UNIVERSITY OF CALIFORNIA

Los Angeles

Efficient Algorithms for Partial Information Management: Bandit Problems and Graph Neural Networks

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Electrical and Computer Engineering

by

Jialin Dong

2024

ABSTRACT OF THE DISSERTATION

Efficient Algorithms for Partial Information Management: Bandit Problems and Graph Neural Networks

by

Jialin Dong
Doctor of Philosophy in Electrical and Computer Engineering
University of California, Los Angeles, 2024
Professor Lin Yang, Chair

Abstract:

This thesis explores the development of efficient algorithms for managing partial information in complex systems, focusing on two key areas: Graph Neural Networks (GNNs) and Bandit Problems. In an era where the boundaries between physical and digital realms are rapidly blurring, these seemingly disparate fields have emerged as integral components in addressing the challenges of modern interconnected systems. The research journey begins with investigations into blind demixing for Internet-of-Things applications, naturally progressing to explorations in nonconvex optimization and high-dimensional statistical analysis. This foundation serves as a springboard for novel contributions in GNNs and Bandit Problems, addressing the pressing need for robust, scalable, and intelligent algorithms capable of handling the complexity of data-driven applications in our increasingly connected world.

The work introduces a Global Neighborhood Sampling algorithm for efficient GNN training on giant graphs, specifically optimized for mixed CPU-GPU hardware setups. This innovation significantly reduces data movement between CPU and GPU, leading to substantial performance improvements over existing state-of-the-art sampling methods. Building on this, the thesis presents G-RAG, a graph-based reranking approach for Retrieval Augmented Generation systems. G-RAG leverages both the connections between retrieved documents and their semantic information, providing a context-informed reranker that outperforms current methods while maintaining a smaller computational footprint.

Delving into the realm of bandit problems, the thesis explores sparse bandit learning with misspecified linear features. This investigation provides valuable insights into how structural assumptions can aid in misspecified bandit learning, demonstrating that algorithms can obtain near-optimal actions by querying a number of actions that scale exponentially with the sparsity parameter rather than the ambient dimension. Furthermore, the research presents a novel feature-mapping framework for solving Markov Decision Processes with delayed feedback, where the agent's observations are delayed by multiple time steps. By carefully addressing the statistical challenges introduced by overlapping action sequences, this approach achieves a regret bound that is independent of the size of the state and action spaces.

The thesis also develops an online stochastic gradient descent (SGD)-based algorithm for stochastic bandit problems with general parametric reward functions. This method employs an action-elimination strategy and a uniform action-selection approach, providing high-probability regret guarantees and effectively handling the bias introduced by greedy action selection. This contribution extends the applicability of bandit algorithms to a broader class of problems with complex reward structures.

Throughout the research, a common thread emerges: the importance of understanding and leveraging the inherent structure in complex systems. This realization serves as a bridge between the work on GNNs and bandit problems, highlighting their complementary nature in modeling and decision-making within large-scale, interconnected systems. The synthesis of these areas leads to novel insights and methodologies with the potential to impact a wide range of applications, from improving recommendation systems and enhancing financial modeling to optimizing large-scale infrastructure networks and advancing human-AI interaction.

This thesis stands at the intersection of several critical domains in computer science and applied mathematics, building upon foundational work in optimization, statistical learning, and network analysis while addressing the pressing needs of emerging technologies in AI and IoT. By bridging theoretical and empirical perspectives, the research advances the state-of-the-art in efficient algorithms for partial information management. The developed techniques not only push the boundaries of our understanding but also offer practical solutions to real-world challenges in managing and leveraging partial information in complex systems.

In conclusion, this work contributes to the development of more robust, scalable, and intelligent systems capable of navigating the complexities of our data-driven world. As we continue to face challenges in areas such as large-scale graph learning, decision-making under uncertainty, and human-AI interaction, the algorithms and insights presented in this thesis pave the way for future advancements in the field, offering a foundation for more adaptive and efficient approaches to partial information management in the ever-evolving landscape of interconnected systems.

The dissertation of Jialin Dong is approved.

University of California, Los Angeles

2024

To my parents who showered me with unconditional love,

and my friends who supported me through all times.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Most importantly, I would like to express my deepest gratitude to my advisor Professor Yang for expanding my research interests and allowing me to explore diverse topics, pursue internships, and collaborate with others during my Ph.D. study. My journey to this point began with my graduate experience. My desire to push boundaries, broaden my horizons, experience new things, and learn from different research approaches made me continue with a Ph.D. when I already had some publications in wireless communication during graduate studies. I am grateful to my graduate advisors, Professor Shi, who introduced me to the world of research, and Professor Lin, who expanded my perspective with his vast expertise and profound knowledge.

I feel incredibly fortunate to have studied at UCLA, which has rich resources and opportunities. During my Ph.D., I have been blessed with the support of my committee members. Professor Vandenberghe's book on convex optimization was my research starting point in 2017, and it has been an honor to take his Convex Optimization course at UCLA. Professor Fragouli's reading group has been instrumental in helping me develop a comprehensive understanding of the bandit problem. I am thankful to Professor Kao for his support during my preliminary exam, and for offering me a grader position, which was vital for my financial support.

The teaching experience at UCLA has been equally rewarding. I am grateful to have been a TA for Professors Ian Robert and Yang Zhang, whose classes are related to my undergraduate major. Their passion for teaching reminded me of my initial encounters with these subjects and the wonderful professors I had then. The TA positions were crucial in supporting my Ph.D. studies. Throughout this journey, Deeona has been an invaluable source of support, helping me overcome various challenges.

My academic journey has been complemented by invaluable industry experiences. My internships at JP Morgan, Google, and Amazon in New York and Seattle have allowed me to explore different cities and collaborate with industry professionals. I am thankful for the amazing mentors I met, including Kshama and Yu-Husan, who understood my thoughts and provided helpful advice.

Collaborations with Anton, Bahare, and Bryan reignited the passion I felt during my first research project, and I learned a great deal from their constructive suggestions. However, it all started with my collaboration with Da during my first year internship at AWS. His earnestness and self-discipline continue to influence me today. Additionally, I want to acknowledge my first work experience as an RA in Hong Kong with Professor Jun Zhang, whose attitude towards work and research has had a profound impact on me.

Lastly, but certainly not least, I want to express my heartfelt thanks to my friends and parents. Their unwavering support has been a constant source of strength during every significant moment, whether at my peak or my lowest points, during job-hunting, life, or research. Their presence and encouragement have been instrumental in my journey, and I am deeply grateful for their enduring support.

VITA

2020-2024	Ph.D. in Electrical Engineering, University of California, Los Angeles
2017-2020	M.E. in Communication Engineering, University of Chinese Academy of Sciences
2015-2016	Exchange Student in Electrical Engineering, University of Michigan Dearborn
2013-2017	B.E. in Communication Engineering, University of Electronic Science and Technology of China
2024	Applied Scientist Intern, Amazon
2023	Research Intern, Google Research
2023	Summer Associate in the AI Research program, JPMorgan Chase & Co.
2020-2021	Applied Scientist Intern, Amazon Web Services Shanghai AI-Labs
2019-2020	Research Assistant, Dept. of Electronic and Information Engineering, The Hong Kong Polytechnic University

PUBLICATIONS

J. Dong, B. Fatemi, B. Perozzi, L. F. Yang, and A. Tsitsulin, “Don’t Forget to Connect! Improving RAG with Graph-based Reranking,” *Under Review.*, 2024.

J. Dong, J. Wang, L. F. Yang “Provably Correct SGD-Based Exploration for Generalized

Stochastic Bandit Problem" in *IEEE International Conference on Smart Applications, Communications and Networking (SmartNets)*, Aug. 2024.

J. Dong, J. Wang, L. F. Yang "Delayed MDPs with Feature Mapping" in *International Joint Conference on Neural Networks (IJCNN)*, July 2024.

J. Dong, K. Dwarakanath, and S. Vyetrenko, "Analyzing the Impact of Tax Credits on Households in Simulated Economic Systems with Learning Agents AAAI 2024 Workshop on AI in Finance for Social Impact, Feb. 2024.

J. Dong and L. F. Yang, "Does Sparsity Help in Learning Misspecified Linear Bandits? in *International Conference on Machine Learning (ICML)*, Jul. 2023.

J. Dong, D. Zheng, L. F. Yang, and G. Karypis, "Global Neighbor Sampling for Mixed CPU-GPU Training on Giant Graphs," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD)*, pp. 289-299, Singapore, Aug. 2021.

Y. Shi, **J. Dong**, and J. Zhang, Low-overhead Communications in IoT Networks: Structured Signal Processing Approaches, *Springer*, Apr. 2020.

J. Dong, J. Zhang, Y. Shi, and J. Wang, "Faster activity and data detection in massive random access: A multi-armed bandit approach," *IEEE Internet of Things Journal* , vol. 9, no. 15, pp. 13664-13678, 2022.

J. Dong, Y. Shi, and Z. Ding, "Blind over-the-air computation and data fusion via provable Wirtinger flow," *IEEE Trans. Signal Process.*, vol. 68, pp. 1136-1151, Feb. 2020.

J. Dong, K. Yang, and Y. Shi, “Ranking from crowdsourced pairwise comparisons via smoothed matrix manifold optimization,” *ACM Trans. Knowl. Discovery Data.*,vol. 14, no. 2, pp. 1-26, Feb. 2020.

J. Dong, K. Yang, and Y. Shi, “Blind Demixing for Low-Latency Communication,” *IEEE Trans. Wireless Commun.*, vol. 18, no. 2, pp. 897-911, Dec. 2018.

J. Dong and Y. Shi, “Nonconvex demixing from bilinear measurements,” *IEEE Trans. Signal Process.*, vol. 66, no. 19, pp. 5152-5166, Oct. 2018.

J. Dong, J. Zhang, and Y. Shi, “Bandit sampling for faster activity and data detection in massive random access,” in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Barcelona, Spain, May 2020.

M. Fu, **J. Dong**, and Y. Shi,“Sparse blind demixing for low-latency wireless random access with massive connectivity,” in *Pro. IEEE Veh. Technol. Conf. (VTC)*, Honolulu, Hawaii, USA, Sept. 2019.

J. Dong, Y. Shi, and Z. Ding, “Sparse blind demixing for low-latency signal recovery in massive IoT connectivity,” in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Brighton, United Kingdom, May 2019.

J. Dong and Y. Shi, “Blind demixing via Wirtinger flow with random initialization,” in *Proc. Int. Conf. Artificial Intell. Stat. (AISTATS)*, Naha, Okinawa, Japan, Apr. 2019.

J. Dong, and Y. Shi, “Nonconvex demixing from bilinear measurement,” *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, Vail, Colorado, USA, Jun. 2018.

J. Dong, K. Yang, and Y. Shi, “Blind demixing for low-latency Communication,” in *Proc. IEEE Wireless Commun. Networking Conf. (WCNC)*, Barcelona, Spain, Apr. 2018.

J. Dong, K. Yang, and Y. Shi, “Ranking from crowdsourced pairwise comparisons via smoothed matrix manifold optimization,” in *ICDM Workshops on Data-driven Discovery of Models (D3M)*, New Orleans, Louisiana, USA, Nov. 2017.

CHAPTER 1

Introduction

In the era of ubiquitous connectivity and data-driven decision-making, complex systems are emerging that require novel approaches in data analysis, optimization, and machine learning. Throughout my graduate research, I have traversed a diverse range of topics, from blind demixing in Internet-of-Things to nonconvex optimization and high-dimensional statistical analysis. This journey has ultimately led me to focus on two pivotal areas of research during my PhD: the development of graph neural networks (GNNs) and the study of bandit problems, where I aim to push the boundaries of our current understanding and drive innovation in these critical fields.

These seemingly disparate fields are, in fact, integral components of a larger picture that aims to address the challenges of modern interconnected systems. My work in blind demixing for IoT applications highlighted the need for robust signal processing in high-dimensional spaces, naturally leading to an exploration of nonconvex optimization techniques. This, in turn, opened doors to the realm of high-dimensional statistical analysis, where the interplay between data structure and algorithmic efficiency became increasingly apparent.

As I delved deeper into these topics, a common thread emerged: the importance of understanding and leveraging the inherent structure in complex systems. This realization served as a bridge to my work on graph neural networks during internships at Amazon and Google. GNNs, with their ability to capture and process relational data, presented a powerful framework for modeling the intricate dependencies in networked systems, from social networks, and computer version to natural language processing.

Concurrently, my experience with multi-agent reinforcement learning at JP Morgan illuminated the challenges of decision-making in dynamic, interactive environments. This naturally led to an interest in bandit problems, which offer a principled approach to balancing exploration and exploitation in uncertain scenarios. The recent work on fine-tuning multi-turn chatbots at Amazon further underscored the relevance of sequential decision-making in human-AI interaction contexts.

Based on all my background, this thesis aims to push the boundaries of our understanding and application of graph neural networks and bandit problems in the context of large-scale, interconnected systems. By bridging these two areas, we seek to develop more adaptive, efficient, and interpretable algorithms capable of handling the complexity of modern data-driven applications.

Our research stands at the intersection of several critical domains in computer science and applied mathematics. It builds upon foundational work in optimization, statistical learning, and network analysis while addressing the needs of emerging technologies in AI/ML. Through this work, we aim to contribute to the development of more robust, scalable, and intelligent systems that can navigate the complexities of our increasingly connected world.

In many real-world systems, the information available to decision-makers is often incomplete or partial. This partial information can arise due to various factors, such as sensor limitations, privacy concerns, or the inherent complexity of the underlying system. Efficiently utilizing this partial information to achieve desired objectives is a fundamental challenge in numerous fields, including machine learning, optimization, and control.

This thesis focuses on developing efficient algorithms that can effectively leverage partial information to tackle complex problems. The research presented in this work spans two key areas: bandit problems and graph neural networks.

Bandit problems are a class of sequential decision-making problems where the decision-maker must choose an action from a set of alternatives, and the reward associated with each action is initially unknown. In many real-world scenarios, the decision-maker may only have access to partial information about the rewards, such as sparse or delayed feedback. This thesis explores novel algorithms that can efficiently navigate these partial information settings, providing strong theoretical guarantees and empirical performance.

This work's second focus is the use of GNNs for learning from graph-structured data. Graph-based data is involved in various domains, such as social networks, recommendation systems, and biological networks. However, training Graph Neural Networks (GNNs) on large-scale graphs can be computationally challenging, as the full graph structure may not fit in memory and may incur high computational costs. This thesis presents efficient sampling-based algorithms that enable GNN training on giant graphs in mixed CPU-GPU hardware setups, where data movement between the CPU and GPU can be a significant bottleneck.

The key contributions of this thesis are as follows:

1. **Efficient Sampling for GNN Training on Giant Graphs**: We propose a novel Global Neighborhood Sampling algorithm that enables efficient training of GNNs on industry-scale graphs, specifically designed for mixed CPU-GPU hardware setups. This approach significantly reduces the data movement between the CPU and GPU, leading to substantial performance improvements over state-of-the-art sampling methods. It demonstrates that partial neighborhood information, when utilized under a well-designed algorithm, is sufficient for effective GNN training.

2. **Graph-based Reranking for Retrieval Augmented Generation**: Reranking is an effective method for optimizing the use of partial information. We introduce G-RAG, a graph neural network-based reranking approach for Retrieval Augmented Generation (RAG) systems. G-RAG leverages both the connections between retrieved documents and their semantic information to provide a context-informed reranker, outperforming state-of-the-art methods while having a smaller computational cost.

