizing the objective function.

The dual variables have corresponding economic interpretations. ϕ_j^* represents the seller's subsidy to buyer j, which the seller minimizes in $\mathbb{P}_2^d$. By complementary slackness [56], the plan is activated only when (4b) becomes tight. The cost share λ_{ij}^* decomposes α_i among queries and is charged only upon execution, eliminating explicit activation charges while ensuring cost recovery. The dual solution (5) reflects buyers' incentive to maximize their subsidies across plans.

(b) Deriving Primal Solutions from Dual Solutions: Based on the dual solution in (5), we now demonstrate the derivation of a primal solution for Problem $\mathbb{P}_2$. The primal solution requires determining two decision variables: plan activation $y_i,\ \forall\ i \in \mathcal{M}$ and query assignment $x_{ij},\ \forall\ i \in \mathcal{M},\ j \in \mathcal{N}$. We first establish two key definitions that guide our derivation:

Definition 1 (Association)**.** *For dual solution $(\boldsymbol{\phi}^*, \boldsymbol{\lambda}^*)$, query j associates plan i if $\phi_j^* \leq d_j - \beta_{ij}$. We define $N(i) = \left\{j \in \mathcal{N} : \phi_j^* \leq d_j - \beta_{ij}\right\}$ and $N(j) = \left\{i \in \mathcal{M} : \phi_j^* \leq d_j - \beta_{ij}\right\}$ as their respective associated sets.*

Definition 2 (Contribution)**.** *Given a dual solution $(\boldsymbol{\phi}^*, \boldsymbol{\lambda}^*)$, a query j contributes to a plan i if $\lambda_{ij}^* > 0$.*

To derive primal solution, we first determine query assignments based on dual solution. For any optimal dual solution $(\boldsymbol{\phi}^*, \boldsymbol{\lambda}^*)$, given $\boldsymbol{\phi}^*$, we can derive feasible cost shares $\boldsymbol{\lambda}^*$ by:

$$\lambda_{ij}^* = \max(0, d_i - \beta_{ij} - \phi_j^*). \tag{6}$$

This formulation establishes two key properties: (1) if query j contributes to plan i, then query j associates plan i (i.e., $j \in N(i)$), since $\lambda_{ij}^* > 0$ implies $\phi_j^* < d_j - \beta_{ij}$; (2) if $j \in N(i)$, then $\phi_j^* = d_j - \beta_{ij} + \lambda_{ij}^*$. These properties indicate that we can assign query j to its associated plan i (i.e., $j \in N(i)$).

Next, we determine the plan activation decisions. Let $\bar{\mathcal{M}}_a = \left\{i \in \mathcal{M} : \sum_{j \in \mathcal{N}} \lambda_{ij}^* = \alpha_i\right\}$ be the set of all plans where the sum of the cost shares equals the plan activation cost (i.e., the corresponding constraint (4b) is tight). For plan $i \in \bar{\mathcal{M}}_a$, the assignment utility of the associated set $N(i)$ minus the plan activation cost is exactly equal to the sum of the dual variables ϕ_j^* of the associating queries, which means

$$\sum_{j \in N(i)} (d_j - \beta_{ij}) - \alpha_i = \sum_{j \in N(i)} (d_j - \beta_{ij} - \lambda_{ij}^*) = \sum_{j \in N(i)} \phi_j^*. \tag{7}$$

Here, the first equality follows since $\lambda_{ij}^* > 0$ implies that $j \in N(i)$ and the second equality follows since $j \in N(i)$ implies that $d_j - \beta_{ij} = \phi_j^* + \lambda_{ij}^*$ according to (6). We also find a structural property of the dual solution, as shown in the following proposition.

Proposition 2. *In an optimal dual solution $(\boldsymbol{\phi}^*, \boldsymbol{\lambda}^*)$, each query must associate at least one plan in $\bar{\mathcal{M}}_a$.*

Proof. Please refer to our technical report [55]. □

Proposition 2 reveals that each query must have at least one associated plan where subsidies fully reflect the activation cost. Accordingly, we can establish strong duality between the relaxed primal problem $\mathbb{P}_2^r$ and its dual $\mathbb{P}_2^d$:

$$\sum_{i \in \mathcal{M}} \sum_{j \in N(i)} (d_j - \beta_{ij}) - \alpha_i = \sum_{i \in \mathcal{M}} \sum_{j \in N(i)} \phi_j^*. \tag{8}$$

which shows that the gap between relaxed problem $\mathbb{P}_2^r$ and dual problem $\mathbb{P}_2^d$ is zero and both solutions are optimal.

Therefore, a potential primary solution activates plans in $\bar{\mathcal{M}}_a$ and assigns query j to plan i if $j \in N(i)$. However, a challenge emerges: a query j may contribute to multiple plans' activation costs in $\bar{\mathcal{M}}_a$, violating constraint (3b). To address this, we select a subset $\mathcal{M}_a$ of $\bar{\mathcal{M}}_a$ and implement a plan redirection strategy that ensures each query contributes to at most one plan through a mapping function $\varphi(\cdot)$ detailed next.

*2) **Algorithm Design**:* We design an Adaptive Dual Descent (*ADD*) algorithm, as shown in Algorithm 1. The key idea is to gradually reduce query subsidies while maintaining dual feasibility, using dual solutions in (5) and (6) for plan activation and query assignments. Query subsidies are fixed upon triggering events. The algorithm consists of two phases:

Phase 1: Initialization of Dynamic Sets (Line 1): The algorithm maintains three dynamic sets: activated plans $\mathcal{M}_a$, unprocessed plans $\mathcal{M}_u$, and unassigned queries $\mathcal{N}_u$. For multi-plan query contributions, we define redirection function $\varphi(\cdot)$ that maps execution plans to their target plans (Lines 10-11).

Phase 2: Adaptive Dual Descent Process (Lines 2-15): The algorithm operates through adaptive dual descent. Query subsidies ϕ_j decrease iteratively, with plan sharing costs $\lambda_{ij} = \max(0, d_j - \beta_{ij} - \phi_j)$ following (6). Two events trigger plan activation and query assignment:

Algorithm 1: Basic Adaptive Dual Descent Algorithm.

Input: $\mathcal{M}$, $\mathcal{N}$, $\boldsymbol{d}$, $\boldsymbol{\beta}$, $\boldsymbol{\alpha}$
Output: $\boldsymbol{y}$, $\boldsymbol{x}$