3. **Sparse Bandit Learning with Misspecified Linear Features**: We explore the impact of sparsity on the learnability of misspecified linear bandits, a challenging setting where the true reward function is not perfectly captured by the given linear features. We show that algorithms can obtain near-optimal actions by querying several actions that scale exponentially with the

sparsity parameter, rather than the ambient dimension, providing a deeper understanding of how to utilize partial information in misspecified bandit learning.

4. **Feature Mapping in Delayed Markov Decision Processes**: We propose a new feature-mapping-based framework to solve Markov Decision Processes with delayed feedback, where the agent only has partial observations due to the delay. By carefully addressing the statistical challenges introduced by the overlapping action sequences, our algorithm achieves a regret bound that is independent of the size of the state and action spaces.

5. **SGD-based Exploration for Generalized Stochastic Bandits**: We develop an online stochastic gradient descent (SGD)-based algorithm for stochastic bandit problems with general parametric reward functions. By employing an action-elimination strategy and a uniform action-selection approach, our method effectively leverages partial gradient information, yielding high-probability regret guarantees and robust performance.

Through these contributions, this thesis advances the state-of-the-art in efficient algorithms for partial information management, bridging the gap between theoretical and empirical perspectives. The developed techniques have the potential to significantly impact a wide range of applications, from large-scale graph learning to decision-making under uncertainty.

In the following chapters, we will investigate the theoretical foundations, methodological innovations, and practical applications of our research, showcasing how this integrated approach can address some of the most challenging problems in modern computing and decision-making systems.

The remainder of this chapter introduces essential notation, concepts, and definitions that will be utilized throughout chapters 2, 3, 4, 5, and 6. The results presented in chapters 2, 4, 5, and 6 have been published as [1], [2], [3], and [4], respectively. Chapter 3 is under review [5].

1.1 Notation

In this dissertation, we use the following notations. The Euclidean norm of a vector $\mathbf{x}$ is denoted by $\|\mathbf{x}\|_2$, and the spectral norm of a matrix $\mathbf{M}$ is denoted by $\|\mathbf{M}\|$. Given a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, the Frobenius norm is denoted as: $\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$ where a_{ij} represents the element in the i-th row and j-th column of the matrix $\mathbf{A}$. We denote the transpose of any column vector $\mathbf{x}$ by $\mathbf{x}^\top$. For any vectors $\mathbf{x}$ and $\mathbf{y}$, we use $\langle \mathbf{x}, \mathbf{y}\rangle$ to denote their inner product. Let $\mathbf{A}$ be a positive semi-definite $d \times d$ matrix and $\boldsymbol{\nu} \in \mathbb{R}^d$. The weighted 2-norm of $\boldsymbol{\nu}$ with respect to $\mathbf{A}$ is defined by $\|\boldsymbol{\nu}\|_{\mathbf{A}} = \sqrt{\boldsymbol{\nu}^\top \mathbf{A}\boldsymbol{\nu}}$. We denote the minimum and maximum eigenvalue of $\mathbf{A}$ by $\lambda_{\min}(\mathbf{A})$ and $\lambda_{\max}(\mathbf{A})$. For a positive integer n, $[n]$ denotes the set $\{1, 2, \ldots, n\}$, while for positive integers $m \leq n$. We use $\mathbf{e}_i$ to denote the i-th standard basis vector. Finally, we use standard $\tilde{O}$ notation for big-O notation that ignores logarithmic factors. For two sequences $\{f(n)\}$ and $\{g(n)\}$, $f(n) = O(g(n))$ denotes that there exists a constant $c > 0$ such that $|f(n)| \leq c|g(n)|$. Additionally, $f(n) = \Omega(g(n))$ means that there exists a constant $c > 0$ such that $|f(n)| \geq c|g(n)|$. Let $f(x)$ and $g(x)$ be functions such that $f(x) \asymp g(x)$ as $x \to \infty$. This indicates that $f(x)$ and $g(x)$ have the same asymptotic growth rate.

1.2 Graph Neural Networks

Graph Neural Networks (GNNs) are a class of deep learning models designed to operate on graph-structured data. They extend traditional neural networks to handle non-Euclidean data representations, making them particularly useful for tasks involving relational and interconnected information.

1.2.1 Graph Representation

A graph G is typically defined as a pair (V, E), where V is the set of nodes (or vertices) and E is the set of edges connecting the nodes. Each node $v_i \in V$ can have associated features, represented as a vector $\mathbf{x}_i$. Similarly, edges can have features $\mathbf{e}_{ij}$ for an edge connecting nodes i and j.

1.2.2 Message Passing

The core idea of GNNs is the message passing mechanism, where nodes collect information from their neighbors. A general formulation of message passing can be expressed as:

$$\mathbf{h}_i^{(l+1)} = \phi\left(\mathbf{h}_i^{(l)}, \mathcal{A}\left(\left\{\psi(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}, \mathbf{e}_{ij}) : j \in \mathcal{N}(i)\right\}\right)\right). \tag{1.1}$$

Here, $\mathbf{h}_i^{(l)}$ is the feature vector of node i at layer l, $\mathcal{N}(i)$ is the set of neighbors of node i, ψ is a function for message passing, $\mathcal{A}$ is an aggregation function (e.g., sum, mean, max), and ϕ is an update function.

1.2.3 Graph Convolution

A simple and popular form of GNN is the Graph Convolutional Network (GCN). The layer-wise propagation rule in a GCN can be written as:

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}). \tag{1.2}$$

In this equation, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with self-loops, $\tilde{\mathbf{D}}$ is the degree matrix of $\tilde{\mathbf{A}}$, $\mathbf{H}^{(l)}$ is the matrix of node features at layer l, $\mathbf{W}^{(l)}$ is a learnable weight matrix, and σ is a non-linear activation function.

This formulation allows GNNs to learn representations that capture both node features and graph structure, enabling a wide range of applications in areas such as social network analysis, recommendation systems, and molecular property prediction.

1.3 Stochastic Linear Bandit

In a stochastic linear bandit setting, at each round t, the agent is presented with a decision set $\mathcal{D}_t$, which is a subset of the d-dimensional real space $\mathbb{R}^d$. The agent then selects an action $\mathbf{x}_t$ from the decision set $\mathcal{D}_t$. After choosing the action $\mathbf{x}_t$, the agent observes a reward y_t, which is derived based on an unknown vector $\boldsymbol{\theta}_* \in \mathbb{R}^d$ and the chosen action $\mathbf{x}_t$, plus some random additive noise η_t.

Mathematically, the reward y_t is expressed as: $y_t = \langle \boldsymbol{\theta}_*, \mathbf{x}_t \rangle + \eta_t$. Here, $\boldsymbol{\theta}_* \in \mathbb{R}^d$ is an unknown vector, and η_t represents the random additive noise. Let T be the total number of rounds played. We define the cumulative regret of the entire process as:

$$R_T := \sum_{t=1}^{T} \langle \boldsymbol{\theta}_*, \mathbf{x}_* \rangle - \langle \boldsymbol{\theta}_*, \mathbf{x}_t \rangle .$$

The optimal action $\mathbf{x}_*$ is defined for $\mathcal{D}_t$ as $\arg\max_{\mathbf{x} \in \mathcal{D}_t} \langle \boldsymbol{\theta}_*, \mathbf{x} \rangle$. The goal is to minimize the cumulative regret and achieve regret that is sublinear in T.

1.4 Finite-horizon Markov Decision Process

A finite-horizon Markov decision process (MDP) is represented as $M = (\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$, where $\mathcal{S}$ denotes the set of states, $\mathcal{A}$ denotes the set of actions, H represents the duration of each episode (horizon), $\mathbb{P} = \{\mathbb{P}_h\}_{h=1}^{H}$ indicates the transition probabilities, and $r = \{r_h\}_{h=1}^{H}$ denotes the reward functions. For each time-step $h \in [H]$, $\mathbb{P}_h(s' \mid s, a)$ signifies the probability of moving to state s' upon taking action a from state s, and $r_h : \mathcal{S} \times \mathcal{A} \to [0, 1]$ represents the reward function. $\mathcal{S}$ and $\mathcal{A}$ are known, but the transition probabilities $\mathbb{P}_h$ and rewards r_h are unknown to the agent and must be discovered through interaction. The agent engages with the unknown environment described by M over multiple episodes. Specifically, at each episode k and time-step $h \in [H]$, the agent observes the state s_h^k, selects an action $a_h^k \in \mathcal{A}$, and receives a reward $r_h^k := r_h\left(s_h^k, a_h^k\right)$.

A policy is a function $\pi : \mathcal{S} \times [H] \to \mathcal{A}$, where $\pi(s, h)$ specifies the action that the policy π prescribes for the agent to take at time-step $h \in [H]$ when in state $s \in \mathcal{S}$. A randomized policy

$\pi : \mathcal{S} \times [H] \to \Delta_{\mathcal{A}}$ maps states and time-steps to probability distributions over actions, such that $a \sim \pi(s,h)$ represents the action that the policy π recommends for the agent to take at time-step $h \in [H]$ when in state $s \in \mathcal{S}$.

The cumulative expected reward achieved under a policy π during and following time-step $h \in [H]$, referred to as the value function $V_h^\pi : \mathcal{S} \to \mathbb{R}$, is defined by

$$V_h^\pi(s) := \mathbb{E}\left[\sum_{h'=h}^{H} r_{h'}\left(s_{h'}, \pi\left(s_{h'}, h'\right)\right) \mid s_h = s\right].$$

We define the state-action value action $Q_h^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ for a policy π at time-step $h \in [H]$ as

$$Q_h^\pi(s,a) := \mathbb{E}\left[\sum_{h'=h+1}^{H} r_{h'}\left(s_{h'}, \pi\left(s_{h'}, h'\right)\right) \mid s_h = s, a_h = a\right].$$

For the function f, we define $\left[\mathbb{P}_h f\right](s,a) := \mathbb{E}_{s' \sim \mathbb{P}_h(\cdot \mid s,a)} f\left(s'\right)$. Let π_* denote the optimal policy, for which $V_h^{\pi_*}(s) := V_h^*(s) = \sup_\pi V_h^\pi(s)$ holds for every $(s,h) \in \mathcal{S} \times [H]$. Therefore, for all $(s,a,h) \in \mathcal{S} \times \mathcal{A} \times [H]$, the Bellman equations for a deterministic policy π and for the optimal deterministic policy are:

$$\begin{aligned} Q_h^\pi(s,a) &= r_h(s,a) + \left[\mathbb{P}_h V_{h+1}^\pi\right](s,a), \quad V_h^\pi(s) = Q_h^\pi(s, \pi(s,h)), \\ Q_h^*(s,a) &= r_h(s,a) + \left[\mathbb{P}_h V_{h+1}^*\right](s,a), \quad V_h^*(s) = \max_{a \in \mathcal{A}} Q_h^*(s,a), \end{aligned}$$

where $V_{H+1}^\pi(s) = V_{H+1}^*(s) = 0$.

The Bellman equations for a randomized policy π and the optimal randomized policy are as follows:

$$\begin{aligned} \tilde{Q}_h^\pi(s,a) &= r_h(s,a) + \left[\mathbb{P}_h \tilde{V}_{h+1}^\pi\right](s,a), \quad \tilde{V}_h^\pi(s) = \mathbb{E}_{a \sim \pi(s,h)}\left[\tilde{Q}_h^\pi(s,a)\right], \\ \tilde{Q}_h^*(s,a) &= r_h(s,a) + \left[\mathbb{P}_h \tilde{V}_{h+1}^*\right](s,a), \quad \tilde{V}_h^*(s) = \max_\theta \mathbb{E}_{a \sim \theta}\left[\tilde{Q}_h^*(s,a)\right]. \end{aligned}$$

CHAPTER 2

Efficient Sampling for Graph Neural Network Training

2.1 Introduction

Many real-world data come naturally in the form of graphs; e.g., social networks, gene expression networks, and knowledge graphs. In recent years, Graph Neural Networks (GNNs) [6, 7, 8] have been proposed to learn from such graph-structured data and have achieved outstanding performance. Yet, in many applications, graphs are usually large, containing hundreds of millions to billions of nodes and tens to hundreds of billions of edges. Learning on such giant graphs is challenging due to the limited memory available on a single GPU or machine. As such, mini-batch training is developed to train GNN models on such giant graphs. However, due to the connectivities between nodes, computing node embeddings with multi-layer GNNs usually involves many nodes in a mini-batch. This leads to substantial computation and data movement between CPUs and GPUs and makes training inefficient.

To remedy this issue, various GNN sampling methods have been developed to reduce the number of nodes in a mini-batch [8, 9, 10, 11, 12]. Node-wise neighbor sampling used by GraphSage [8] samples a fixed number of neighbors for each node independently. Even though it reduces the number of neighbors in a mini-batch, the number of nodes in a layer still grows exponentially. FastGCN [10] and LADIES [11] sample a fixed number of nodes in each layer, which results in isolated nodes when used on large graphs. In addition, LADIES requires significantly more computation on sampling and potentially slows down the overall training speed. The work by Liu et al. [12] alleviates the neighborhood explosion and reduces the sampling variance by applying a bandit sampler. However, this method leads to a very large sampling overhead and does not scale to large graphs. These sampling methods are usually evaluated on small to medium-sized graphs. When applying them to industry-scale graphs, they have suboptimal performance or substantial computation overhead as we discovered in our experiments.

To address some of these problems and reduce training time, LazyGCN [13] periodically samples a *mega-batch* of nodes and edges and reuses it to sample further mini-batches. By loading each mega-batch in GPU memory once, LazyGCN can mitigate data movement/preparation overheads. However, LazyGCN requires very large mega-batches (cf. Figure 2.4) to match the accuracy of standard GNN training, which makes it impractical for graphs with hundreds of millions of nodes.

We design an efficient and scalable sampling method that takes into account the characteristics of training hardware into consideration. GPUs are the most efficient hardware for training GNN models. Due to the small GPU memory size, state-of-the-art GNN frameworks, such as DGL [14] and Pytorch-Geometric [15], keep the graph data in CPU memory and perform mini-batch computations on GPUs when training GNN models on large graphs. We refer to this training strategy as *mixed CPU-GPU training*. The main bottleneck of mixed CPU-GPU training is data copy between CPUs and GPUs (cf. Figure 2.1). Because mini-batch sampling occurs in the CPU, we need a low-overhead sampling algorithm to enable efficient training. Motivated by the hardware characteristics, we developed the *Global Neighborhood Sampling* (GNS) approach that samples a global set of nodes periodically for all mini-batches. The sampled set is small so that we can store all of their node features in GPU memory and we refer to this set of nodes as *cache*. The cache is used for neighbor sampling in a mini-batch. Instead of sampling and neighbors of a node, GNS gives the priorities of sampling neighbors that exist in the cache. This is a fast way of biasing node-wise neighbor sampling to reduce the number of distinct nodes of each mini-batch and increase the overlap between mini-batches. When coupled with GPU cache, this method drastically reduces the amount of data copied between GPU and CPU to speed up training. In addition, we deploy importance sampling that reduces the sampling variance and also allows us to use a small cache size to train models.

We develop a highly optimized implementation of GNS and compare it with efficient implementations of other sampling methods provided by DGL, including node-wise neighbor sampling and LADIES. We show that GNS achieves state-of-the-art model accuracy while speeding up training by a factor of $2\times - 4\times$ compared with node-wise sampling and by a factor of $2\times - 14\times$ compared with LADIES.