1 $\mathcal{M}_u := \mathcal{M}$; $\mathcal{N}_u := \mathcal{N}$; $\mathcal{M}_a := \emptyset$;
2 **while** $\mathcal{N}_u \neq \emptyset$ **do**
3 $\quad e_q = \max\{d_j - \beta_{ij} : i \in \mathcal{M} \setminus \mathcal{M}_u, j \in \mathcal{N}_u\}$;
4 $\quad e_p = \max\{\tau : \exists\, i \in \mathcal{M}_u : \sum_{j\in\mathcal{N}_u} \max\{0, d_j - \beta_{ij} - \tau\} + \sum_{j\in\mathcal{N}\setminus\mathcal{N}_u} \max\{0, d_j - \beta_{ij} - \phi_j\} = \alpha_i\}$;
5 $\quad e = \max\{e_q, e_p\}$;
$\quad$ // **Query Assignment Events Occur**
6 $\quad$ **for** $i \in \mathcal{M} \setminus \mathcal{M}_u$ *and* $j \in \mathcal{N}_u$ *with* $d_j - \beta_{ij} = e$ **do**
7 $\quad\quad x_{\varphi(i),j} = 1$; $\phi_j = e$; $\mathcal{N}_u = \mathcal{N}_u \setminus \{j\}$;
$\quad$ // **Plan Activation Events Occur**
8 $\quad$ **for** $i \in \mathcal{M}_u$ *with* $\sum_{j\in\mathcal{N}_u} \max\{0, d_j - \beta_{ij} - e\} + \sum_{j\in\mathcal{N}\setminus\mathcal{N}_u} \max\{0, d_j - \beta_{ij} - \phi_j\} = \alpha_i$ **do**
9 $\quad\quad \mathcal{M}_u = \mathcal{M}_u \setminus \{i\}$;
10 $\quad\quad$ **if** $\exists\, i' \in \mathcal{M}_a$ *and* $j \in \mathcal{N} \setminus \mathcal{N}_u$ *with* $\phi_j < d_j - \beta_{i'j}$ *and* $\phi_j < d_j - \beta_{ij}$ **then**
11 $\quad\quad\quad \varphi(i) = i'$;
12 $\quad\quad$ **else**
13 $\quad\quad\quad \varphi(i) = i$; $y_i = 1$; $\mathcal{M}_a = \mathcal{M}_a \cup \{i\}$;
14 $\quad\quad$ **for** $j \in \mathcal{N}_u$ *with* $d_j - \beta_{ij} \geq e$ **do**
15 $\quad\quad\quad x_{\varphi(i),j} = 1$; $\phi_j = e$; $\mathcal{N}_u = \mathcal{N}_u \setminus \{j\}$;

- *Query Assignment Events (e_q)* given in Line 3 occur when the sharing cost of an unassigned query to an activated plan becomes zero, indicating $\phi_j = d_j - \beta_{ij}$. The algorithm then assigns query j to plan i by setting $x_{\varphi(i),j} = 1$, fixes ϕ_j at the current value, and updates the set $\mathcal{N}_u$ (Lines 6-7).
- *Plan Activation Events (e_p)* defined in Line 4 occur when the accumulated sharing costs of plan i reach its activation threshold $\sum_{j\in\mathcal{N}} \lambda_{ij} = \alpha_i$ (Lines 8-9). The algorithm either redirects plan i to an active plan i' or activates it independently. Redirection occurs when a query j contributes positively to both plans ($\lambda_{ij} > 0$ and $\lambda_{i'j} > 0$), and sets $\varphi(i)$ to i' (Lines 10-11). Otherwise, plan i is directly activated (Lines 12-13). The algorithm then assigns all unassigned queries $j \in \mathcal{N}$ with $d_j - \beta_{ij} \geq e$ to plan i and freezes their ϕ_j values (Lines 14-15).

Concurrent events are processed in arbitrary order. The algorithm continues until all queries are handled.[3] Based on the algorithm design, we next analyze its theoretical properties.

*3) **Theoretical Analysis**:* We establish the theoretical guarantees for the *ADD* algorithm in this subsection, analyzing both its approximation ratio and computational complexity. Let $b_{ij} = d_j - \beta_{ij}$ be the assignment utility of executing query j in plan i. We define

$$\rho = \max_{i,i'\in\mathcal{M},\ j,j'\in\mathcal{N}} \frac{(b_{ij} + b_{i'j} + b_{i'j'})}{b_{ij'}} \tag{9}$$

[3]To ensure completeness, there is a null plan n with $\alpha_n = 0$ and $d_j - \beta_{nj} = 0$ to handle any unassigned queries.

as the homogeneity ratio of query utilities, measuring the maximum ratio between assignment utilities across any pair of plans and any pair of queries. This ratio is a bounded constant in practice, as the assignment utilities typically fall within a bounded range. Our main theoretical result is as follows:

Theorem 1. *Algorithm 1 computes feasible primary and dual solutions for Problems $\mathbb{P}_2$ and $\mathbb{P}_2^d$, and guarantees $p\rho/3 \geq d$, where p and d denote the primal and dual objective values.*

Proof. Please refer to our technical report [55]. □

For any maximization integer program $\mathbb{P}$, the approximation ratio measures the ratio between the optimal solution p^* and the algorithm's solution p, bounded by d/p where d denotes the dual solution. A smaller ratio indicates better approximation quality, with 1 being the ideal case where the algorithm achieves optimality. Following Theorem 1, we have:

Remark 1. *The basic ADD algorithm achieves a bounded constant approximation ratio of $\rho/3$.*

This theoretical guarantee yields several key implications. The approximation ratio $\rho/3$ characterizes the system utility structure. In homogeneous utility settings ($b_{ij} \approx b$, $\forall\ i \in \mathcal{M},\ j \in \mathcal{N}$), ρ converges to three, and the approximation ratio approaches one, as redirection yields approximate optimal value. For heterogeneous utility, empirical evaluation in Section VI demonstrates *ADD*'s effectiveness through adaptive query-plan assignment optimization. We further establish the computational complexity through the following theorem:

Theorem 2. *The time complexity of Algorithm 1 is $\mathcal{O}(MN\log(MN))$.*

Proof. Please refer to our technical report [55]. □

Theorem 2 shows that the *ADD* algorithm maintains polynomial-time complexity while providing a bounded constant approximation guarantee. This polynomial-time complexity is crucial for our problem requiring repeated invocations, as it ensures the scalability with problem size.

To further enhance the performance of the ADD algorithm, we propose two greedy optimization strategies that leverage the problem structure in the following subsection.

B. Greedy Optimization Strategies

Building upon the *ADD* algorithm, we propose two greedy strategies to enhance the solution quality: the dynamic reassignment strategy (DRS) in Section IV-B1, and the iterative reactivation strategy (IRS) in Section IV-B2.

*1) **Dynamic Reassignment Strategy (DRS)**:* We augment the *ADD* algorithm with dynamic query reassignment (*DRS*) for plan activation. When evaluating plan i's activation, we consider both unassigned queries and potential reassignments that yield higher utility. Both assignment types contribute to amortizing activation costs. The *DRS* redefines e_p as τ that satisfies $\exists\ i \in \mathcal{M} \setminus \mathcal{M}_a$:

$$\sum_{j\in\mathcal{N}} \max(0, d_j - \beta_{ij} - \tau) + \sum_{j\in\mathcal{N}\setminus\mathcal{N}_u} \max(0, \beta_{\sigma(j),j} - \beta_{ij}) = \alpha_i \tag{10}$$

For an already assigned query $j \in \mathcal{N} \setminus \mathcal{N}_u$, its contribution to the activation of plan i is $\max(0, \beta_{\sigma(j),j} - \beta_{ij})$, where $\sigma(j)$ is j's assigned plan. In (10), the first term captures unassigned query benefits, and the second term reflects reassignment gains. These gains amortize plan i's activation cost a_i.

For the theoretical bound, we conduct a factor-revealing analysis detailed in our online report [55]:

Lemma 1. *For any set of activated plans $\mathcal{M}_x \subseteq \mathcal{M}$, Algorithm 1 augmented with DRS achieves a social welfare of at least $\delta_P p(\mathcal{M}_x) - \delta_Q c(\mathcal{M}_x)$ for Problem $\mathbb{P}_2$, with an approximation ratio $\theta = \min \frac{\delta_P p(\mathcal{M}_x) - \delta_Q c(\mathcal{M}_x)}{p(\mathcal{M}_x) - c(\mathcal{M}_x)}$, where $c : 2^{\mathcal{M}} \to R$ denote the activation cost function, $p : 2^{\mathcal{M}} \to R$ denote the assignment utility function, $\delta_Q \in [0, 1]$ and δ_P is the optimal value of Problem $\mathbb{P}_3$ in [55].*

This lemma proves *DRS* strategy maintains a provable performance guarantee. After obtaining the solution from the *DRS*-augmented *ADD* algorithm, we propose a greedy strategy to iteratively optimize the activated plan set.

*2) **Iterative Reactivation Strategy (IRS)**:* We propose an iterative reactivation strategy (*IRS*) that operates on the activated plan set $\mathcal{M}_a$ determined by the *DRS*-augmented *ADD* algorithm. *IRS* activates additional plans based on their efficiency ratio $(p(\mathcal{M}_a \cup i) - p(\mathcal{M}_a))/\alpha_i$. The optimality of this greedy strategy is guaranteed by the following lemma:

Lemma 2. *Given the optimal activated plan set $\mathcal{M}_a^*$, for any activated plan set $\mathcal{M}_a$, there exists a plan $i \in \mathcal{M}_a^*$ such that*

$$\frac{p(\mathcal{M}_a \cup \{i\}) - p(\mathcal{M}_a)}{\alpha_i} \geq \frac{p(\mathcal{M}_a^*) - p(\mathcal{M}_a)}{c(\mathcal{M}_a^*)}. \tag{11}$$

Proof. Please refer to our technical report [55]. □

Lemma 2 shows that at least one plan from the optimal set $\mathcal{M}_a^*$ has a marginal utility-to-cost ratio no worse than the average ratio of the optimal set $\mathcal{M}_a^*$. This property is fundamental to our greedy algorithm's guarantee.

Based on Lemma 2, *IRS* iteratively activates plans to optimize Problem $\mathbb{P}_2$. In each iteration, we select plan $i \in \mathcal{M}$ maximizing ratio $(p(\mathcal{M}_a \cup \{i\}) - p(\mathcal{M}_a))/\alpha_i$ and add to $\mathcal{M}_a$. Process terminates when $p(\mathcal{M}_a \cup \{i\}) - p(\mathcal{M}_a) \leq \alpha_i,\ \forall i \in \mathcal{M}$. After each activation, queries are assigned to optimal plans where $\sigma^*(j) = \text{argmax}_{i \in \mathcal{M}_a}(d_j - \beta_{ij}),\ \forall j \in \mathcal{N}$.

By incorporating *DRS* and *IRS* strategies, based on Lemmas 1 and 2, we can establish stronger theoretical guarantees for our solution to $\mathbb{P}_2$ as follows:

Remark 2. *The ADD algorithm, augmented with DRS and IRS strategies, achieves an improved approximation ratio of $\min(\rho/3, \theta)$ for Problem $\mathbb{P}_2$.*

In this section, we have presented an efficient algorithm for the one-round scheduling problem $\mathbb{P}_2$, based on *ADD* and two greedy strategies. This design serves as the foundation for the online scheduling solution presented in the next section.

V. An Online Scheduling Framework

In this section, we solve the online scheduling problem (i.e., Problem $\mathbb{P}_1$ under information uncertainty). Section V-A presents key ideas, Section V-B details algorithm design, and Section V-C provides theoretical analysis.

Algorithm 2: Online Scheduling Framework *OSTOR*.

Input: $\mathcal{M}$, $\mathcal{N}$, Time-varying information
Output: $\boldsymbol{y}$, $\boldsymbol{x}$
Set $\boldsymbol{w}^{(0)} = \mathbf{0}$;
for $t \in \mathcal{T}$ **do**
  Input: $\boldsymbol{u}^{(t)}$, $\boldsymbol{\beta}^{(t)}$, $\boldsymbol{\alpha}^{(t)}$
  for $j \in \mathcal{N}$ **do**
    if $u_j^{(t-1)} < 1$ **then**
      $d_j^{(t)} = u_j^{(t)}(1 - w_j^{(t-1)})$;
    else
      $d_j^{(t)} = 0$;
  $\{\boldsymbol{y}^{(t)}, \boldsymbol{x}^{(t)}\} = \mathcal{A}_{ADD}^{G}(\mathcal{M}, \mathcal{N}, \boldsymbol{d}^{(t)}, \boldsymbol{\beta}^{(t)}, \boldsymbol{\alpha}^{(t)})$;
  for $j \in \mathcal{N}$ *and* $\sum_{i \in \mathcal{M}} x_{ij}^{(t)} = 1$ **do**
    Calculate $F_j^{(t)}$ according to (13);
    $S_j^{(t)} = S_j^{(t-1)} - F_j^{(t)}$;
    $w_j^{(t)} = w_j^{(t-1)} \left(1 + \frac{u_j^{(t)} - F_j^{(t)}}{S_j^{(t)}}\right) + \frac{u_j^{(t)} - F_j^{(t)}}{\zeta S_j^{(t)}}$;
  Output: $\boldsymbol{y}^{(t)}$, $\boldsymbol{x}^{(t)}$

A. Algorithm Idea

The key challenge in solving Problem $\mathbb{P}_1$ under information uncertainty lies in preserving the long-term budget constraints (1b). In the ideal case, these constraints require each buyer's budget to support participation across all T rounds for optimal social welfare. However, in the online setting, premature budget exhaustion would prevent future participation, leading to suboptimal outcomes.

To address this challenge, we take a two-fold approach. First, we develop a theoretically guaranteed primal-dual approach to decompose scheduling problem $\mathbb{P}_1$ into one-round scheduling problem $\mathbb{P}_2$, which can be efficiently solved by the algorithm presented in Section IV using only current system information. Second, similar to [25, 57], we design a dynamic utility scaling mechanism to prevent rapid budget depletion. For each buyer j, we introduce a scaling factor $w_j^{(t)}$ tracking cumulative budget utilization, which evolves from 0 to 1, with $w_j^{(t)} = 1$ indicating complete budget exhaustion. The original utility $u_j^{(t)}$ is then scaled to:

$$d_j^{(t)} = u_j^{(t)}(1 - w_j^{(t)}). \tag{12}$$

This scaling naturally reduces the scheduling priority of queries with depleting budgets through the factor $(1 - w_j^{(t)})$, thereby extending budget lifetimes across the entire period.

Finally, by dynamically scaling utilities based on budget usage, we decompose the multi-round online scheduling problem into multiple one-round problems $\mathbb{P}_2$. Next, we present our online scheduling framework in the following subsection.

B. Algorithm Design

In this subsection, we present *OSTOR*, an online scheduling framework for trading continuous queries, which leverages primal-dual decomposition to address information uncertainty. The core idea of *OSTOR* is to transform the online scheduling problem into a sequence of one-round problems through three phases: utility scaling, scheduling decision, and scaling factor update, as detailed in Algorithm 2.

OSTOR initializes scaling factors $w_j^{(0)} = 0$ for all queries (Line 1). In each round $t \in \mathcal{T}$, it schedules based on current system state ($\boldsymbol{u}^{(t)}$, $\boldsymbol{\beta}^{(t)}$, $\boldsymbol{\alpha}^{(t)}$) through three phases (Line 2):

Phase 1: Utility Scaling (Lines 3-7) computes scaled utility $d_j^{(t)}$ for each query j using factor $w_j^{(t-1)}$. This scaling excludes queries with depleted budgets ($w_j^{(t-1)} = 1$) and weights others by the remaining budget.

Phase 2: Scheduling Decision (Line 8) solves one-round scheduling by invoking $\mathcal{A}_{ADD}^{G}$ (ADD with two greedy strategies) to determine $(\boldsymbol{y}^{(t)}, \boldsymbol{x}^{(t)})$. This phase optimizes decisions using scaled utilities from Phase 1.