The main contributions of the work are described below:

1. We analyze the existing sampling methods and demonstrate their main drawbacks on large graphs.
2. We develop an efficient and scalable sampling algorithm that addresses the main overhead in mixed CPU-GPU mini-batch training and show a substantial training speedup compared with efficient implementations of other training methods.
3. We demonstrate that this sampling algorithm can train GNN models on graphs with over 111 million nodes and 1.6 billion edges.

2.2 Background

In this section, we review GNNs and several state-of-the-art sampling-based training algorithms, including node-wise neighbor sampling methods and layer-wise importance sampling methods. The fundamental concepts of mixed-CPU-GPU training architecture are introduced. We discuss the limitations of state-of-the-art sampling methods in mixed CPU-GPU training.

2.2.1 Existing GNN Training Algorithms

Full-batch GNN Given a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, the input feature of node $v \in \mathcal{V}$ is denoted as $\mathbf{h}_v^{(0)}$, and the feature of the edge between node v and u is represented as $\mathbf{w}_{uv}$. The representation of node $v \in \mathcal{V}$

at layer ℓ can be derived from a GNN model given by:

$$\mathbf{h}_v^\ell = g(\mathbf{h}_v^{\ell-1}, \bigcup_{u\in\mathcal{N}(v)} f(\mathbf{h}_u^{\ell-1}, \mathbf{h}_v^{\ell-1}, \mathbf{w}_{uv})), \tag{2.1}$$

where f, $\bigcup$, and g are pre-defined or parameterized functions for computing feature data, aggregating data information, and updating node representations, respectively. For instance, in GraphSage training [8], the candidate aggregator functions include mean aggregator [6], LSTM aggregator [16], and max pooling aggregator [17]. The function g is set as a nonlinear activation function.

Given training dataset $\{(\mathbf{x}_i, y_i)\}_{v_i\in\mathcal{V}_s}$, the parameterized functions will be learned by minimizing the loss function:

$$\mathcal{L} = \frac{1}{|\mathcal{V}_s|}\sum_{v_i\in\mathcal{V}_s} \ell(y_i, \mathbf{z}_i^L), \tag{2.2}$$

where $\ell(\cdot,\cdot)$ is a loss function, $\mathbf{z}_i^L$ is the output of GNN with respect to the node $v_i \in \mathcal{V}_s$ where $\mathcal{V}_S$ represents the set of training nodes. For full-batch optimization, the loss function is optimized by gradient descent algorithm where the gradient for each node $v_i \in \mathcal{V}_S$ is computed as $\frac{1}{|\mathcal{V}_s|}\sum_{v_i\in\mathcal{V}_S} \nabla\ell(y_i, \mathbf{z}_i^L)$. During the training process, full-batch GNN must store and aggregate all nodes' representations across all layers. The expensive computation time and memory costs prohibit full-batch GNN from handling large graphs. Additionally, the convergence rate of full-batch GNN is slow because model parameters are updated only once at each epoch.

Mini-batch GNN To address this issue, a mini-batch training scheme has been developed which optimizes via mini-batch stochastic gradient descent $\frac{1}{|\mathcal{V}_B|}\sum_{v_i\in\mathcal{V}_B} \nabla\ell(y_i, \mathbf{z}_i^L)$ where $\mathcal{V}_B \in \mathcal{V}_S$. These methods first uniformly sample a set of nodes from the training set, known as *target nodes*, and sample neighbors of these target nodes to form a mini-batch. The mini-batch training methods focus on reducing the number of neighbor nodes for aggregation via various sampling strategies to reduce the memory and computational cost. The state-of-the-art sampling algorithm is discussed in the sequel.

Node-wise Neighbor Sampling Algorithms. Hamilton et al. [8] proposed an unbiased sampling method to reduce the number of neighbors for aggregation via neighbor sampling. It randomly selects at most s_{node} (defined as *fan-out* parameter) neighborhood nodes for every target node; followed by computing the representations of target nodes via aggregating feature data from the sampled neighborhood nodes. Based on the notations in (2.1), the representation of node $v \in \mathcal{V}$ at layer ℓ can be described as follows:

$$\mathbf{h}_v^\ell = g\bigg(\mathbf{h}_v^{\ell-1}, \bigcup_{u \in \mathcal{N}_\ell(v)} f\big(\frac{1}{s_{\text{node}}}\mathbf{h}_u^{\ell-1}, \mathbf{h}_v^{\ell-1}\big)\bigg), \tag{2.3}$$

where $\mathcal{N}_\ell(v)$ is the sampled neighborhood nodes set at ℓ-th layer such that $|\mathcal{N}_\ell(v)| = s_{\text{node}}$. The neighbor sampling procedure is repeated recursively on target nodes and their sampled neighbors when dealing with multiple-layer GNN. Even though the node-wise neighbor sampling scheme addresses the memory issue of GNN, there exists excessive computation under this scheme because the scheme still results in the exponential growth of neighbor nodes with the number of layers. This yields a large volume of data movement between CPU and GPU for mixed CPU-GPU training.

Layer-wise Importance Sampling Algorithms. To address the scalability issue, Chen et al. [10] proposed an advanced layer-wise method called FastGCN. Compared with the node-wise sampling method, it yields extra variance when sampling a fixed number of nodes for each layer. To address the variance issue, it performs degree-based importance sampling on each layer. The representation of node $v \in \mathcal{V}$ at layer ℓ of FastGCN model is described as follows:

$$\mathbf{h}_v^\ell = g\bigg(\mathbf{h}_v^{\ell-1}, \bigcup_{q_u \in q(v)} f\big(\frac{1}{s_{\text{layer}}}\mathbf{h}_u^{\ell-1}/q_u, \mathbf{h}_v^{\ell-1}\big)\bigg), \tag{2.4}$$

where the sample size denotes as s_{layer}, $q(v)$ is the distribution over $v \in \mathcal{V}$ and q_u is the probability assigned to node u. A major limitation is that FastGCN performs sampling on every layer independently, which yields approximate embeddings with large variances. Moreover, the subgraph sampled by FastGCN is not representative of the original graph. This leads to poor performance and the number of sampled nodes required to guarantee convergence during the training process is large.

The work by Zhou et al. [11] proposed a sampling algorithm known as LAyer-Dependent Importance Sampling (LADIES) to address the limitation of FastGCN and exploit the connection between different layers. Specifically, at ℓ-th layer, LADIES samples nodes reachable from the nodes in the previous layer. However, this method [11] comes with a cost. To ensure node connectivity between layers, the method needs to extract and merge the entire neighborhood of all nodes in the previous layer and compute the sampling probability for all candidate nodes in the next layer. Thus, this sampling method has a significantly higher computation overhead. Furthermore, when applying this method on a large graph, it still constructs a mini-batch with many isolated nodes, especially for nodes in the first layer (Table 2.5).

LazyGCN. Even though layer-wise sampling methods effectively address the neighborhood explosion issue, they failed to investigate computational overheads in preprocessing data and loading fresh samples during training. Ramezan et al. [13] proposed a framework called LazyGCN which decouples the frequency of sampling from the sampling strategy. It periodically samples *mega-batches* and effectively reuses *mega-batches* to generate mini-batches and alleviate the preprocessing overhead. There are some limitations in the LazyGCN setting. First, this method requires large mini-batches to guarantee model accuracy. For example, their experiments on Yelp and Amazon datasets use a batch size of 65,536. This batch size is close to the entire training set, yielding overwhelming overhead in a single mini-batch computation. Its performance deteriorates for smaller batch sizes even with sufficient epochs (Figure 2.4). Second, their evaluation is based on inefficient implementations with very large sampling overhead. In practice, the sampling computation overhead is relatively low in the entire mini-batch computation (Figure 2.1) when using proper development tools.

Even though both GNS and LazyGCN cache data in GPU to accelerate computation in mixed CPU-GPU training, they use very different strategies for caching. LazyGCN caches the entire graph structure of multiple mini-batches sampled by node-wise neighbor sampling or layer-wise sampling and suffers from the problems in these two algorithms. Due to the exponential growth of the neighborhood size in node-wise neighbor sampling, LazyGCN cannot store a very large mega-batch in GPU and can generate a few mini-batches from the mega-batch. Our experiments show that LazyGCN runs out of GPU memory even with a small mega-batch size and mini-batch size on large graphs (OAG-paper and OGBN-papers100M in Table 2.2). Layer-wise sampling may result in many isolated nodes in a mini-batch. In addition, LazyGCN uses the same sampled graph structure when generating mini-batches from mega-batches, this potentially leads to overfitting. In contrast, GNS cache nodes and use the cache to reduce the number of nodes in a mini-batch; GNS always sample a different graph structure for each mini-batch and thus it is less likely to overfit.

2.2.2 Mixed CPU-GPU Training

Due to limited GPU memory, state-of-the-art GNN frameworks (e.g., DGL [14] and Pytorch Geometric [15]) train GNN models on large graph data by storing the whole graph data in CPU memory and performing mini-batch computation on GPUs. This allows users to take advantage of large CPU memory and use GPUs to accelerate GNN training. In addition, mixed CPU-GPU training makes it easy to scale GNN training to multiple GPUs or multiple machines [18].

A mixed CPU-GPU training strategy usually involves six steps: *1)* sample a mini-batch from the full graph, *2)* slice the node and edge data involved in the mini-batch from the full graph, *3)* copy the above-sliced data to GPU, *4)* perform forward computation on the mini-batch, *5)* perform backward propagation, and *6)* run the optimizer and update model parameters. Steps 1–2 are done by the CPU, whereas steps 4–6 are done by the GPU.

We benchmark the mini-batch training of GraphSage [8] models with node-wise neighbor sampling provided by DGL, which provides very efficient neighbor sampling implementation and graph kernel computation for GraphSage. Figure 2.1 shows the breakdown of the time required to train GraphSage on the OGBN-products graph and the OAG-paper graph (see Table 2.2 for dataset information). Even though sampling happens in the CPU, its computation accounts for 10% or less with sufficient optimization and parallelization. However, the speed of copying node data in the CPU (step 2) is limited by the CPU memory bandwidth, and moving data to the GPU (step 3) is limited by the PCIe bandwidth. Data copying accounts for most of the time required by mini-batch training. For example, the training spends 60% and 80% of the per mini-batch time in copying data from CPU to GPU on OGBN-products and OAG-paper, respectively. The training on OAG-paper takes significantly more time on data copy because OAG-paper has 768-dimensional

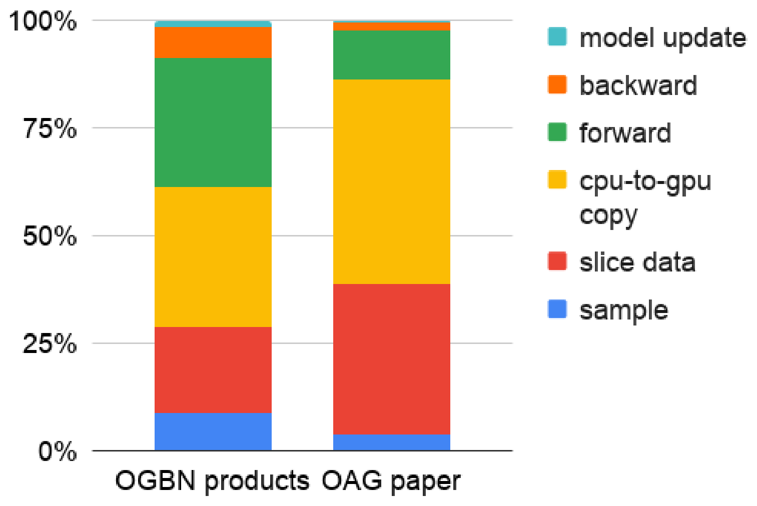


Figure 2.1: Runtime breakdown (%) of each component in mini-batch training for an efficient GraphSage implementation in DGL.

BERT embeddings [19] as node features, whereas OGBN-products use 100-dimensional node features.

These results show that when the different components of mini-batch training are highly optimized, the main bottleneck of mixed CPU-GPU training is data copy (both data copy in CPU and between CPU and GPUs). To speed up training, it is essential to reduce the overhead of data copy, without significantly increasing the overhead of other steps.

2.3 Global Neighbor Sampling (GNS)

To overcome the drawbacks of the existing sampling algorithms and tackle the unique problems in mixed CPU-GPU training, we developed a new sampling approach, called *Global Neighborhood Sampling* (GNS), that has low computational overhead and reduces the number of nodes in a mini-batch without compromising the model accuracy and convergence rate. Like node-wise and layer-wise sampling, GNS uses mini-batch training to approximate the full-batch GNN training on giant graphs.

Table 2.1: Summary of notations and definitions.

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	$\mathcal{G}$ denotes the graph consist of set of $\lvert\mathcal{V}\rvert$ nodes and $\lvert\mathcal{E}\rvert$ edges.
L, K	L is the total number of layers in GCN, and K is the dimension of embedding vectors (for simplicity, assume it is the same across all layers).
$b, s_{\text{node}}, s_{\text{layer}}$	For batch-wise sampling, b denotes the batch size, s_{node} is the number of sampled neighbors per node for node-wise sampling, and s_{layer} is the number of sampled nodes per layer for layer-wise sampling.
$\mathcal{N}(v)$	Denotes the set of neighbors of node $v \in \mathcal{V}$.
$\mathcal{N}_\ell(v)$	Denotes the set of sampled neighbors of node $v \in \mathcal{V}$ at ℓ-th layer.
$\mathcal{N}_C(v)$	Denotes the set of neighbors of node $v \in \mathcal{V}$ in the cache.
$\mathcal{C}, p_v^{\text{cache}}$	Denotes the set of cached nodes which are sampled from $\mathcal{V}$ corresponding to the probability of p_v^{cache} for $v \in \mathcal{V}$.
$p_v^{(\ell)}$	Denotes importance sampling coefficients with respect to the node $v \in \mathcal{V}$ at ℓ-th layer in Algorithm 1.
$\mathcal{V}_S$, $\lvert\mathcal{V}_S\rvert$	Denotes the training set and the size of the training set.
target node	The node in the mini-batch where the mini-batch is sampled at random from the training node-set.
$\mathcal{V}_B$, $\lvert\mathcal{V}_B\rvert$	Denotes the set of target nodes and the number of target nodes in a mini-batch.

2.3.1 Overview of GNS

Instead of sampling neighbors independently like node-wise neighbor sampling, GNS periodically samples a global set of nodes following a probability distribution $\mathcal{P}$ to assist in neighbor sampling.

$\mathcal{P}_i$ defines the probability of node i in the graph being sampled and placed in the set. Because GNS only samples a small number of nodes to form the set, we can copy all of the node features in the set to GPUs. Thus, we refer to the set of nodes as *node cache* $\mathcal{C}$. When sampling neighbors of a node, GNS prioritizes the sampled neighbors from the cache and samples additional neighbors outside the cache only if the cache does not provide sufficient neighbors.

Because the nodes in the cache are sampled, we can compute the node sampling probability from the probability of a node appearing in the cache, i.e., $\mathcal{P}$. We rescale neighbor embeddings by importance sampling coefficients $p_u^{(\ell)}$ from $\mathcal{P}$ in the mini-batch forward propagation so that the expectation of the aggregation of sampled neighbors is the same as the aggregation of the full neighborhood.

$$\mathbb{E}\left(\sum_{u \in \mathcal{N}_\ell(v)} p_u^{(\ell)} * h_u^\ell\right) = \sum_{u \in \mathcal{N}(v)} h_u^\ell \tag{2.5}$$

Algorithm 1 illustrates the entire training process.