Phase 3: Scaling Factor Update (Lines 9-12) updates the scaling factor $w_j^{(t)}$ based on scheduling decisions. First, it calculates each query's execution cost as follows:

$$F_j^{(t)}(\boldsymbol{y}^{(t)}, \boldsymbol{x}^{(t)}) = \sum_{i \in \mathcal{M}} x_{ij}^{(t)} \beta_{ij}^{(t)} + \sum_{i \in \mathcal{M}: x_{ij}^{(t)} = 1} \frac{\alpha_i^{(t)}}{\sum_{j \in \mathcal{N}} x_{ij}^{(t)}}, \tag{13}$$

comprising assignment cost (first term) and shared activation cost (second term). Then $w_j^{(t)}$ is updated in Line 12, where $\zeta = (1+\Gamma)^{\frac{1}{\Gamma}} - 1$ and $\Gamma = \max_{j \in \mathcal{N}, t \in \mathcal{T}} \left((u_j^{(t)} - F_j^{(t)}(\boldsymbol{y}^{(t)}, \boldsymbol{x}^{(t)}))/S_j^{(t)} \right)$ is the maximum utility-to-budget ratio. This update maintains budget constraint (1b) across T rounds, as proved in Theorem 3.

The following subsection establishes the theoretical properties and performance guarantees of the proposed algorithm.

C. Theoretical Analysis

We now present the theoretical analysis of Algorithm 2, focusing on two key aspects: competitive ratio and computational complexity. For an online algorithm, the competitive ratio measures the worst-case ratio between the optimal offline solution and the algorithm's solution. We establish the competitive ratio through a sequence of lemmas that progressively analyze the algorithm's properties, similar to [25, 58]. First, we show that our algorithm generates feasible dual solutions:

Lemma 3. *Algorithm 2 produces a feasible solution for Problem $\mathbb{P}_1^d$.*

Proof. Please refer to our technical report [55]. □

To obtain the competitive ratio, we need to analyze the gap between primal and dual objectives. Let $P^{(t)}$ and $D^{(t)}$ denote the objective values of Problems $\mathbb{P}_1$ and $\mathbb{P}_1^d$ after t iterations, respectively. The following lemma establishes a crucial relationship between the incremental changes:

Lemma 4. *Algorithm 2 ensures: $(\rho/3+1/\zeta)\Delta P(t) \geq \Delta D(t)$, where $\Delta P(t) = P^{(t)} - P^{(t-1)}$ and $\Delta D(t) = D^{(t)} - D^{(t-1)}$ represent the incremental changes in round t.*

Proof. Please refer to our technical report [55]. □

While these lemmas establish the solution feasibility and a bounded primal-dual gap, we also need to analyze the budget constraint satisfaction. Let $u_j'^{(t)} = u_j^{(t)} - F_j^{(t)}(\boldsymbol{y}^{(t)}, \boldsymbol{x}^{(t)})$ denote the marginal utility and $\kappa = \max_{j \in \mathcal{N}, t \in \mathcal{T}} \left((u_j^{(t+1)} - F_j^{(t+1)}(\boldsymbol{x}^{(t+1)}, \boldsymbol{y}^{(t+1)}))/S_j^{(t)} \right)$ denote the marginal next utility-to-budget ratio. We have:

Lemma 5. *Algorithm 2 ensures a relaxed budget constraint:*

$$\sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{M}} u_j'^{(t)} x_{i,j}^{(t)} \leq (1+\kappa) S_j^{(t)} \ \forall j \in \mathcal{N}. \tag{14}$$

Proof. Please refer to our technical report [55]. □

Combining these lemmas, we prove that *OSTOR* achieves a constant competitive ratio while satisfying all constraints:

Theorem 3. *Algorithm 2 achieves a competitive ratio of $(1+\kappa)(\rho/3+1/\zeta)$ for the online scheduling problem $\mathbb{P}_1$.*

Proof. Please refer to our technical report [55]. □

As κ and Γ approach 0 (indicating high utility-to-budget efficiency), the competitive ratio approaches its optimal value of $\rho/3+1/(e-1)$ with $\zeta = e-1$. This optimal ratio exceeds the one-round problem's primal-dual gap by only $1/(e-1)$, demonstrating the effectiveness of our online framework, as shown by our experimental results. Moreover, *OSTOR* maintains efficiency with the following complexity:

Theorem 4. *The time complexity of Algorithm 2 is $\mathcal{O}(TMN\log(MN))$.*

Proof. Please refer to our technical report [55]. □

Theorem 4 shows that Algorithm 2 achieves polynomial time complexity where T is the number of time slots, M is the number of plans, and N represents the number of queries.

In this section, we have introduced the algorithm design for the online scheduling problem $\mathbb{P}_1$, including the algorithm ideas and theoretical analysis. In the following section, we will evaluate the framework performance through experiments.

VI. Experimental Evaluation

In this section, we conduct comprehensive experiments on both real-world and synthetic datasets to evaluate *OSTOR*'s performance. Section VI-A describes the experimental settings, and Section VI-B presents a series of experimental result analyses and corresponding insights.

A. Experimental Settings

1) Datasets: We evaluate OSTOR on four[4] real-world and two synthetic datasets, each partitioned into 200 time slots with user preferences η_j distributed uniformly in $[20, 40]$. For each

[4]We also tested on WEATHER [59] and POWER [60] datasets. Due to page limitations, the description, experimental results, and analysis of these two datasets are provided in our online technical report.[55].

dataset, we compute time-varying utilities as $u_j^{(t)} = v_b^{(t)} \times \eta_j$, where $v_b^{(t)}$ is the dataset-specific base query utility:

- TRAFFIC [61]: Collects iCloud traffic records from 20 countries with frame ID, time, and length attributes. The base query utility $v_b^{(t)}$ is calculated as the product of data arrival rate $\lambda^{(t)}$ and average frame length $L^{(t)}$, reflecting network traffic patterns.
- IOT [62]: Contains IoT sensor data with packet ID, time, and bytes information. The base query utility $v_b^{(t)}$ is computed as the product of data arrival rate $\lambda^{(t)}$ and average bytes $b^{(t)}$, representing network load characteristics.
- FINANCIAL [63]: Captures 10-year historical data of US stocks and ETFs. The base query utility $v_b^{(t)}$ is calculated as the product of the closing price $p^{(t)}$ and trading volume $v^{(t)}$, capturing market activity dynamics.
- SOCIAL [64]: Comprises social media posts with 15 features spanning user interactions, content characteristics, and temporal-spatial dimensions. The base query utility $v_b^{(t)}$ is derived from the product of retweet count $r^{(t)}$ and like count $l^{(t)}$, reflecting content engagement levels.
- UNIFORM and NORMAL: Generate base query utilities and user preferences that follow uniform and normal distributions, respectively, enabling controlled evaluation under diverse scenarios.

*2) **Workload**:* Following [20, 65], we implement a synthetic aggregate query generator for evaluation. We consider plan computation sharing types discussed in Section II where queries share filters but differ in temporal requirements. Each aggregate query is characterized by range r (window size) and slide s (sliding interval). Our workload uses 100 queries and ten execution plans. The slide s follows uniform distribution over $[1, 1024]$, while range r equals $s \times \gamma$, with overlap factor γ uniform over $[1, 100]$. The cost model combines continuous query systems and cloud pricing [49]. Plan activation cost is $\alpha_i^{(t)} = \theta_p \lambda^{(t)} + \theta_m \gamma_i$ and query assignment cost is $\beta_{ij}^{(t)} = \theta_p \gamma_j / s_i$, where processing cost $\theta_p = 1.0$ and memory cost $\theta_m = 0.36$ are derived from Amazon Timestream [49].