In the remaining sections, we first discuss the cache sampling in Section 2.3.2. We explain the sampling procedure in Section 2.3.3. To reduce the variance in GNS, an importance sampling scheme is further developed in Section 2.3.4. We then establish the convergence rate of GNS which is inspired by the paper [13]. It shows that under the mild assumption, GNS enjoys a comparable convergence rate as underlying node-wise sampling in training, which is demonstrated in Section 2.3.5. The notations and definitions used in the following are summarized in Table 2.1.

2.3.2 Sample Cache

GNS periodically constructs a cache of nodes $\mathcal{C}$ to facilitate neighbor sampling in mini-batch construction. GNS uses a biased sampling approach to select a set of nodes C that, with high probability, can be reached from nodes in the training set. The features of the nodes in the cache are loaded into GPUs beforehand.

Ideally, the cache needs to meet two requirements: 1) to keep the entire cache in the GPU

Algorithm 1: Minibatch Training with GNS

```
Input  : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$;
         list of target nodes of mini-batches $\{\mathcal{B}_1, \cdots, \mathcal{B}_M\}$;
         input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$;
         number of epochs $T$;
         depth $L$; weight matrices $\mathbf{W}^\ell, \forall \ell \in \{1, ..., L\}$;
         cache sampling probability $\mathcal{P}$;
         nonlinear activation function $g$;
         differentiable aggregator functions $f_\ell, \forall \ell \in \{1, ..., L\}$.
Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{B}$
1:  for $t = 0$ to $T$ do
2:    $\mathcal{C} \leftarrow sample\_cache(\mathcal{V}, \mathcal{P}, \{\mathcal{B}_1, \cdots, \mathcal{B}_M\})$
3:    for $\mathcal{B} \in \{\mathcal{B}_1, \cdots, \mathcal{B}_M\}$ do
4:      $\mathcal{B}^L \leftarrow \mathcal{B}$
5:      for $\ell = L...1$ do
6:        $B^{\ell-1} \leftarrow \{\}$
7:        for $u \in \mathcal{B}^\ell$ do
8:          $\mathcal{N}_\ell(u), \mathcal{P}_\ell(u) \leftarrow sample(\mathcal{N}(u), C)$
9:          $\mathcal{B}^{\ell-1} \leftarrow \mathcal{B}^{\ell-1} \cup \mathcal{N}_\ell(u)$;
10:         $\mathcal{P}^{\ell-1} \leftarrow \mathcal{P}^{\ell-1} \cup \mathcal{P}_\ell(u)$
11:       end for
12:     end for
13:     $\mathbf{h}_u^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{B}^0$
14:     for $\ell = 1...L$ do
15:       for $u \in \mathcal{B}^\ell$ do
16:         Compute importance sampling coefficients
              $p_{u'}^{(\ell-1)}$ for $\forall u' \in \mathcal{N}_\ell(u)\}$
17:         $\mathbf{h}_{\mathcal{N}(u)}^\ell \leftarrow f_\ell(\{p_{u'}^{(\ell-1)} \mathbf{h}_{u'}^{\ell-1}, \forall u' \in \mathcal{N}_\ell(u)\})$
18:         $\mathbf{h}_u^\ell \leftarrow g\left(\mathbf{W}^\ell \cdot (\mathbf{h}_u^{\ell-1}, \mathbf{h}_{\mathcal{N}(u)}^\ell)\right)$
19:       end for
20:     end for
21:   end for
22: end for
```

1

memory, the cache has to be sufficiently small; 2) to have sampled neighbors come from the cache, the nodes in the cache have to be reachable from the nodes in the training set with a high probability.

Potentially, we can uniformly sample nodes to form the cache, which may require a large number of nodes to meet requirement number two. Therefore, we deploy two approaches to define the sampling probability for the cache. If the majority of the nodes in a graph are in the training set, we define the sampling probability based on node degree. For node i, the probability of being sampled in the cache is given by

$$p_i = \deg(i) / \sum_{k \in \mathcal{V}} \deg(k). \tag{2.6}$$

For a power-law graph, we only need to maintain a small cache of nodes to cover the majority of the nodes in the graph.

If the training set only accounts for a small portion of the nodes in the graph, we use short random walks to compute the sampling probability. Define $\mathcal{N}_\ell(v)$ as the number of sampled neighbor nodes corresponding to node $v \in \mathcal{V}$ in each layer,

$$\mathbf{d} = [\mathcal{N}_\ell(v_1)/\deg(v_1), \cdots, \mathcal{N}_\ell(v_{|\mathcal{V}|})/\deg(v_{|\mathcal{V}|})]^\top. \tag{2.7}$$

The node sampling probability $\boldsymbol{P}^\ell \in \mathbb{R}^{|\mathcal{V}|}$ for the ℓ-th layer is represented as

$$\boldsymbol{P}^\ell = (\mathbf{DA} + \mathbf{I})\boldsymbol{P}^{\ell-1}, \tag{2.8}$$

where $\mathbf{A}$ is the adjacency matrix and $\mathbf{D} = \mathrm{diag}(\mathbf{d})$. $\boldsymbol{P}^0$ is

$$p_i^0 = \begin{cases} \frac{1}{|\mathcal{V}_S|}, & \text{if } i \in \mathcal{V}_S \\ 0, & \text{otherwise.} \end{cases} \tag{2.9}$$

The sampling probability for the cache is set as $\boldsymbol{P}^L$, where L is the number of layers in the multi-layer GNN model.

As the experiments will later show (cf. Section 2.4), the size of the cache $\mathcal{C}$ can be as small as 1% of the number of nodes ($|\mathcal{V}|$) without compromising the model accuracy and convergence rate.

2.3.3 Sample Neighbors with Cache

When sampling k neighbors for a node, GNS first restricts sampled neighbor nodes from the cache $\mathcal{C}$. If the number of neighbors sampled from the cache is less than k, it samples the remaining neighbors uniformly at random from its neighborhood.

A simple way of sampling neighbors from the cache is to compute the overlap of the neighbor list of a node with the nodes in the cache. Assuming one lookup in the cache has $O(1)$ complexity, this algorithm will result in $O(|\mathcal{E}|)$ complexity, where $|\mathcal{E}|$ is the number of edges in the graph. However, this complexity is significantly larger than the original node-wise neighbor sampling $O(\sum_{i\in\mathcal{V}_B}\min(k,|\mathcal{N}(i)|))$ in a power-law graph, where $\mathcal{V}_B$ is the set of target nodes in a mini-batch and $|\mathcal{N}(i)|$ is the number of neighbors of node i. Instead, we construct an induced subgraph $\mathcal{S}$ that contains the nodes in the cache and their neighbor nodes. This is done once, right after we sample nodes in the cache. For an undirected graph, this subgraph contains the neighbors of all nodes that reside in the cache. During neighbor sampling, we can get the cached neighbors of node i by reading the neighborhood $\mathcal{N}_S(i)$ of node i in the subgraph. Constructing the subgraph $\mathcal{S}$ is much more lightweight, usually $\ll O(|\mathcal{E}|)$.

We parallelize the sampling computations with multiprocessing. That is, we create a set of processes to sample mini-batches independently and send them back to the trainer process for mini-batch computation. The construction of subgraphs $\mathcal{S}$ for multiple caches $\mathcal{C}$ can be parallelized.

2.3.4 Importance Sampling Coefficient

The nodes in the cache are sampled non-uniformly at random. So are the neighbors of a node. To approximate the expectation of the uniform sampling method, we assign importance weights to the nodes, thereby rescaling the neighbor features for aggregation:

$$\mathbf{h}^{\ell}_{\mathcal{N}(u)} \leftarrow f_{\ell}(\{1/p^{(\ell-1)}_{u'} \cdot \mathbf{h}^{\ell-1}_{u'}, \forall u' \in \mathcal{N}_{\ell}(u)\}). \tag{2.10}$$

To establish the importance sampling coefficient, we begin with computing the probability of the sampled node $u' \in \mathcal{N}_\ell(u)$ being contained in the cache, given by

$$p_{u'}^{\mathcal{C}} = 1 - (1 - p_{u'})^{|\mathcal{C}|}, \tag{2.11}$$

where the sampling probability $p_{u'}$ refers to (2.6) and $|\mathcal{C}|$ denotes the size of cache set. The importance sampling coefficient $p_{u'}^{(\ell-1)}$ can be represented as

$$p_{u'}^{(\ell-1)} = p_{u'}^{\mathcal{C}} \frac{k}{\max\{k, \mathcal{N}_C(i)\}}. \tag{2.12}$$

2.3.5 Theoretical Analysis

In this section, we establish the convergence rate of GNS which is inspired by the work of Ramezani et al. [13]. It shows that under the mild assumption, GNS enjoys a comparable convergence rate as underlying node-wise sampling in training. Here, the convergence rate of GNS mainly depends on the graph degree and the size of the cached set. We focus on a two-layer GCN for simplicity and denote the loss functions of full-batch, mini-batch, and proposed GNS as

$$J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i \in \mathcal{V}} f_i \left(\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \frac{1}{|\mathcal{N}(j)|} \sum_{k \in \mathcal{N}(j)} g_{jk}(\boldsymbol{\theta}) \right) \tag{2.13}$$

$$J_{\mathcal{B}}(\boldsymbol{\theta}) = \frac{1}{B} \sum_{i \in \mathcal{V}_{\mathcal{B}}} f_i \left(\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \frac{1}{|\mathcal{N}(j)|} \sum_{k \in \mathcal{N}(j)} g_{jk}(\boldsymbol{\theta}) \right) \tag{2.14}$$

$$\begin{aligned} \widetilde{J}_{\mathcal{B}}(\boldsymbol{\theta}) = \\ \frac{1}{B} \sum_{i \in \mathcal{V}_{\mathcal{B}}} f_i \Bigg(\frac{1}{|\mathcal{N}_2^{\mathrm{u}}(i) \cap \mathcal{C}|} \sum_{j \in \mathcal{N}_2^{\mathrm{u}}(i) \cap \mathcal{C}} \frac{1}{|\mathcal{N}_1^{\mathrm{u}}(j) \cap \mathcal{C}|} \\ \sum_{k \in \mathcal{N}_1^{\mathrm{u}}(j) \cap \mathcal{C}} p_k^{(1)} g_{jk}(\boldsymbol{\theta}) \Bigg), \end{aligned} \tag{2.15}$$

respectively, where the outer and inner layer function are defined as $f(\cdot) \in \mathbb{R}$ and $g(\cdot) \in \mathbb{R}^n$, and their gradients as $\nabla f(\cdot) \in \mathbb{R}^n$ and $\nabla g(\cdot) \in \mathbb{R}^{n \times n}$, respectively. Specifically, the function $g_{jk}(\cdot)$

depends on the nodes contained in two layers. For simplicity, we denote $\mathcal{N}^{\mathcal{C}}(j) := \mathcal{N}_1^{\mathrm{u}}(j) \cap \mathcal{C}$. We denote $|\mathcal{N}(i)| = N^i$, $|\mathcal{N}_\ell^{\mathrm{u}}(i)| = N_\ell^i, \ell = 1, 2$ and $|\mathcal{N}^{\mathcal{C}}(j)| = N_{\mathcal{C}}^j$ in the following.

The following assumption gives the Lipschitz continuous constant of the gradient of the composite function $J(\theta)$, which plays a vital role in theoretical analysis.

Assumption 1 *Suppose $f(\cdot)$ is L_f-Lipschitz continuous, $g(\cdot)$ is L_g-Lipschitz continuous, $\nabla f(\cdot)$ is L'_f-Lipschitz continuous, $\nabla g(\cdot)$ is L'_g-Lipschitz continuous.*

Theorem 1 *Denote N_ℓ^i as the number of the neighborhood nodes for $i \in \mathcal{V}$ sampling uniformly at random at ℓ-th layer. The cached nodes in the set $\mathcal{C}$ with the size of $|\mathcal{C}|$ are sampled without replacement according to p_v^{cache}. The dimension of the node feature is denoted as n and the size of the mini-batch is denoted as B. Define $\widetilde{C} = |\mathcal{C}|/|\mathcal{V}|$ and $C_d = \sum_{v_i \in \mathcal{V}} \deg(v_i)/|\mathcal{V}|$ with the constant $c > 0$. Under Assumption 1, with probability exceeding $1 - \delta$, GNS optimized by stochastic gradient descent can achieve*

$$\mathbb{E}\left[\|\nabla J(\hat{\boldsymbol{\theta}})\|^2\right] \leq O\left(\sqrt{\frac{\mathrm{MSE}}{t}}\right), \tag{2.16}$$

where $\hat{\boldsymbol{\theta}} = \min_t \mathbb{E}\left[\|\nabla J\left(\boldsymbol{\theta}_t\right)\|\right]$ with $\boldsymbol{\theta}_t = \{\mathbf{W}_t^\ell\}_{\ell=1}^L$ and

$$\begin{aligned}\mathrm{MSE} \leq & O\left(L_f'^2 \frac{\log(4n/\delta) + 1/2}{B}\right) + O\left(L_f'^2 L_g^4 \frac{\log(4n/\delta) + 1/2}{c\widetilde{C} C_d N_1^j N_2^i}\right) \\ & + O\left(L_g'^2 L_f^2 \frac{\log(4n/\delta)}{c\widetilde{C} C_d N_1^j N_2^i}\right).\end{aligned} \tag{2.17}$$

Proof: *The details on the proof of Theorem 1 are provided in Appendix A.1.* □

Variance of GNS We aim to derive the average variance of the embedding for the output nodes at each layer. Before moving forward, we provide several useful definitions. Let B denote the size of the nodes in one layer. Consider the underlying embedding: $\mathbf{Z} = \mathbf{L}\mathbf{H}\boldsymbol{\Theta}$, where $\mathbf{L}$ is the Laplacian matrix, $\mathbf{H}$ denotes the feature matrix and $\boldsymbol{\Theta}$ is the weight matrix, Let $\tilde{\mathbf{Z}} \in \mathbb{R}^{B \times d}$ with the dimension of feature d denote the estimated embedding derived from the sample-based method.

Denote $\mathbf{P} \in \mathbb{R}^{B\times|\mathcal{V}|}$ as the row selection matrix which samples the embedding from the whole embedding matrix. The variance can be represented as $\mathbb{E}[\|\tilde{\mathbf{Z}} - \mathbf{PZ}\|_F]$. Denote $\mathbf{L}_{i,*}$ as the i-th row of matrix $\mathbf{L}, \mathbf{L}_{*,j}$ is the j -th column of matrix $\mathbf{L}$, and $\mathbf{L}_{i,j}$ is the element at the position (i, j) of matrix $\mathbf{L}$.

For each node at each layer, its embedding is estimated based on its neighborhood nodes established from the cached set $\mathcal{C}$. Based on the Assumption 1 in [11], we have

$$\begin{aligned}
&\mathbb{E}\left[\|\tilde{\mathbf{Z}} - \mathbf{PZ}\|_F^2\right] \\
=&\sum_{i=1}^{|\mathcal{V}|} q_i \cdot \mathbb{E}\left[\left\|\tilde{\mathbf{Z}}_{i,*} - \mathbf{Z}_{i,*}\right\|_2^2\right] \\
=&\sum_{i=1}^{|\mathcal{V}|} q_i \left\|\mathbf{L}_{i,*}\right\|_0 \left(\sum_{j=1}^{|\mathcal{V}|} p_{ij} \cdot s_j \left\|\mathbf{L}_{i,j}\mathbf{H}_{j,*}\mathbf{\Theta}\right\|_2^2 - \left\|\mathbf{L}_{i,*}\mathbf{H}\mathbf{\Theta}\right\|_F^2\right) \\
=&\sum_{i=1}^{|\mathcal{V}|} q_i \left\|\mathbf{L}_{i,*}\right\|_0 \big(\sum_{j=1}^{|\mathcal{V}|} p_{ij} \cdot s_j \left\|\mathbf{L}_{i,j}\mathbf{H}_{j,*}\mathbf{\Theta}\right\|_2^2 - \\
&q_i \cdot p_{ij} \cdot s_j \|\mathbf{L}\mathbf{H}\mathbf{\Theta}\|_F^2\big)
\end{aligned} \tag{2.18}$$

where q_i is the probability of node i being contained in the first layer via neighborhood sampling and p_{ij} is the importance sampling coefficient related to node i and j. Moreover, s_j is the probability of node i being in the cache set $\mathcal{C}$.

Under Assumption 1 and 2 in [11] such that $\left\|\mathbf{H}_{i,*}\mathbf{\Theta}\right\|_2 \leq \gamma$ for all $i \in [|\mathcal{V}|]$ and $\left\|\mathbf{L}_{i,*}\right\|_0 \leq \frac{C}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|} \left\|\mathbf{L}_{i,*}\right\|_0$ and the definition of the importance sampling coefficient, we arrive

$$\begin{aligned}
&\mathbb{E}\left[\|\tilde{\mathbf{Z}} - \mathbf{PZ}\|_F^2\right] \\
\leq&\sum_{i=1}^{|\mathcal{V}|} q_i \left\|\mathbf{L}_{i,*}\right\|_0 \sum_{j=1}^{|\mathcal{V}|} p_{ij} \cdot s_j \left\|\mathbf{L}_{i,j}\mathbf{H}_{j,*}\mathbf{\Theta}\right\|_2^2 \\
\leq&C \sum_{i=1}^{|\mathcal{V}|}\sum_{j=1}^{|\mathcal{V}|} q_i \cdot p_{ij} \cdot s_j \left\|\mathbf{L}_{i,j}\mathbf{H}_{j,*}\mathbf{\Theta}\right\|_2^2 \\
\leq&\frac{CB_{\text{out}}C_d\gamma\|\mathbf{L}\|_F^2}{|\mathcal{V}|},
\end{aligned} \tag{2.19}$$

where C_d denotes the average degree and B_{out} is the size of nodes at the output layer.

2.3.6 Summary and Discussion

GNS shares many advantages with various sampling methods and can avoid their drawbacks. Like node-wise neighbor sampling, it samples neighbors on each node independently and, thus, can be implemented and parallelized efficiently. Due to the cache, GNS tends to avoid the neighborhood explosion in multi-layer GNN. GNS maintains a global and static distribution to sample the cache, which requires only one-time computation and can be easily amortized during the training. In contrast, LADIES computes the sampling distribution for every layer in every mini-batch, which makes the sampling procedure expensive. Even though GNS constructs a mini-batch with more nodes than LADIES, forward and backward computation on a mini-batch is not the major bottleneck in many GNN models for mixed CPU-GPU training. Even though both GNS and LazyGCN deploy caching to accelerate computation in mixed CPU-GPU training, they use cache very differently. GNS uses a cache to reduce the number of nodes in a mini-batch to reduce computation and data movement between CPU and GPUs. It captures the majority of the connectivities of nodes in a graph. LazyGCN caches and reuses the sampled graph structure and node data. This requires a large mega-batch size to achieve good accuracy, which makes it difficult to scale to giant graphs. Because LazyGCN uses node-wise sampling or layer-wise sampling to sample mini-batches, it suffers from the problems inherent to these two sampling algorithms. For example, as shown in the experiment section, LazyGCN cannot construct a mega-batch with node-wise neighbor sampling on large graphs.

2.4 Experiments

2.4.1 Datasets and Setup

We evaluate the effectiveness of GNS under inductive supervise setting on the following real-world large-scale datasets: Yelp [9], and Amazon [9], OAG-paper [1], OGBN-products [20], OGBN-

[1]`https://s3.us-west-2.amazonaws.com/dgl-data/dataset/OAG/oag_max_paper.dgl`

papers100M [20]. OAG-paper is the paper citation graph in the medical domain extracted from the OAG graph [21]. On each of the datasets, the task is to predict the labels of the nodes in the graphs. Table 2.2 provides various statistics for these datasets.

Table 2.2: Dataset statistics.

Dataset	Nodes	Edges	Avg. Deg	Feature	Classes	Multiclass	Train / Val / Test
Yelp	716,847	6,977,410	10	300	100	Yes	0.75 / 0.10 / 0.15
Amazon	1,598,960	132,169,734	83	200	107	Yes	0.85 / 0.05 / 0.10
OAG-paper	15,257,994	220,126,508	14	768	146	Yes	0.43 / 0.05 / 0.05
OGBN-products	2,449,029	123,718,280	51	100	47	No	0.10 / 0.02 / 0.88
OGBN-Papers100M	111,059,956	3,231,371,744	30	128	172	No	0.01 / 0.001 / 0.002

For each trial, we run the algorithm with ten epochs on Yelp, Amazon OGBN-products, OGBN-Papers100M dataset, and each epoch proceeds for $\frac{\text{\# train set}}{\text{batch size}}$ iterations. For the OAG-paper dataset, we run the algorithm with three epochs. We compare GNS with node-wise neighbor sampling (used by GraphSage), LADIES, and LazyGCN for training 3-layer GraphSage. The detailed settings concerning these four methods are summarized as follows:

- **GNS**: GNS is implemented by DGL [14] and we apply GNS on all layers to sample neighbors. The sampling fan-outs of each layer are 15, 10 for the third and second layers. We sample nodes in the first layer (input layer) only from the cache. The size of the cached set is $1\% \cdot |\mathcal{V}|$.
- **Node-wise neighbor sampling (NS)** [2] [8]: NS is implemented by DGL [14]. The sampling fan-outs of each layer are 15, 10, and 5.
- **LADIES** [3] [11]: LADIES is implemented by DGL [14]. We sample 512 and 5000 nodes for LADIES per layer, respectively.

[2]`https://github.com/dmlc/dgl/tree/master/examples/pytorch/graphsage`

[3]`https://github.com/BarclayII/dgl/tree/ladies/examples/pytorch/ladies`

- **LazyGCN** [4] [13]: we use the implementation provided by the authors. We set the recycle period size as $R = 2$ and the recycling growth rate as $\rho = 1.1$. The sampler of LazyGCN is set as node wise sampling with 15 neighborhood nodes in each layer.

GNS, NS, and LADIES are parallelized with multiprocessing. For all methods, we use the batch size of 1000. We use two metrics to evaluate the effectiveness of sampling methods: micro F1-score to measure the accuracy and the average running time per epoch to measure the training speed.

We run all experiments on an AWS EC2 g4dn.16xlarge instance with 32 CPU cores, 256GB RAM, and one NVIDIA T4 GPU.

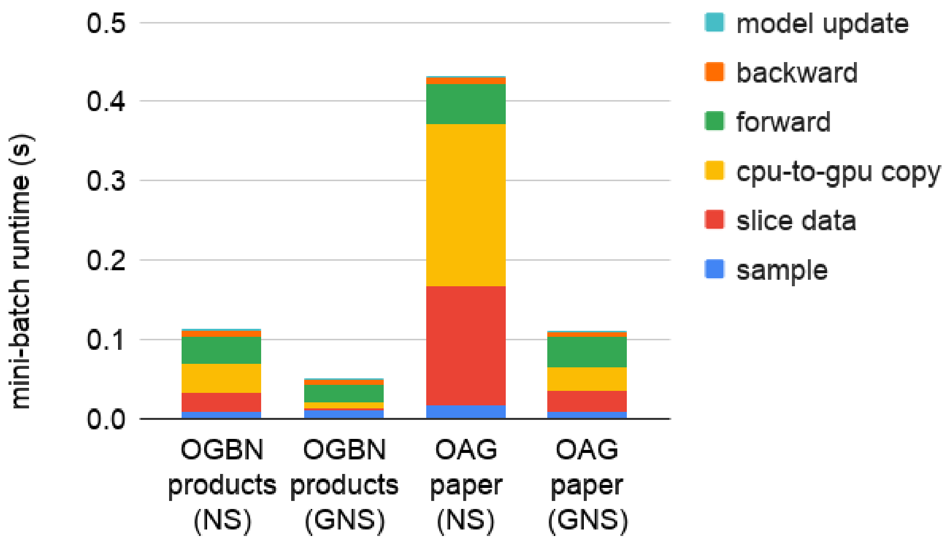


Figure 2.2: Runtime breakdown (s) of each component in mini-batch training of NS and GNS on OGBN-products and OAG-paper graphs.

2.4.2 Experiment Results

We evaluate the test F1-score and average running time per epoch by using different methods in the case of the large-scale dataset.

[4]`https://github.com/MortezaRamezani/lazygcn`

Table 2.3: Performance of different sampling approaches.

Dataset(hidden layer dimension)	Metric	NS	LADIES (512)	LADIES (5000)	LazyGCN	GNS
Yelp(512)	F1-Score(%)	62.54	59.32	61.04	35.58	63.20
	Time per epoch (s)	58.5	62.1	237.9	1248.7	23.1
Amazon(512)	F1-Score(%)	76.69	76.46	77.05	31.08	76.13
	Time per epoch (s)	89.5	613.4	3234.2	3280.2	42.8
OAG-paper(256)	F1-Score(%)	50.23	43.51	46.72	N/A	49.23
	Time per epoch (s)	3203.2	2108.0	7956.0		819.4
OGBN-products(256)	F1-Score(%)	78.44	70.32	75.36	69.78	78.01
	Time per epoch (s)	25.6	45.4	223.5	264.2	11.9
OGBN-Papers100M(256)	F1-Score(%)	63.61	57.94	59.23	N/A	63.31
	Time per epoch (s)	462.2	152.7	313.2		98.5

The results were obtained by training a 3-layer GraphSage with a hidden state dimension of 512 on Yelp and Amazon datasets, and 256 on the rest, using four methods. We update the model with a mini-batch size of 1000 and an ADAM optimizer with a learning rate of 0.003 for all training methods. We use the efficient implementation of node-wise neighbor sampling, LADIES, and GNS in DGL, parallelized with multiprocessing. The number of sampling workers is 4. The column labeled "LADIES(512)" means sampling 512 nodes at each layer and "LADIES(5000)" means sampling 5000 nodes in each layer. In GNS, the size of cached was 1% of $|\mathcal{V}|$. LazyGCN runs out of GPU memory on OAG-paper and OGBN-papers100M.

As is shown in Table 2.3, GNS can obtain a comparable accuracy score compared to NS with $2\times - 4\times$ speed in training time, using a small cache. The acceleration is attributed to the smaller number of nodes in a mini-batch, especially a smaller number of nodes in the input layer (Table 2.4). This significantly reduces the time of data copy between CPU and GPUs as well as reducing the computation overhead in a mini-batch (Figure 2.2). In addition, a large number of input nodes have been cached in the GPU, which further reduces the time used in data copy between CPU to GPU. GNS scales well to giant graphs with 100 million nodes as long as the CPU memory of the machine can accommodate the graph. In contrast, LADIES cannot achieve state-of-the-art model accuracy and its training speed is slower than NS on many graphs. Our experiments also show that LazyGCN cannot achieve good model accuracy with a small mini-batch size, which is not friendly to giant graphs. In addition, The LazyGCN implementation provided by the authors fails to scale to giant graphs (e.g., OAG-paper and OGBN-products) due to the out-of-memory error even with a small mega-batch. Our method is robust to a small mini-batch size and can easily scale to giant graphs.

Table 2.4: The average number of input nodes in a mini-batch of NS and GNS as well as the average number of input nodes from the cache of GNS.

	#input nodes (NS)	#input nodes (GNS)	#cached nodes (GNS)
Yelp	151341	24150	5796
Amazon	132288	19063	13986
OGBN-products	433928	88137	21552
OAG-paper	408854	102984	56422
OGBN-Papers100M	507405	155128	111923

We plot the convergence rate of all of the training methods on OGBN-products based on the test F1-scores (Figure 2.3). In this study, LADIES samples 512 nodes per layer, and GNS caches 1% of nodes in the graph. The result indicates that GNS achieves similar convergence and accuracy as NS even with a small cache, thereby confirming the theoretical analysis in Section 2.3.5, while

LADIES and LazyGCN fail to converge to good model accuracy.

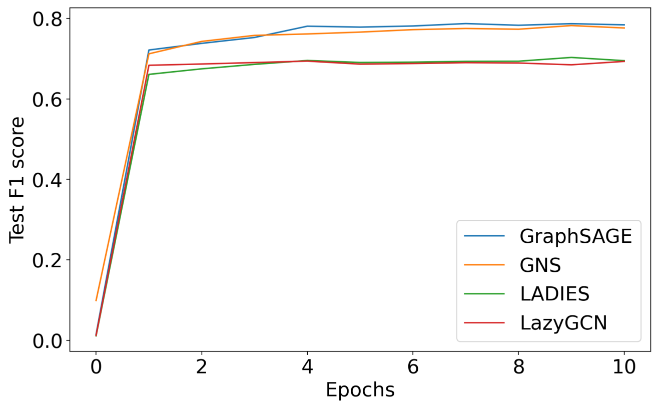


Figure 2.3: Comparison of the accuracy (F1 score) v.s. epochs.

Table 2.5: Percentage of isolated training nodes in LADIES.

# of sampled nodes/layer	256	512	1000	5000	10000
% of isolated target nodes	52.7	45.2	24.0	3.9	0

Percentage of isolated nodes in the first layer when training three-layer GCN on OGBN-products with LADIES.

One of reasons why LADIES suffers poor performance is that it tends to construct a mini-batch with many isolated nodes, especially for nodes in the first layer (Table 2.5). When training three-layer GCN on OGBN-products with LADIES, the percentage of isolated nodes in the first layer under different numbers of layerwise sampled nodes is illustrated in Tables 2.5.

LazyGCN requires a large batch size to train GCN on large graphs, which usually leads to out-of-memory. Under the same setting of the paper [13], we investigate the performance of nodewise LazyGCN on the Yelp dataset with different mini-batch sizes. As shown in Figure 2.4, LazyGCN performs poorly at the small mini-batch size. This may be caused by training with less representative graph data in mega-batch when recycling a small batch.

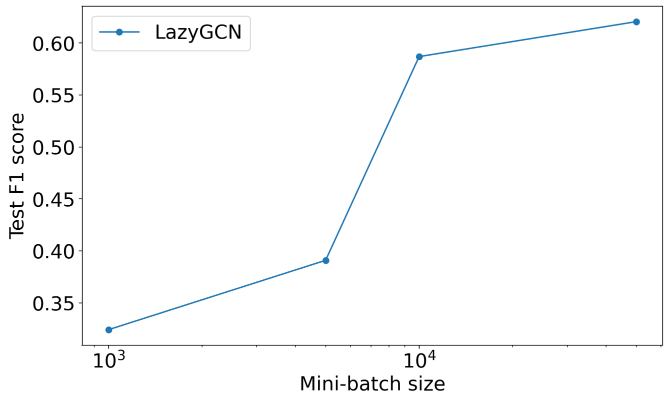


Figure 2.4: The effect of mini-batch size on the performance of LazyGCN on the Yelp dataset.

2.4.3 Hyperparameter Study

In this section, we explore the effect of various parameters in GNS on the OGBN-products dataset. Table 2.6 summarizes the test F1-score for different values of cache update period sizes P and cache sizes. A cache size as small as 0.01% can still achieve fairly good accuracy. As long as the cache size is sufficiently large (e.g., $1\% \cdot |\mathcal{V}|$), properly reducing the frequency of updating the cache (e.g., $P = 1, 2, 5$) does not affect performance. Note that it is better to get a $.1\%$ sample every epoch than a single 1% sample every 10 epochs, and $.01\%$ sample every epoch that the 0.1% sample every 10 epochs.