*3) **Baselines**:* Given the current absence of online scheduling models for trading continuous queries in the market, we evaluate *OSTOR* against two traditional approaches primarily designed for one-time query-based data trading [1, 32]:

1) **TNA** (Time-slot-based Non-Adaptive scheduling): This baseline makes scheduling decisions independently for each time slot t by maximizing the immediate social welfare $\sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{N}} (u_j^{(t)} - \beta_{ij}^{(t)}) x_{ij}^{(t)} - \sum_{i\in\mathcal{M}} \alpha_i^{(t)} y_i^{(t)}$, subject to the assignment constraints. TNA utilizes complete information within the current time slot and can quickly respond to temporal changes in utilities and costs. The key difference from *OSTOR* is that TNA does not consider long-term budget constraints in its decision-making process, while *OSTOR* employs dynamic scaling factors $w_j^{(t)}$ to balance immediate gains against future opportunities.
2) **TOFF** (Time-zero Offline scheduling): This baseline determines the entire scheduling strategy at initial time $t = 0$, assuming $u_j^{(t)} = u_j^{(0)}$, $\beta_{ij}^{(t)} = \beta_{ij}^{(0)}$, and $\alpha_i^{(t)} = \alpha_i^{(0)}$ for all $t \in \mathcal{T}$. TOFF provides consistent scheduling decisions and can optimize the global objective if the system remains static. The key difference from *OSTOR* is that TOFF cannot adapt to temporal variations in utilities and costs, while *OSTOR* continuously adjusts its decisions based on the latest system information through its adaptive dual descent mechanism.

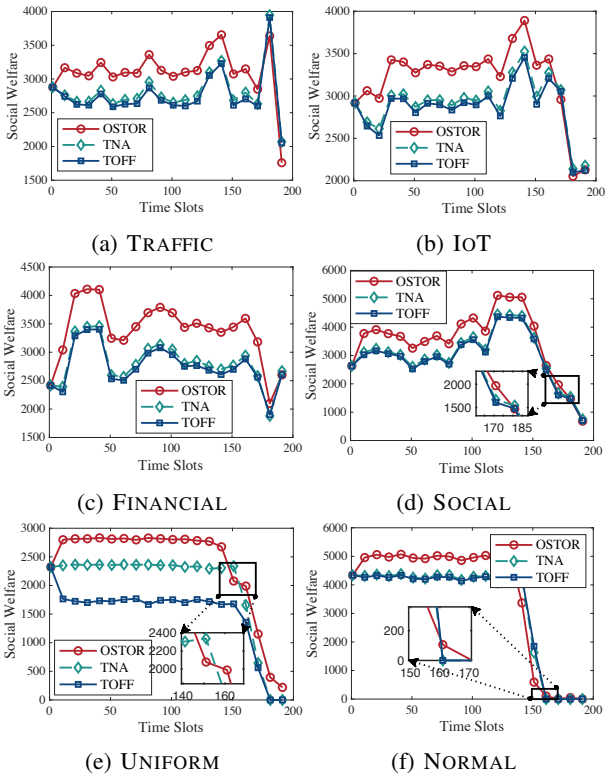


Fig. 6: Performance across time span.

These baselines represent two fundamental approaches in data trading: immediate optimization (TNA) and static global planning (TOFF), serving as meaningful references for evaluating *OSTOR*'s performance in handling both temporal dynamics and long-term constraints.

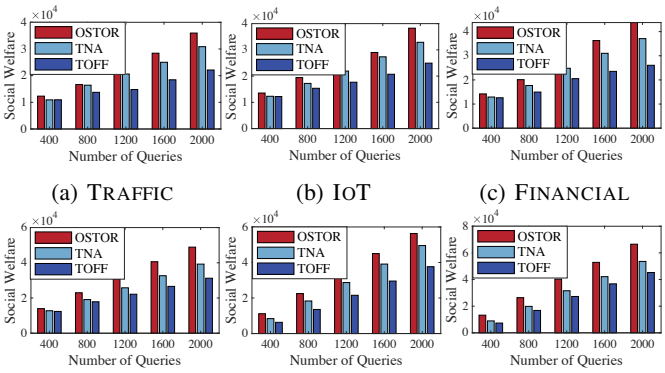
B. Result Analysis

In this subsection, we analyze experimental results. We evaluate framework performance over time in Section VI-B1, analyze parameter sensitivity in Section VI-B2, and conduct ablation studies on greedy optimization strategies in Section VI-B3. All results are averaged over ten runs.

*1) **Overall Performance**:* We compare the social welfare of different algorithms across six datasets over a 200-time-slot period, as illustrated in Fig. 6. The experimental results demonstrate that *OSTOR* consistently outperforms baseline algorithms. Compared to TOFF, it achieves average improvements of 12.58%, 11.97%, 18.42%, 15.93%, 57.31%, and 9.77% in Traffic, IoT, Financial, Social, Uniform, and Norm datasets, respectively. The improvements over TNA are 9.53%, 9.00%, 16.65%, 14.14%, 15.89%, and 9.76%. In real-world datasets (i.e., the first four datasets), *OSTOR* maintains a consistent advantage throughout the period from time slot 0 to 150, effectively managing natural fluctuations. During

slots 160-200 (i.e., budget exhaustion period), *OSTOR* shows gradual degradation in Uniform dataset compared to sharp drops in baselines. The performance patterns exhibit distinct characteristics between real-world and synthetic datasets, with real-world data showing higher volatility while Uniform and Norm display more stable trajectories. The significant improvement in Uniform (57.31%) demonstrates *OSTOR*'s effectiveness under long-term constraints. These experimental results validate *OSTOR*'s robust performance in managing budget allocation under information uncertainty.

*2) **Parameter Study**:* This subsection investigates the parameter sensitivity of our framework by systematically varying key parameters. We analyze the impact of each parameter while fixing others at their default values, evaluating how these variations affect the framework's performance. All metrics are averaged over the entire time horizon for fair comparison.



(a) TRAFFIC (b) IOT (c) FINANCIAL

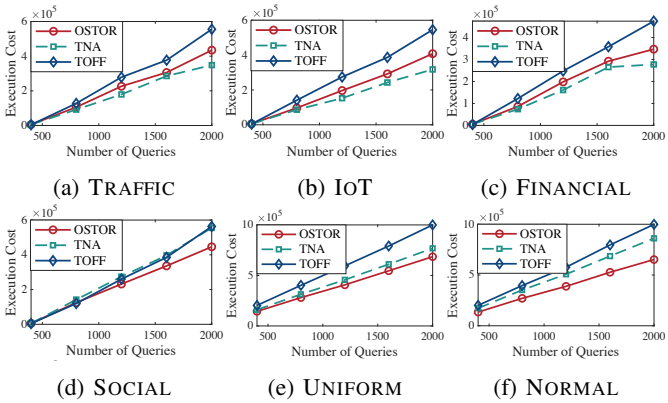
(d) SOCIAL (e) UNIFORM (f) NORMAL

Fig. 7: Impact of the number of queries.

a) Number of Queries: In this experiment, we evaluate the scalability of *OSTOR* by varying the number of queries from 400 to 2000, as illustrated in Figs. 7 and 8. The results indicate that *OSTOR* consistently outperforms the baseline methods across all datasets.