2.5 Conclusions

In this chapter, we propose a new effective sampling framework to accelerate GNN mini-batch training on giant graphs by removing the main bottleneck in mixed CPU-GPU training. GNS creates a global cache to facilitate neighbor sampling and periodically updates the cache. Therefore, it reduces data movement between CPU and GPU. We empirically demonstrate the advantages of the proposed algorithm in convergence rate, computational time, and scalability to giant graphs. Our proposed method has a significant speedup in training on large-scale datasets. We also theoretically

Table 2.6: GNS sensitivity to update period and cache size.

	cache update period size P			
Size of cache	$P = 1$	$P = 2$	$P = 5$	$P = 10$
$\lvert\mathcal{V}\rvert \times 1\%$	78.34	78.40	78.17	77.54
$\lvert\mathcal{V}\rvert \times .1\%$	78.04	77.31	76.16	74.71
$\lvert\mathcal{V}\rvert \times .01\%$	76.29	72.83	71.60	71.21

Performance in terms of test-set F1-score for different cache sizes and update periods.

analyze GNS and show that even with a small cache size, it enjoys a comparable convergence rate as the node-wise sampling method.

CHAPTER 3

Partial Information Selection in Retrieval Augmented Generation

3.1 Introduction

Retrieval Augmented Generation (RAG) [22] has improved many text generation problems. One example is Open-Domain Question Answering (ODQA) [23] which involves answering natural language questions without limiting the domain of the answers. RAG merges the retrieval and answering processes, which improves the ability to effectively collect knowledge, extract useful information, and generate answers. Even though it is successful in fetching relevant documents, RAG is not able to utilize connections between documents. In the ODQA setting, this leads to the model disregarding documents containing answers, a.k.a. *positive documents*, with less apparent connections to the question context. We can identify these documents if we connect them with positive documents whose context is strongly relevant to the question context.

To find connections between documents and select highly relevant ones, the reranking process plays a vital role in further effectively filtering retrieved documents. A powerful reranker also reduces the complexity of the reading process if it can successfully identify the positive documents. Thus, this dissertation focuses on using reranking to improve RAG – as it is a fundamental bridge between the retrieval and reading processes.

Pre-trained language models (LMs) like BERT [19], RoBERTa [24], and BART [25] have been widely used to enhance reranking performance by estimating the relevant score between questions and documents. Recently, the Abstract Meaning Representation (AMR) graph has been integrated

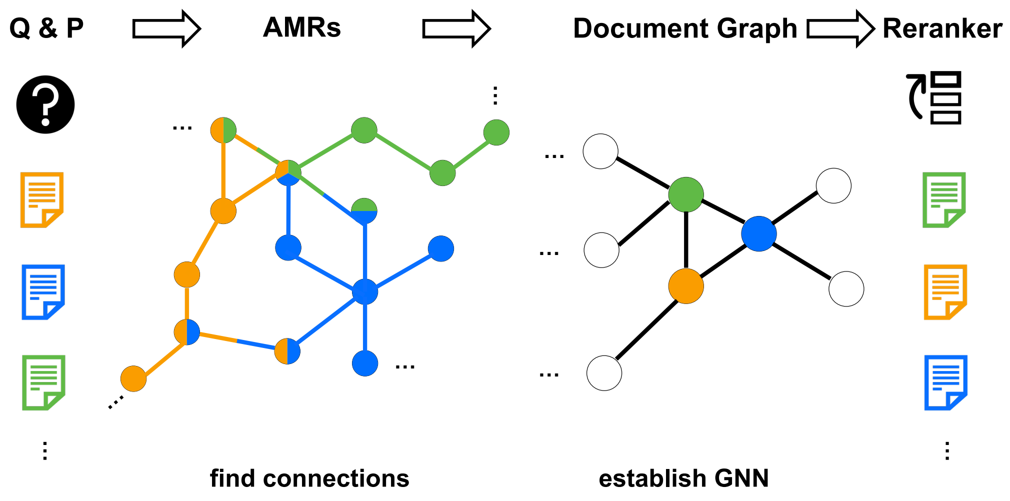


Figure 3.1: G-RAG uses two graphs for re-ranking documents: The Abstract Meaning Representation (AMR) graph is used as a feature for the document-level graph. A document graph is then used to rerank the document.

with an LM to enhance the system's ability to comprehend complex semantics [26]. While the current rerankers exhibit admirable performance, certain limitations persist.

Firstly, as mentioned above, most of the current works fail to capture important connections between different retrieved documents. Some recent work [27] tries to incorporate external knowledge graphs to improve the performance of the reading process in RAG but at the cost of significant memory usage for knowledge graph storage. The connection between documents has not been considered in the reranking process yet. Secondly, even though the AMR graph improves the understanding of the complex semantics, state-of-the-art [26] work integrates redundant AMR information into the pre-trained language models. This extra information can cause potential overfitting, in addition to increases of in computational time and GPU cost. Thirdly, current papers utilize common pre-trained language models as rerankers which are insufficient given the fast pace of LLM development. With the recent breakthroughs from LLM, researchers are curious about how LLMs perform (without fine-tuning) on the reranking task.

To address these challenges and limitations, we propose a method based on document graphs, where each node represents a document, and each edge represents that there are common concepts between two documents. We incorporate the connection information between different documents into the edge features and update the edge features through the message-passing mechanism. For node features, even though we aim to add AMR information to compose a richer understanding of complex semantics, we won't overwhelmingly add all AMR-related tokens as node-level features. Instead, we investigate the determining factor that facilitates the reranker to identify more relevant documents and encode this key factor to node features.

Moreover, instead of using the cross-entropy loss function during the training, we apply pairwise ranking loss in consideration of the essential aim of ranking. We also investigate the performance of a publicly available LLM, i.e., PaLM 2 [28] with different versions, as a reranker on an ODQA. According to the moderate performance of PaLM 2 on reranking tasks, we provide several potential reasons and emphasize the irreplaceable role of reranker model design to improve RAG. The framework of graph-based reranking in the proposed G-RAG is illustrated in Fig 3.1.

Our contributions can be summarized as follows:

1. To improve RAG for ODQA, we propose a document-graph-based reranker that leverages connections between different documents. When the documents share similar information with their neighbor nodes, it helps the reranker to successfully identify the documents containing answer context that is only weakly connected to the question.

2. We introduce new metrics to assess a wide range of ranking scenarios, including those with tied ranking scores. The metrics effectively evaluate this scenario by diminishing the optimistic effect brought by tied rankings. Based on these metrics, our proposed method outperforms state-of-the-art and requires fewer computational resources.

3. We assess the performance of a publicly available LLM (PaLM 2 [28]) as a reranker, exploring variations across different model sizes. We find that excessive ties within the generated ranking scores hinder the effectiveness of pre-trained large language models in improving RAG through reranking.

3.2 Related Work

RAG in ODQA. RAG [29, 22] combines information retrieval (via Dense Passage Retrieval, DPR [30]) and a reading process in a differentiable manner for ODQA. A line of literature focuses on developing rerankers for further improving RAG. Approaches like monoT5 [31] and monoELECTRA [32] use proposed pre-trained models. Moreover, [33] proposes a fine-tuned T5 version as a reranker. More recently, [34] developed a reranker module by fine-tuning the reader's neural networks through a prompting method. However, the above approaches neglect to investigate the connections among documents and fail to leverage this information during the reranking process. These methods are prone to fail to identify the documents containing gold answers that may not exhibit obvious connections to the question context. To address this issue, our proposed method is based on document graphs and is more likely to identify valuable information contained in a document if most of its neighboring document nodes in the graph share similar information.

Graphs in ODQA. Knowledge graphs, which represent entities and their relations, have been leveraged in ODQA [27, 35, 36, 37] to improve the performance of RAG. However, KG-based methods require large external knowledge bases and entity mapping from documents to the entities in the knowledge graph, which would increase the memory cost. Our proposed method does not depend on external knowledge graphs. While recent work by [26] uses AMR graphs generated from questions and documents to construct embeddings, their focus remains on text-level relations within a single document. In contrast, our approach uniquely leverages document graphs to characterize cross-document connections, a novel application within the RAG reranking process.

Abstract Meaning Representation (AMR). AMR [38] serves as a promising tool for representing textual semantics through a rooted, directed graph. In the AMR graph, nodes represent basic semantic units like entities and concepts, while edges denote the connections between them. AMR graphs have more structured semantic information compared to the general form of natural language [39, 40]. A line of literature has integrated AMR graphs into learning models. Recently, [26] have applied AMR to ODQA to deal with complex semantic information. Even though the performance of the reranker and the reader is improved in [26], their method also increases the computational time and GPU memory cost. This issue may arise by integrating all tokens of AMR nodes and edges without conscientiously selecting the key factors. To address this issue, our method aims to investigate the graph structure of AMR graphs and identify the key factors that improve the performance of the reranker.

LLMs in Reranking. LLMs such as ChatGPT [41], PaLM 2 [28], LLaMA [42], and GPT4 [43], have proven to be capable of providing answers to a broad range of questions due to their vast knowledge repositories and chain-of-thought reasoning capability. With this breakthrough, researchers are seeking to explore potential improvements that LLMs can bring to improve RAG in ODQA, such as [44, 45]. At the same time, several studies [46, 47] have scrutinized the efficacy of LLMs in Question-Answering. [46] indicates the superiority of the DPR [30] + FiD [48] approach over LLM in ODQA. Despite these investigations, the potential of LLMs without fine-tuning as rerankers to improve RAG remains unexplored, as existing studies often take pre-trained language models such as BERT [19], RoBERTa [24], and BART [25] in the reranker role.

3.3 Proposed Method: G-RAG

G-RAG leverages the rich structural and semantic information provided by the AMR graphs to enhance document reranking. 3.3.1 details how we use AMR graph information and build a graph structure among the retrieved documents. 3.3.2 outlines the design of our graph neural network architecture for reranking documents.

3.3.1 Establishing Document Graphs via AMR

In ODQA datasets we consider, one *document* is a text block of 100 words that come from the text corpus. For each question-document pair, we concatenate the question q and document

p as "question:¡question text¿¡document text¿" and then exploit AMRBART [49] to parse the sequence into a singular AMR graph. The AMR graph for question q and document p is denoted as $G_{qp} = \{V, E\}$, where V and E are nodes and edges, respectively. Each node is a concept, and each edge is denoted as $e = (s, r, d)$ where s, r, d represent the source node, relation, and the destination node, respectively. Our reranker aims to rank among the top 100 documents retrieved by DPR [30]. Thus, given one question q and documents $\{p_1, \cdots, p_n\}$ with $n = 100$, we establish the undirected document graph $\mathcal{G}_q = \{\mathcal{V}, \mathcal{E}\}$ based on AMRs $\{G_{qp_1}, \cdots, G_{qp_n}\}$. For each node $v_i \in \mathcal{V}$, it corresponds to the document p_i. For $v_i, v_j \in \mathcal{V}$, $i \neq j$, if the corresponding AMR G_{qp_i} and G_{qp_j} have common nodes, there will be an undirected edge between v_i and v_j (with a slight abuse in notation) denoted as $e_{ij} = (v_i, v_j) \in \mathcal{E}$. We remove isolated nodes in $\mathcal{G}_q$. In the following, we will construct the graph neural networks based on the document graphs to predict whether the document is relevant to the question. Please refer to Appendix B.1 for AMR graph statistics, i.e., the number of nodes and edges in AMR graphs, of the common datasets in ODQA.

3.3.2 Graph Neural Networks for Reranking

Following Section 3.3.1, we construct a graph among the $n = 100$ retrieved documents denoted as $\mathcal{G}_q$ given the question q. We aim to exploit both the structural information and the AMR semantic information to rerank the retrieved documents. To integrate the semantic information of documents, the pre-trained language models such as BERT [19], and RoBERTa [24] are powerful tools to encode the document texts as node features in graph neural networks. Even though [26] integrates AMR information into LMs, it increases the computational time and GPU memory usage. To address this, we proposed node and edge features for graph neural networks, which simultaneously exploit the structural and semantic information of AMR but avoid adding redundant information.

3.3.2.1 Generating Node Features

Our framework applies a pre-trained language model to encode all the n retrieved documents in $\{p_1, p_2, \cdots, p_n\}$ given a question q. The document embedding is denoted as $\tilde{X} \in \mathbb{R}^{n \times d}$ where d is the hidden dimension, and each row of $\tilde{X}$ is given by

$$\tilde{x}_i = \mathrm{Encode}(p_i) \text{ for } i \in \{1, 2, \cdots n\}. \tag{3.1}$$

Since AMR brings more complex and useful semantic information, we intend to concatenate document text and corresponding AMR information as the input of the encoder. However, if we integrate all the information into the embedding process as the previous work [26] did, it would bring high computational costs and may lead to overfitting. To avoid this, we investigate the determining factor that facilitates the reranker to identify more relevant documents. By studying the structure of AMRs for different documents, we note that almost every AMR has the node "question", where the word "question" is included in the input of the AMR parsing model, given by "question:(question text)(document text)". Thus, we can find the single source shortest path starting from the node "question". When listing every path, the potential connection from the question to the answer becomes much clearer. By looking into the nodes covered in each path, both the structural and semantic information can be collected. The embedding enables us to utilize that information to identify the similarity between question and document context.

To better illustrate the structure of the shortest path, we also conduct some experiments to show the statistic of the shortest path, see Fig B.2 in Appendix. We study the shortest single source paths (SSSPs) starting from "question" in the AMR graphs of documents from the train set of Natural Question (NQ) [50] and TriviaQA (TQA)[51] dataset. The analysis shows that certain negative documents cannot establish adequate connections to the question context within their text. Moreover, negative documents encounter another extreme scenario where paths contain an abundance of information related to the question text but lack valuable information such as the gold answers. This unique pattern provides valuable insight that can be utilized during the encoding process to improve the reranker performance.

Thus, the proposed document embedding is given by $X \in \mathbb{R}^{n \times d}$ and each row of X can be given by, for $i \in \{1, 2, \cdots n\}$:

$$x_i = \text{Encode}(\text{concat}(p_i, a_i)), \tag{3.2}$$

where a_i denotes the AMR information to the document p_i. There are two steps to get the representation of a_i: 1) find the shortest single source paths (SSSPs) starting from the node "question" in AMR graph G_{qp_i} and each path is not the subset of other paths, e.g., an example path is ['question', 'cross', 'world-region', 'crucifix', 'number', 'be-located-at', 'country', 'Spain']; 2) extract the node concepts to construct a_i, e.g, part of a_i can be represented as ¡... question cross world-region crucifix number be-located-at country Spain ...¿. $X \in \mathbb{R}^{n \times d}$ (3.2) will be the initial node representation of graph neural networks.