As shown in Fig. 7, *OSTOR* consistently outperforms baseline methods in social welfare. In network-centric datasets (Traffic and IoT), it improves social welfare by up to 31.34% over TOFF and 11.11% over TNA. In dynamic datasets (Financial and Social), its advantage grows with higher query loads, reaching 31.83% over TOFF and 17.77% over TNA. For periodic datasets (Weather and Power), it captures recurring patterns, surpassing TOFF by 30.70% and TNA by 18.65%. In synthetic datasets (Uniform and Normal), *OSTOR* demonstrates strong scalability, achieving up to 35.66% and 21.67% improvements over TOFF and TNA, respectively.

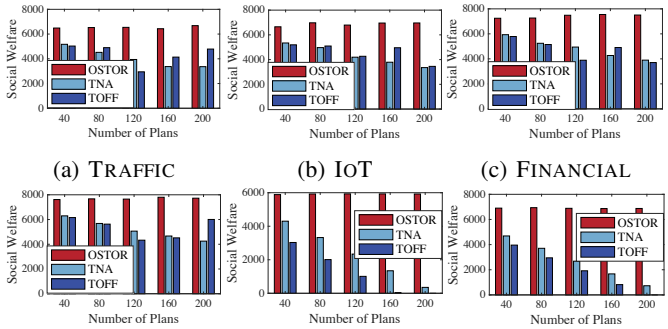
Regarding execution costs, as illustrated in Fig. 8, *OSTOR* achieves substantial cost reductions over TOFF across datasets. Savings reach 26.14% in network-centric, 23.03% in dynamic, 37.03% in periodic, and 33.55% in synthetic datasets. The cost advantage grows as queries exceed 1200, especially in periodic datasets, where *OSTOR* leverages data periodicity. While slightly costlier than TNA due to proactive resource



(a) TRAFFIC (b) IOT (c) FINANCIAL

(d) SOCIAL (e) UNIFORM (f) NORMAL

Fig. 8: Impact of the number of queries.



(a) TRAFFIC (b) IOT (c) FINANCIAL

(d) SOCIAL (e) UNIFORM (f) NORMAL

Fig. 9: Impact of the number of plans.

allocation, it maintains a strong balance between performance and efficiency, optimizing resource use under high query loads.

b) Number of Plans: In this experiment, we further evaluate the scalability of *OSTOR* by varying the number of plans from 40 to 200, as depicted in Fig. 9. The results indicate that *OSTOR* consistently outperforms baseline algorithms.

OSTOR demonstrates significant performance gains across all datasets, consistently surpassing baseline methods. In network-centric datasets, it improves social welfare by 37.65% over TNA and 33.27% over TOFF, optimizing resource allocation under high connectivity. In dynamic datasets, it outperforms TNA by 35.72% and TOFF by 36.75%, adapting to evolving conditions. For periodic datasets, *OSTOR* achieves a 45.00% improvement over TOFF by leveraging temporal patterns. The largest gains occur in synthetic datasets, exceeding TNA by 60% and TOFF by 79.45%, demonstrating robustness across distributions. As plans increase, baseline methods degrade, especially in synthetic datasets, where TNA and TOFF decline beyond 120 plans due to limited adaptability.

These results highlight *OSTOR*'s scalability in maintaining high social welfare across plan numbers, particularly in real-world datasets, where it remains stable within 6000–7000. This robustness is evident in synthetic datasets, where *OSTOR* adapts to diverse distributions, mitigating baseline degradation. Moreover, these results validate *OSTOR*'s ability to manage complex data dynamics across query plan complexities.

TABLE I: SOCIAL WELFARE OF ABLATION STUDY.

Dataset	MR2	OT-1	OT-2	*OSTOR*
TRAFFIC	3.0760	3.0990	3.1803	**3.2046**
IOT	3.0978	3.1397	3.2100	**3.2328**
FINANCIAL	3.1438	3.1657	3.2386	**3.2640**
SOCIAL	3.2447	3.3234	3.3394	**3.4056**
UNIFORM	2.0316	2.1171	2.2097	**2.2937**
NORMAL	3.3503	3.3924	3.4671	**3.5135**

*3) **Ablation Study**:* Through ablation studies, we investigate the impact of two greedy optimization strategies in Section IV-B: Dynamic Reassignment Strategy (*DRS*) and Iterative Reactivation Strategy (*IRS*). We evaluate four variants: (1) MR2: no greedy strategies; (2) OT-1: only *IRS*; (3) OT-2: only *DRS*; (4) *OSTOR*: both strategies. Table I shows the normalized social welfare across algorithm variants.

The *OSTOR* framework consistently achieves the best performance by combining the two greedy strategies. When examining individual strategies, *DRS* shows stronger improvements over *IRS* across datasets. In Traffic and IoT datasets, *DRS* (OT-2) outperforms *IRS* (OT-1) by 0.0813 and 0.0703, respectively. The Financial dataset shows similar patterns, with OT-2 achieving 3.2386, exceeding OT-1's 3.1657 by 0.0729. In the Social dataset, the performance gap between strategies narrows, with OT-2 (3.3394) showing a 0.0160 improvement over OT-1 (3.3234). Comparing the full *OSTOR* implementation to the baseline MR2, we observe consistent improvements: 0.1286 in Traffic, 0.1350 in IoT, 0.1202 in Financial, 0.1609 in Social, 0.2621 in Uniform, and 0.1632 in Norm datasets. The most significant improvement appears in the Uniform dataset, where *OSTOR* achieves 2.2937, surpassing MR2's 2.0316 by 0.2621. These results demonstrate the effectiveness of combining these greedy strategies in *OSTOR* to achieve a robust solution.

In this section, we have verified the framework's performance on both real-world and synthetic datasets, demonstrating *OSTOR*'s adaptability across diverse scenarios. Through comprehensive parameter analysis and ablation studies, we validate the robustness of the proposed framework.

VII. DISCUSSION

In this section, we discuss the practical considerations and potential limitations of *OSTOR* to provide a comprehensive understanding of its applicability in real-world scenarios.

A. Practical Implementation

Deploying *OSTOR* in real-world systems requires integration with stream processing platforms like Apache Kafka and Flink [45, 66]. While effective in controlled settings, real deployments involve added complexities. Below, we discuss several practical considerations:

- Real-time Utility Estimation and Monitoring: Implementation requires accurate utility estimation over time-based windows, which aligns well with modern stream processing platforms [67]. These estimators should integrate with monitoring tools and emphasize recent data to effectively track metrics such as throughput, latency, and data quality.
- Scalability Management: For high-velocity and high-volume data streams, scalability becomes crucial. The implementation strategy should incorporate efficient data structures, caching mechanisms, and pre-computed result storage.
- Performance Optimization: To ensure optimal performance in stream processing environments, we recommend minimizing network delays through batch processing of similar queries and implementing strategic component placement and stream partitioning [68].
- System Reliability: The implementation should leverage built-in fault-tolerance features such as checkpointing and state management [69] to ensure reliable query trading and maintain system stability under high load or failures.

B. Limitations and Challenging Scenarios

Despite OSTOR's robust performance, several limitations and challenging scenarios may affect its effectiveness:

- Dynamic Environment Challenges: Frequent changes in user preferences and data patterns may increase uncertainty in utility and cost estimation, potentially requiring more computation. Techniques such as adaptive learning rates [70], online learning [71], and bandit algorithms [72] could help the system adapt more effectively.
- Extreme Utility Distribution Issues: When user utilities vary greatly or follow skewed distributions, *ADD*'s convergence may slow down. Approaches like normalization, parameter tuning, robust optimization [73], and distribution transformations [74] may improve stability.
- Strategic Behavior Concerns: The system may be vulnerable to misreported utilities or coordinated manipulation. Incorporating reputation systems [75], historical behavior analysis, or game-theoretic models [76] could help mitigate such risks.

Each challenge represents an opportunity for future research and system enhancement, with established frameworks providing potential solutions.