3.3.2.2 Edge Features

Besides the node features, we also adequately leverage edge features associated with undirected edges in AMR $\{G_{qp_1}, \cdots, G_{qp_n}\}$. Let $\hat{E} \in \mathbb{R}^{n \times n \times l}$ denote the edge features of the graph. Then, $\hat{E}_{ij\cdot} \in \mathbb{R}^l$ represents the l-dimensional feature vector of the edge between the node v_i and node v_j $i \neq j$, and $\hat{E}_{ijk}$ denotes the k-th dimension of the edge feature in $\hat{E}_{ij\cdot}$. In our framework, $l = 2$ and $\hat{E}$ is given by:

$$\begin{cases} \hat{E}_{ij\cdot} = 0, \text{no connection between } G_{qp_i} \text{ and } G_{qp_j}, \\ \hat{E}_{ij1} = \text{\# common nodes between } G_{qp_i} \text{ and } G_{qp_j}, \\ \hat{E}_{ij2} = \text{\# common edges between } G_{qp_i} \text{ and } G_{qp_j}. \end{cases} \tag{3.3}$$

We then normalize the edge feature $\hat{E}$ to avoid the explosive scale of output node features when being multiplied by the edge feature in graph convolution operations. Thus, our derived feature E is normalized on the first and second dimension, respectively. Similar edge normalization has also been considered in the paper [52]. $E \in \mathbb{R}^{n \times n \times l}$ will be the initial edge representation of graph neural networks.

3.3.2.3 Representation Update

Based on the above initial node and edge representations, we arrive at updating representations in the graph neural networks. Given a document graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = n$, the input feature of node $v \in \mathcal{V}$ is denoted as $\mathbf{x}_v^0 \in \mathbb{R}^d$, and the initial representation of the edge between node v and u is given by $\mathbf{e}_{uv}^0 \in \mathbb{R}^l$ with $l = 2$. Let $\mathcal{N}(v)$ denote the neighbor nodes of the node $v \in \mathcal{V}$. The representation of node $v \in \mathcal{V}$ at layer ℓ can be derived from a GNN model given by:

$$\mathbf{x}_v^{\ell} = g(\mathbf{x}_v^{\ell-1}, \bigcup_{u \in \mathcal{N}(v)} f(\mathbf{x}_u^{\ell-1}, \mathbf{e}_{uv}^{\ell-1})), \tag{3.4}$$

where f, $\bigcup$, and g are functions for computing feature, aggregating data, and updating node representations, respectively. Specifically, the function f applies different dimensional edge features as weights to the node features, given by

$$f(\mathbf{x}_u^{\ell-1}, \mathbf{e}_{uv}^{\ell-1}) = \sum_{m=1}^{l} e_{uv}^{\ell-1}(m)\mathbf{x}_u^{\ell-1}. \tag{3.5}$$

We choose mean aggregator [53] as the operation $\bigcup$. The parameterized function g is a non-linear learnable function that aggregates the representation of the node and its neighbor nodes. Simultaneously, the representation of edge starting from $v \in \mathcal{V}$ at layer ℓ is given by:

$$\mathbf{e}_{v\cdot}^{\ell} = g(\mathbf{e}_{v\cdot}^{\ell-1}, \bigcup_{u \in \mathcal{N}(v)} \mathbf{e}_{u\cdot}^{\ell-1}). \tag{3.6}$$

3.3.2.4 Reranking Score and Training Loss

Given a question q and its document graph $\mathcal{G}_q = \{\mathcal{V}, \mathcal{E}\}$, we have the output node representations of GNN, i.e., $\mathbf{x}_v^L$, where L is the number of GNN layers. With the same encoder in (3.2), the question q is embedded as

$$\mathbf{y} = \text{Encode}(q). \tag{3.7}$$

The reranking score for each node $v_i \in \mathcal{V}$ corresponding the document p_i is calculated by

$$s_i = \mathbf{y}^{\top}\mathbf{x}_{v_i}^{L}, \tag{3.8}$$

for $i = 1, \cdots, n$ and $|\mathcal{V}| = n$. The cross-entropy training loss of document ranking for the given question q is:

$$\mathcal{L}_q = -\sum_{i=1}^{n} y_i \log \left(\frac{\exp(s_i)}{\sum_{j=1}^{n} \exp(s_j)} \right) \tag{3.9}$$

where $y_i = 1$ if p_i is the positive document, and 0 for the negative document. The cross-entropy loss may fail to deal with the unbalanced data in ODQA where the number of negative documents is much greater than the number of positive documents. Besides the cross-entropy loss function, the pairwise loss function has been a powerful tool for ranking [54]. Given a pair of scores s_i and s_j, the ranking loss is given by :

$$\mathcal{RL}_q(s_i, s_j, r) = \max\left(0, -r\left(s_i - s_j\right) + 1\right), \tag{3.10}$$

where $r = 1$ if document i should be ranked higher than document j, and vice-versa for $r = -1$. We conduct experiments based on both loss functions and emphasize the advantage of the ranking loss (3.10) over the cross-entropy loss (3.9).

3.4 Experiments

3.4.1 Setting

Datasets. We conduct experiments on two representative ODQA datasets Natural Questions(NQ) [50] and TriviaQA (TQA) [51]. NQ is derived from Google Search Queries and TQA includes questions from trivia and quiz-league websites. Detailed dataset statistics are presented in Table B.1 in Appendix B.1. Note that the gold answer lists in dataset NQ usually have much fewer elements than the dataset TQA, which leads to a much smaller number of positive documents for each question.

We use DPR [30] to retrieve 100 documents for each question and generate the AMR graph for each question-document pair using AMRBART [49]. The dataset with AMR graphs is provided by [26][1]. Please refer to Appendix B.1 for more details on the AMR statistic information. We conducted our experiments on a Tesla A100 40GB GPU, demonstrating the low computational needs of G-RAG.

Model Details. For the GNN-based reranking models, we adopt a 2-layer Graph Convolutional Network [53] with hidden dimension chosen from $\{8, 64, 128\}$ via hyperparameter-tuning. The dropout rate is chosen from $\{0.1, 0.2, 0.4\}$. We initialize the GNN node features using pre-trained models, e.g, BERT [19], GTE [55], BGE [56], Ember [57]. We base our implementation of the embedding model on the HuggingFace Transformers library [58]. For training our framework, we adopt the optimizer AdamW [59] with the learning rate chosen from $\{5e-5, 1e-4, 5e-4\}$. Batch size is set to 5. We set the learning rate warm-up with $1,000$ steps. The number of total training steps is 50k, and the model is evaluated every 10k steps.

3.4.1.1 Metrics

Even though Top-K accuracy, where the ground-truth ranking is based on DPR scores [30], is commonly used in the measurement of reranking [60, 48], this metric is unsuitable for indicating the overall reranking performance for all positive documents. Moreover, with the promising development of LLM in learning the relevance between texts, DPR scores may lose their advantage and fairness. To address this issue, other metrics such as Mean Reciprocal Rank (MRR) and Mean Hits@10 (MHits@10) are used for measuring the reranking performance [26]. To be specific, The Mean Reciprocal Rank (MRR) score of a positive document is given by $\mathrm{MRR} = \frac{1}{|\mathcal{Q}|}\sum_{q\in\mathcal{Q}}(\frac{1}{|\mathcal{P}^+|}\sum_{p\in\mathcal{P}^+}\frac{1}{r_p})$, where $\mathcal{Q}$ is the question set from the evaluating dataset, $\mathcal{P}^+$ is the set of positive documents, and r_p is the rank of document p estimated by the reranker. The MHits@10 indicates the percentage of positive documents that are ranked in the Top 10, given by $\mathrm{MHits@10} = \frac{1}{|\mathcal{Q}|}\sum_{q\in\mathcal{Q}}(\frac{1}{|\mathcal{P}^+|}\sum_{p\in\mathcal{P}^+}\mathbb{I}(r_p <= 10))$, where the indication $\mathbb{I}(A) = 0$ if the event A is true, otherwise 0.

[1]`https://github.com/wangcunxiang/Graph-aS-Tokens/tree/main`

The above metrics work well for most cases, however, they may fail to fairly characterize the ranking performance when there are ties in ranking scores, which is common in relevant scores generated by LLMs such as ChatGPT [41], PaLM 2 [28], LLaMA [42], and GPT4 [43]. Please refer to Fig B.3 in the Appendix for the detailed prompt and results of relevant scores between questions and documents. To address ties in the ranking scores, we propose variants of MRR and MHits@10. Denote $r_p^{(t)}$ as the rank of the document p with t ties. In other words, the relevant score between the question and the document p is the same as other $t-1$ documents. The variant of MRR for tied ranking is named Mean Tied Reciprocal Ranking (MTRR), represented as

$$\begin{aligned}\mathrm{MTRR} = \frac{1}{|\mathcal{Q}|}\sum_{q\in\mathcal{Q}}\Bigg(\frac{1}{|\mathcal{P}^+|}\sum_{p\in\mathcal{P}^+}\frac{1}{r_p^{(t)}}\mathbb{I}(t=1) \\ + \frac{2}{r_p^{(t)}+r_p^{(t)}+t-1}\mathbb{I}(t>1)\Bigg).\end{aligned} \tag{3.11}$$

The metric MTRR addresses the tied rank $r_p^{(t)}$ estimated by the reranker via averaging the optimistic rank $r_p^{(t)}$ and the pessimistic rank $r_p^{(t)}+t-1$. The metrics MRR and MTRR are the same when there is no ranking tie. The variant of MHits@10 for tied ranking is Tied Mean Hits@10 (TMHit@10). Denote $\mathcal{H}(p)$ as the set that includes all the ranks $\{r_{p_i}^{(t_i)}\}$ that are higher than the rank of document p, i.e., $r_p^{(\tau)}$. Based on these notations, we present the new metric as:

$$\mathrm{TMHits@10} = \frac{1}{|\mathcal{Q}|}\sum_{q\in\mathcal{Q}}\left(\frac{1}{|\mathcal{P}^+|}\sum_{p\in\mathcal{P}^+}\mathrm{Hits@10}(p)\right), \tag{3.12}$$

where $\mathrm{Hits@10}(p)$ is defined as

$$\begin{cases} 0, \text{if } \sum_i t_i > 10 \text{ for } \forall\, r_{p_i}^{(t_i)} \in \mathcal{H}(p), \\ (10-\sum_i t_i)/\tau, \text{if } 0 < \sum_i t_i < 10 \\ \quad \text{for } \forall\, r_{p_i}^{(t_i)} \in \mathcal{H}(p) \text{ and } \tau > 1, \\ 10/\tau, \text{if } \mathcal{H}(p) = \emptyset \text{ and } \tau > 10, \\ 1, \text{otherwise.} \end{cases}$$

If there are ties in the Top-10 ranking, the metric TMHit@10 diminishes the optimistic effect by dividing the hit number (no greater than 10) by the number of ties.

3.4.2 Comparing Reranker Systems

We compare our proposed algorithm with baselines as follows with fixed hyper-parameter in the absence of fine-tuning, where the hidden dimension is 8, the dropout rate is 0.1, and the learning rate is 1e-4. **w/o reranker:** Without an additional reranker, the ranking score is based on the retrieval scores provided by DPR [30]. **BART:** The pre-trained language model BART [25] severs as the reranker. **BART-GST:** The method proposed by Wang et al. integrates graph-as-token into the pre-trained model [26]. For each dataset, we use the best performance provided in the paper. **RGCN-S:** It is introduced by [26] that stacks the RGCN model on the top of the transformer. Even though this method is based on graph neural networks, it doesn't rely on the document graphs but constructs nodes in the graph model based on the text alignment in the question-document pair. **MLP:** The initial node features are only based on document text as described in (3.1) with the BERT [19] encoder. After the node features go through MLP, we get the relevant scores via (3.8) and take the cross-entropy function (3.9) as training loss. **GCN:** Besides updating node representations via GCN, the rest setting is the same as MLP. We also conduct experiments with different GNN models. Please refer to Appendix B.2 for details. **G-RAG:** The initial node features are based on document text and AMR information as described in (3.2). The rest of the setting is the same as GCN. **G-RAG-RL:** Using the ranking loss function and keep the other setting the same as G-RAG.

	NQ		**TQA**	
Strategy /Metric	**MRR_dev /MRR_test**	**MH@10_dev /MH@10_test**	**MRR_dev /MRR_test**	**MH@10_dev /MH@10_test**
w/o reranker	20.2/18.0	37.9/34.6	12.1/12.3	25.5/25.9
BART	25.7/23.3	49.3/45.8	16.9/17.0	37.7/38.0
BART-GST	**28.4**/25.0	**53.2/48.7**	17.5/17.6	39.1/**39.5**
RGCN-S	26.1/23.1	49.5/46.0	—	—
MLP	19.2/17.8	40.0/38.8	17.6/17.1	34.0/31.4
GCN	22.6/22.4	47.6/44.2	18.2/17.4	38.0/37.0
G-RAG	25.1/24.2	49.1/47.2	18.5/**18.3**	38.5/39.1
G-RAG-RL	27.3/**25.7**	49.2/47.4	**19.8/18.3**	**42.9**/39.4

Table 3.1: Results on the dev/test set of NQ and TQA without hyperparameter fine-tuning.

The results on MRR and MHits@10 on the NQ and TQA datasets are provided in Table 3.1. Note that the results on NQ always outperform the results on TQA, this is due to a smaller number of positive documents making it easy to put most of the positive documents into the Top 10. Generally speaking, TQA is a more complex and robust dataset than NQ. Models with graph-based approaches, such as GCN and G-RAG, show competitive performance across metrics. These methods have advantages over the baseline models, i.e., without reranker and MLP. In conclusion, based on the simulation results, the proposed method G-RAG-RL emerges as a strong model, indicating the effectiveness of graph-based strategies and the benefit of pairwise ranking loss on identifying positive documents. To effectively present the potential advantages of the proposed G-RAG over state-of-the-art benchmarks, we conducted experiments across various embedding models with fine-tuning parameters in the next section.

3.4.3 Using different LLMs as Embedding Models

The feature encoder always plays a vital role in natural language processing-related tasks. Better embedding models are more likely to fetch similarities across contexts and help identify highly relevant contexts. Besides the BERT model used in the state-of-the-art reranker, many promising embedding models have been proposed recently. To evaluate the effectiveness of different embedding models, i.e., BERT [19], GTE [55], BGE [56], Ember [57], we conduct the experiments under the same setting as G-RAG-RL. The results are illustrated in Table 3.2. For the convenience of comparison, we directly add two results from Section 3.4.2, i.e., BART-GST and BERT in Table 3.2. It shows that Ember performs consistently well across both datasets and most evaluation metrics. In conclusion, Ember appears to be the top-performing model, followed closely by GTE and BGE, while BART-GST and BERT show slightly lower performance across the evaluated metrics. Thus our fine-tuning result is based on G-RAG-RL with Ember as the embedding model. The grid search setting for hyperparameter is introduced in Section 3.4.1. We only run 10k iterations for each setting and pick up the one with the best MRR. The result with hyperparameter tuning, i.e., Ember (HPs-T), is added in Table 3.2. Even though BART-GST demonstrates competitive performance in some

	NQ		TQA	
Embedding /Metric	**MRR_dev /MRR_test**	**MH@10_dev /MH@10_test**	**MRR_dev /MRR_test**	**MH@10_dev /MH@10_test**
BART-GST	28.4/25.0	**53.2**/48.7	17.5/17.6	39.1/39.5
BERT	27.3/25.7	49.2/47.4	19.8/18.3	42.9/39.4
GTE	**29.9**/26.3	52.6/47.7	19.2/19.3	41.8/40.3
BGE	28.7/27.4	52.1/48.2	18.7/18.3	43.4/40.7
Ember	29.0/26.1	52.9/48.0	19.8/18.6	**44.3/42.0**
Ember (HPs-T)	28.9/**27.7**	51.1/**50.0**	**20.0/19.4**	41.6/41.4

Table 3.2: G-RAG with changing the embedding model.

scenarios, it is prone to be overfitting especially in terms of MRR in the NQ dataset. However, the proposed methods, i.e., Ember and Ember (HPs-T), are more likely to avoid overfitting and achieve the highest values in all test sets.