VIII. CONCLUSION

This paper has presented *OSTOR*, the first online scheduling framework for trading continuous queries. The framework addresses two key challenges in trading continuous query: information uncertainty and computational complexity. To tackle information uncertainty, *OSTOR* has decomposed the online scheduling problem into one-round problems that only require current system information. For computational complexity, it has employed an adaptive dual descent (*ADD*) algorithm with bounded approximation ratio with polynomial time complexity, augmented with two structure-aware greedy optimization strategies: dynamic reassignment (*DRS*) and iterative reactivation (*IRS*). Our extensive experimental evaluation has demonstrated that *OSTOR* has achieved substantial improvements in social welfare and considerable reductions in query execution costs on both real-world and synthetic datasets, compared to existing data trading methods. Future research directions should include extending the framework to address information asymmetry scenarios, where buyers' utility functions remain private and unknown to the seller.

REFERENCES

[1] Z. Cong, X. Luo, J. Pei, F. Zhu, and Y. Zhang, "Data pricing in machine learning pipelines," *Knowledge and Information Systems*, vol. 64, no. 6, pp. 1417–1455, 2022.

[2] Grand View Research, "Data marketplace market report." https://www.grandviewresearch.com, 2023.

[3] AWS Data Exchange. https://aws.amazon.com/cn/data-exchange.

[4] SnowflakeMarketplace. https://www.snowflake.com/en/data-cloud/marketplace.

[5] Xignite. https://www.xignite.com.

[6] Y. An, Z. Zhen, S. Zhang, R. Zhu, and C. Zong, "Approximate continuous k representative skyline queries over memory limitation-based streaming data," in *International Conference on Advanced Data Mining and Applications*, pp. 94–106, Springer, 2023.

[7] R. Zhu, Y. Jia, X. Yang, B. Zheng, B. Wang, and C. Zong, "Multiple continuous top-k queries over data stream," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 1575–1588, IEEE, 2024.

[8] M. Fragkoulis, P. Carbone, V. Kalavri, and A. Katsifodimos, "A survey on the evolution of stream processing systems," *The VLDB Journal*, vol. 33, no. 2, pp. 507–541, 2024.

[9] Q. Liu, A. King, and T. Ge, "Reducing resource usage for continuous model updating and predictive query answering in graph streams," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 2653–2666, IEEE, 2024.

[10] X. Dai, Z. Wang, J. Xie, X. Liu, and J. C. Lui, "Conversational recommendation with online learning and clustering on misspecified users," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[11] M. Liu, Z. Li, K. Cai, J. Allcock, S. Zhang, and J. C. Lui, "Quantum bgp with online path selection via network benchmarking," in *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*, pp. 1401–1410, IEEE, 2024.

[12] X. Wang, J. Ye, and J. C. Lui, "Online learning aided decentralized multi-user task offloading for mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 23, no. 4, pp. 3328–3342, 2023.

[13] J. Ye, D. Lin, K. Cai, C. Zhou, J. He, and J. C. Lui, "Data-driven rate control for rdma networks: A lightweight online learning approach," in *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*, pp. 1–11, IEEE, 2023.

[14] M. Xiao, M. Li, and J. J. Zhang, "Locally differentially private personal data markets using contextual dynamic pricing mechanism," *IEEE Transactions on Dependable and Secure Computing*, 2023.

[15] Z. Liu, C. Hu, C. Ruan, L. Zhang, P. Hu, and T. Xiang, "A privacy-preserving matching service scheme for power data trading," *IEEE Internet of Things Journal*, 2024.

[16] Z. He and Z. Cai, "Trading aggregate statistics over private internet of things data," *IEEE Transactions on Computers*, 2023.

[17] H. Cai, Y. Yang, W. Fan, F. Xiao, and Y. Zhu, "Towards correlated data trading for high-dimensional private data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 3, pp. 1047–1059, 2023.

[18] J. Li, J. Li, X. Wang, R. Qin, Y. Yuan, and F.-Y. Wang, "Multi-blockchain based data trading markets with novel pricing mechanisms," *IEEE/CAA Journal of Automatica Sinica*, vol. 10, no. 12, pp. 2222–2232, 2023.

[19] R. Castro Fernandez, "Protecting data markets from strategic buyers," SIGMOD '22, p. 1755–1769, 2022.

[20] J. Cheng, N. Ding, J. C. Lui, and J. Huang, "Continuous query-based data trading," in *Abstracts of the 2024 ACM SIGMETRICS/IFIP PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems*, pp. 73–74, 2024.

[21] J. Park, K. Han, and B. Lee, "Green cloud? an empirical analysis of cloud computing and energy efficiency," *Management Science*, vol. 69, no. 3, pp. 1639–1664, 2023.

[22] Y. Fu, X. Miao, H. Peng, C. Na, S. Deng, and J. Yin, "Online query-based data pricing with time-discounting valuations," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 3449–3461, IEEE, 2024.

[23] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[24] K. Lee and Y. Kim, "Online pricing and resource scheduling for profit maximization of cloud storage providers," *IEEE Transactions on Cloud Computing*, 2024.

[25] W. Shi, L. Zhang, C. Wu, Z. Li, and F. C. Lau, "An online auction framework for dynamic resource provisioning in cloud computing," *IEEE/ACM transactions on networking*, vol. 24, no. 4, pp. 2060–2073, 2015.

[26] J. Mohan, A. Phanishayee, J. Kulkarni, and V. Chidambaram, "Looking beyond gpus for dnn scheduling on multi-tenant clusters," in *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pp. 579–596, 2022.

[27] H. Cui, Z. Tang, J. Lou, W. Jia, and W. Zhao, "Latency-aware container scheduling in edge cluster upgrades: A deep reinforcement learning approach," *IEEE Transactions on Services Computing*, 2024.

[28] Q. Deng, Q. Zuo, and Z. Li, "Privacy-preserving stable data trading for unknown market based on blockchain," *IEEE Transactions on Mobile Computing*, 2025.

[29] P. Abla, T. Li, D. He, H. Huang, S. Yu, and Y. Zhang, "Fair and privacy-preserved data trading protocol by exploiting blockchain," *IEEE Transactions on Information Forensics and Security*, 2024.

[30] H. Hou, L. Qiao, Y. Yuan, C. Chen, and G. Wang, "A scalable query pricing framework for incomplete graph data," in *International Conference on Database Systems*

for Advanced Applications, pp. 97–113, Springer, 2023.
[31] C. Chen, Y. Yuan, Z. Wen, Y.-P. Wang, and G. Wang, "Gshop: Towards flexible pricing for graph statistics," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 2612–2624, IEEE, 2024.
[32] M. Zhang, F. Beltrán, and J. Liu, "A survey of data pricing for data marketplaces," *IEEE Transactions on Big Data*, 2023.
[33] W. Chen, R. Huo, C. Sun, S. Wang, and T. Huang, "Efficient and non-repudiable data trading scheme based on state channels and stackelberg game," *IEEE Transactions on Mobile Computing*, 2024.
[34] J. Fang, T. Feng, X. Guo, R. Ma, and Y. Lu, "Blockchain-cloud privacy-enhanced distributed industrial data trading based on verifiable credentials," *Journal of cloud computing*, vol. 13, no. 1, p. 30, 2024.
[35] H. Xu, S. Qi, Y. Qi, W. Wei, and N. Xiong, "Secure and lightweight blockchain-based truthful data trading for real-time vehicular crowdsensing," *ACM Transactions on Embedded Computing Systems*, vol. 23, no. 1, pp. 1–31, 2024.
[36] I. Bauer-Hänsel, Q. Liu, C. J. Tessone, and G. Schwabe, "Designing a blockchain-based data market and pricing data to optimize data trading and welfare," *International Journal of Electronic Commerce*, vol. 28, no. 1, pp. 3–30, 2024.
[37] T. Singh, R. Kalra, S. Mishra, Satakshi, and M. Kumar, "An efficient real-time stock prediction exploiting incremental learning and deep learning," *Evolving Systems*, vol. 14, no. 6, pp. 919–937, 2023.
[38] L. Antonelli, H. Badir, H. Bazza, S. Bimonte, S. Rizzi, *et al.*, "Requirements engineering for continuous queries on iort data: A case study in agricultural autonomous robots monitoring," in *Proceedings of the 26th International Conference on Enterprise Information Systems (Volume 2)*, vol. 2, pp. 113–120, SciTePress, 2024.
[39] M. Zhang, X. Li, Y. Miao, B. Luo, W. Xu, Y. Ren, and R. H. Deng, "Privacy-preserved data disturbance and truthfulness verification for data trading," *IEEE Transactions on Information Forensics and Security*, 2024.
[40] R. Zhou, X. Zhang, J. C. Lui, and Z. Li, "Dynamic pricing and placing for distributed machine learning jobs: An online learning approach," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 4, pp. 1135–1150, 2023.
[41] F. Liang, Z. Zhang, H. Lu, C. Li, V. Leung, Y. Guo, and X. Hu, "Resource allocation and workload scheduling for large-scale distributed deep learning: A survey," *arXiv preprint arXiv:2406.08115*, 2024.
[42] Q. Zhang, M. Ikram, and K. Xu, "Online optimization of vehicle-to-grid scheduling to mitigate battery aging," *Energies*, vol. 17, no. 7, p. 1681, 2024.
[43] J. R. Daduna and L. Xie, "Vehicle scheduling," in *Encyclopedia of Optimization*, pp. 1–7, Springer, 2024.
[44] A. Arasu, S. Babu, and J. Widom, "The cql continuous query language: semantic foundations and query execution," *The VLDB Journal*, vol. 15, pp. 121–142, 2006.
[45] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache flink: Stream and batch processing in a single engine," *The Bulletin of the Technical Committee on Data Engineering*, vol. 38, no. 4, 2015.
[46] Y. Zhong, J. Li, and S. Zhu, "Continuous spatial keyword search with query result diversifications," *World Wide Web*, vol. 26, no. 4, pp. 1935–1948, 2023.
[47] I. Livaja, K. Pripužić, S. Sovilj, and M. Vuković, "A distributed geospatial publish/subscribe system on apache spark," *Future generation computer systems*, vol. 132, pp. 282–298, 2022.
[48] S. Babu and J. Widom, "Continuous queries over data streams," *ACM Sigmod Record*, vol. 30, no. 3, pp. 109–120, 2001.
[49] Amazon Web Services. https://aws.amazon.com/.
[50] A. Mhedhbi, C. Kankanamge, and S. Salihoglu, "Optimizing one-time and continuous subgraph queries using worst-case optimal joins," *ACM Transactions on Database Systems (TODS)*, vol. 46, no. 2, pp. 1–45, 2021.
[51] V. Rosenfeld, S. Breß, and V. Markl, "Query processing on heterogeneous cpu/gpu systems," *ACM Computing Surveys (CSUR)*, vol. 55, no. 1, pp. 1–38, 2022.
[52] M. Sharaf and A. Labrinidis, *Scheduling Strategies for Data Stream Processing*, pp. 2475–2479. Boston, MA: Springer US, 2009.
[53] D. Tang, Z. Shang, W. W. Ma, A. J. Elmore, and S. Krishnan, "Resource-efficient shared query execution via exploiting time slackness," in *Proceedings of the 2021 International Conference on Management of Data*, pp. 1797–1810, 2021.
[54] R. M. Karp, *Reducibility among combinatorial problems*. Springer, 2010.
[55] Technical Report. https://drive.google.com/file/d/16JGQiyOQraoHY4HPtaL-Ouf-XCfuaupY/view?usp=sharing.
[56] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
[57] R. Zhou, Z. Li, and C. Wu, "Scheduling frameworks for cloud container services," *IEEE/acm transactions on networking*, vol. 26, no. 1, pp. 436–450, 2018.
[58] R. Zhou, Z. Li, and C. Wu, "An efficient online placement scheme for cloud container clusters," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1046–1058, 2019.
[59] M. Gruben, "Austin weather dataset," 2019.
[60] A. Kannal, "Solar power generation data," 2020.
[61] UMass Trace Repository, "Investigating traffic analysis attacks on apple icloud private relay." https://traces.cs.umass.edu/docs/traces/network/.
[62] Tianchi, "Ddos botnet attack on iot devices." https://tianchi.aliyun.com/dataset/92825.
[63] B. Marjanovic, "Price-volume data for all us stocks & etfs," 2017.
[64] K. Parmar, "Social media sentiments analysis dataset,"

2023.
[65] W. Yue, L. Benson, and T. Rabl, "Desis: Efficient window aggregation in decentralized networks," EDBT, 2023.
[66] J. Kreps, N. Narkhede, J. Rao, *et al.*, "Kafka: A distributed messaging system for log processing," in *Proceedings of the NetDB*, vol. 11, pp. 1–7, Athens, Greece, 2011.
[67] Z. Chen, F. Zhang, Y. Chen, X. Fang, G. Feng, X. Zhu, W. Chen, and X. Du, "Enabling window-based monotonic graph analytics with reusable transitional results for pattern-consistent queries," *Proceedings of the VLDB Endowment*, vol. 17, no. 11, pp. 3003–3016, 2024.
[68] T. P. Raptis, C. Cicconetti, and A. Passarella, "Efficient topic partitioning of apache kafka for high-reliability real-time data streaming applications," *Future Generation Computer Systems*, vol. 154, pp. 173–188, 2024.
[69] G. Siachamis, K. Psarakis, M. Fragkoulis, A. Van Deursen, P. Carbone, and A. Katsifodimos, "Checkmate: Evaluating checkpointing protocols for streaming dataflows," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 4030–4043, IEEE, 2024.
[70] H. Sun, L. Shen, Q. Zhong, L. Ding, S. Chen, J. Sun, J. Li, G. Sun, and D. Tao, "Adasam: Boosting sharpness-aware minimization with adaptive learning rate and momentum for training deep neural networks," *Neural Networks*, vol. 169, pp. 506–519, 2024.
[71] E. Bartz and T. Bartz-Beielstein, *Online Machine Learning*. Springer, 2024.
[72] Y. Lin, Y. Endo, J. Lee, and S. Kamijo, "Bandit-nas: Bandit sampling and training method for neural architecture search," *Neurocomputing*, vol. 597, p. 127684, 2024.
[73] C. Li, S. Han, S. Zeng, and S. Yang, "Robust optimization," in *Intelligent Optimization: Principles, Algorithms and Applications*, pp. 239–251, Springer, 2024.
[74] A. H. Zemanian, *Distribution theory and transform analysis: an introduction to generalized functions, with applications*. Courier Corporation, 1987.
[75] C. P. Fernandes, C. Montez, D. D. Adriano, A. Boukerche, and M. S. Wangham, "A blockchain-based reputation system for trusted vanet nodes," *Ad Hoc Networks*, vol. 140, p. 103071, 2023.
[76] A. E. Atakan and M. Ekmekci, "The role of information in auctions," *Journal of Mathematical Economics*, vol. 114, p. 103027, 2024.