3.4.4 Investigating PaLM 2 Scores

To evaluate the performance of large language models on the reranking task, we conduct zero-short experiments on the dev & test sets of the NQ and TQA datasets. An example of an LLM-generated relevance score is illustrated in Figure B.3 in the Appendix.

In general, we observe that scores generated by PaLM 2 are integers between 0 and 100 that are divisible by 5. This often leads to ties in document rankings. To address the ties in the ranking score, we use the proposed metrics MTRR (Eq. 3.11) and TMHits@10 (Eq. 3.12) to evaluate the performance of reranker based on PaLM 2 [28]. For the convenience of comparison, we copy `w/o rerank`, BART, and G-RAG results from Section 3.4.2. Since there is no tied ranking provided by `w/o rerank` and BART, the MRR and MHits@10 have the same values as MTRR and TMhits@10, respectively.

The performance results are provided in Table 3.3. The results demonstrate that LLMs with zero-

Strategy/Metric	**NQ**				**TQA**			
	MTRR		**TMH**		**MTRR**		**TMH**	
	dev	**test**	**dev**	**test**	**dev**	**test**	**dev**	**test**
w/o reranker	20.2	18.0	37.9	34.6	12.1	12.3	25.5	25.9
BART	25.7	23.3	**49.3**	45.8	16.9	17.0	37.7	38.0
PaLM2 XS	14.9	14.0	34.1	34.2	11.6	12.5	29.1	31.6
PaLM2 L	18.6	17.9	40.7	39.7	12.7	12.9	34.7	35.6
G-RAG-RL	**27.3**	**25.7**	49.2	**47.4**	**19.8**	**18.3**	**42.9**	**39.4**

Table 3.3: Results of PaLM 2 being the reranker. Small embedding models outperform LLMs in this setting. In comparison, G-RAG-RL considerably improves the results compared to both language model types by leveraging connection information across documents. We use Tied Mean Hits@10.

shot learning do not do well in reranking tasks. This may be caused by too many ties in the relevance scores, especially for small-size LLM where there are more of them. This result emphasizes the importance of reranking model design in RAG even in the LLM era. More qualitative examples based on PaLM 2 are provided in Appendix B.3.

We compare the results of both approaches with G-RAG which brings additional perspective to these results. Leveraging the information about connections of entities across documents and documents themselves brings significant improvements up to 7 percentage points.

3.5 Conclusions

Our proposed model, G-RAG, addresses limitations in existing ODQA methods by leveraging implicit connections between documents and strategically integrating AMR information. Our method identifies documents with valuable information significantly better even when this information is only weakly connected to the question context. This happens because documents connected by the document graph share information that is relevant to the final answer. We integrate key AMR information to improve performance without increasing computational cost. We also proposed two metrics to fairly evaluate the performance of a wide range of ranking scenarios including tied ranking scores. Furthermore, our investigation into the performance of PaLM 2 as a reranker emphasizes the significance of reranker model design in RAG, as even an advanced pre-trained LLM might face challenges in the reranking task. There are more directions for future study. For instance, designing more sophisticated models to better process AMR information and integrating this information into node & edge features will bring further improvements in reranking. Further, while a pre-trained LLM does not have impressive performance as a reranker itself, fine-tuning it may be extremely useful for enhancing the performance of RAG systems.

CHAPTER 4

Exploration of Partial Information in Sparse Bandit Problems

4.1 Introduction

Bandit and reinforcement learning problems in real-world applications, e.g., autonomous driving [61], healthcare [62], recommendation system [63], marketing and advertising [64], are challenging due to the magnificent state/action space. To address this challenge, a function approximation framework has been introduced, which first extracts feature vectors for state/action space and then approximates the value functions of all policies in RL (or the reward functions of all actions in bandit problems) with feature representations. In some real-world applications, feature representations may not have vanilla linear mapping. In these scenarios, a linear feature representation can approximate the value functions (or the reward functions) with a small uniform error known as misspecification. Unfortunately, [65] shows that searching for an $O(\varepsilon)$-optimal action in these scenarios requires pulling at least $\Omega(\exp(d))$ queries. However, if we relax the goal of finding $O(\varepsilon)$-optimal action, there is still a chance. Instead, [66] find an action that is suboptimal with an error of at most $O(\varepsilon\sqrt{d})$ within $\mathrm{poly}(d/\varepsilon)$ queries, where d is the dimension of the feature vectors.

By scrutinizing the novel result proposed by [66], the dependence on $\sqrt{d}$ raises concern regarding the potential blowup of the approximation error. We are modestly optimistic that some structural patterns, such as sparsity, in feature representation schemes are beneficial to break the $\varepsilon\sqrt{d}$ barrier. This idea comes from a vast literature that studies high-dimensional statistics in sparse linear regression [67, 68] and successfully applies it to sparse linear bandits [69, 70, 71, 72, 73, 74]. Moreover, the sparsity-structure in linear bandits are meaningful and crucial to many practical applications where there are many potential features but no apparent evidence on which are relevant, such as personalized health care and online advertising [75, 70]. The essential difference in sparse linear bandits between this dissertation and state-of-the-art is the study of the possible model misspecification, i.e., the ground truth reward means might be an ε error away from a sparse linear representation for any action.

Model misspecification is widely seen in practice and has been widely studied only in the dense model (also known as misspecified linear bandits) [76, 77, 78, 79], where the best polynomial-sample algorithm suffers a $O(\varepsilon\sqrt{d})$ estimation error, which can be prominent when the feature dimension d is sufficiently large. However, it is unexplored whether a structural sparsity assumption on the ground-truth parameter could break the $\varepsilon\sqrt{d}$ barrier. Additionally, there is little understanding of the conditions when linear features are "useful" for bandit problems and reinforcement learning with misspecification.

Contribution.

- We establish novel algorithms that obtain $O(\varepsilon)$-optimal actions by querying $O(\varepsilon^{-s}d^s)$ actions, where s is the sparsity parameter. For fixed sparsity s, the algorithm finds an $O(\varepsilon)$-optimal action with $\mathrm{poly}(d/\varepsilon)$ queries, breaking the $O(\varepsilon\sqrt{d})$ barrier. The ε^{-s} dependence in the sample bound can be further improved to $\tilde{O}(s)$ if we allow an $O(\varepsilon\sqrt{s})$ suboptimality.
- We establish information-theoretical lower bounds to show that our upper bounds are nearly tight. In particular, we show that any sound algorithms that can obtain $O(\Delta)$-optimal actions need to query $\Omega(\exp(m\varepsilon/\Delta))$ samples from the bandit environment, where the approximate error Δ, defined in Definition 1, satisfies $\Delta \geq \varepsilon$. Hence, for approximation error of the form $O(s^{\delta}\varepsilon)$, for any $0 < \delta < 1$, $\exp(s)$-dependence in the sample complexity is not avoidable.
- We further break the $\exp(s)$ sample barrier by showing an algorithm that achieves $O(s\varepsilon)$ sub-optimal actions while only querying $\mathrm{poly}(s/\varepsilon)$ samples in the regime the action features possess specific benign structures (hence "good" features). We then relax the benign feature

requirement to arbitrary feature settings and propose an algorithm with efficient sample complexity of $\mathrm{poly}(s/\varepsilon)$.

In summary, our results provide a nearly complete picture of how sparsity can help in misspecified bandit learning and provide a deeper understanding of when linear features are "useful" for the bandit and RL with misspecification.

4.2 Related Work

This section summarizes the state-of-the-art in several areas of interest related to our work: function approximation, misspecified feature representation, and sparsity in bandits and reinforcement learning.

Function approximation in bandits and reinforcement learning Function approximation schemes that approximate value functions in RL (reward function in bandit problem) with feature representations are widely used for generalization across large state/action spaces. A recent line of work studies bandits [80, 81, 82, 83] and RL with linear function approximation [84, 78, 85, 86, 87, 88]. Beyond the linear setting, there is a flurry line of research studying RL with general function approximation [89, 90, 91] and bandits with general function approximation [92, 93, 94, 95, 96]. The regret upper bound $O(\mathrm{poly}(d)\sqrt{n})$ can be achieved in the above papers, where d is the ambient dimension (or complexity measure such as eluder dimension) of the feature space and n is the number of rounds.

Misspecified bandits and reinforcement learning Recently, interest has been aroused in dealing with the situation when the value function in RL (or the rewards functions in bandits) is approximated by a linear function where the approximation error is at most ε, also known as the misspecified linear bandit and reinforcement learning. The misspecification facilitates us to establish a more complicated reward function than a linear function. For instance, it enables the characterization of a reward function that may change over the rounds, which is common in real-world applications such as education, healthcare, and recommendation systems [83].

The paper [65] showed that no matter whether value-based learning or model-based learning, the agent needs to sample an exponential number of trajectories to find an $O(\varepsilon)$-optimal policy for reinforcement learning with ε-misspecified linear features. This result shows that good features (e.g., linear features with small misspecification) are not sufficient for sample-efficient RL if the approximation error guarantee is close to the misspecification error. By relaxing the objective of achieving $O(\varepsilon)$-optimality, [66] showed that $\mathrm{poly}(d/\varepsilon)$ samples are sufficient to obtain an $O(\varepsilon\sqrt{d})$-optimal policy (in the simulator model setting of RL), where d is the feature dimension, indicating the same features are "good" in a different requirement. The hard instances used in both papers are bandit instances and hence provide understanding for misspecified linear bandit problems as well.

Many works in the literature, such as [97, 98, 77, 99, 84], can also deal with misspecification in linear bandits or RL with linear features. These algorithms can only achieve a $O(\varepsilon\sqrt{d})$ error guarantee at best (when their regret bounds are translated to PAC bounds) with $\mathrm{poly}(d/\varepsilon)$ samples.

Sparse linear bandits and reinforcement learning In this section, we briefly review the literature on the sparse linear bandits and RL, where no misspecification is considered. We also note that these results are stated in regret bounds, which can be easily converted to PAC bounds.

The paper [70] proposed an online-to-confidence-set conversion approach which achieves a regret upper bound of $O(\sqrt{sdn})$, where s is a known parameter on the sparsity. A matching lower bound is given in [100][Chapter 24.3], which shows that polynomial dependence on d is generally unavoidable without additional assumptions. To address this limitation, another line of literature [101, 71, 72] studied the sparse contextual linear bandits where the action set is different in each round and follows some context distribution. [101] developed a doubly-robust Lasso bandit approach with an $O(s\sqrt{n})$ upper bound. [71] considered the scenario where each arm has an underlying parameter and derived a $O(Ks^2(\log(n))^2)$ upper bound which was improved to $O(Ks^2\log(n))$ by [72], where K is the number of arms. [69] proposed a structured greedy algorithm to achieve an $O(s\sqrt{n})$ upper bound. [102] derived a $\Omega(n^{2/3})$ minimax regret lower bound for sparse linear bandits where the feature vectors lack a well-conditioned exploration distribution.

There are many previous works studying feature selection in reinforcement learning. Specifically, [103, 104, 105, 106] proposed algorithms with ℓ_1-regularization for temporal-difference (TD) learning. [107] and [108] proposed Lasso-TD to estimate the value function in sparse reinforcement learning and derived finite-sample MDP statistical analysis. [109] provided nearly optimal statistical analysis of high dimensional batch reinforcement learning (RL) using sparse linear function approximation. [110] derived an $O(d\sqrt{n})$ regret bound in high-dimensional sparse linear quadratic systems where d is the dimension of the state space. The hardness of online reinforcement learning in fixed horizon has been studied by [111], which shows that linear regret is generally unavoidable in this case, even if there exists a policy that collects well-conditioned data.

4.3 Preliminary

Throughout this chapter, $f(n) = O(g(n))$ denotes that there exists a constant $c > 0$ such that $|f(n)| \leq c|g(n)|$ and $\tilde{O}(\cdot)$ ignores poly-logarithmic factors. $f(n) = \Omega(g(n))$ means that there exists a constant $c > 0$ such that $|f(n)| \geq c|g(n)|$. In addition, the notation $f(n) = \Theta(g(n))$ means that there exists constants $c_1, c_2 > 0$ such that $c_1|g(n)| \leq |f(n)| \leq c_2|g(n)|$. For a given integer n, let $[n]$ denote the set $\{1, \cdots, n\}$. Let $C > 0$ denote a suitably universal large constant. For a matrix $A \in \mathbb{R}^{m\times n}$, the set of rows is denoted by $\mathrm{rows}(A)$. Define an index set $\mathcal{M} \subseteq [d]$ such that $|\mathcal{M}| = s$. Let $\Phi_{\mathcal{M}} \in \mathbb{R}^{k\times s}$ be the submatrix of $\Phi \in \mathbb{R}^{k\times d}$ and $\theta_{\mathcal{M}} \in \mathbb{R}^s$ be the sub-vector of $\theta \in \mathbb{R}^d$.

Consider a bandit problem where the expected rewards are nearly a linear function of their associated features. Let $\Phi \in \mathbb{R}^{k\times d}$ denote the feature matrix whose rows are feature vectors corresponding to k actions. In rounds $t \in [n]$, the agent chooses actions $(a_t)_{t=1}^n$ with $a_t \in \mathrm{rows}(\Phi)$ and receives a reward

$$r_{a_t} = \langle a_t, \theta^* \rangle + \nu_{a_t}\,, \tag{4.1}$$

where $\nu_{a_t} \in [-\varepsilon, \varepsilon]$, $\varepsilon > 0$ for $t \in [n]$ and $\theta^* \in \mathbb{R}^d$ is an unknown parameter vector. We only consider *deterministic* rewards as small unbiased noises from rewards that do not change the sample complexity analysis of this chapter by much but complicate the presentation. In Appendix C.3, we provide additional discussion on the noisy setting of the rewards.

We make the mild boundedness assumption for each element of the feature matrix such that $\mathrm{rows}(\Phi) \in \mathbb{S}_B^{d-1}$. The parameter vector θ^* is assumed to be s-sparsity:

$$\|\theta^*\|_0 = \sum_{j=1}^{d} \mathbb{1}\{\theta_j^* \neq 0\} = s \text{ and } \|\theta^*\|_2 \leq 1.$$

We also assume that $\forall\, x \in \mathrm{rows}(\Phi)$, there is $\|x\|_2 \leq 1$.

4.4 Main Results

In this section, we first present an $O(\varepsilon)$-optimal algorithm that takes $O(\varepsilon^{-s}d^s)$ queries in Section 4.4.1 for ε-misspecified s-sparse linear bandit. Then we derive a nearly matching lower bound in Section 4.4.2.

4.4.1 An Algorithm that Breaks the $\Omega(\exp(d))$ Sample Barrier

The core idea of our algorithm is based on an elimination-type argument. In particular, we would guess an estimator $\hat{\theta}$ for θ^* and a index set $\mathcal{M} \subset [d]$. Then for each guess of $\hat{\theta}$ and $\mathcal{M}$, we check the actions that have similar features restricting to $\mathcal{M}$. Querying an action in this group allows us to rule out the guess of $\mathcal{M}$ and $\hat{\theta}$ if they were not correct. If the ground truth θ^* is dense, this algorithm would take $\Omega(\exp(d))$ queries. Fortunately, since $|\mathcal{M}| = s$, we can establish an $O(\varepsilon)$-net with a small size and eliminate the incorrect parameters efficiently. Below, we present the algorithm more formally.

Define an index set $\mathcal{M} \subseteq [d]$ such that $|\mathcal{M}| = s$. Let $\mathcal{M}^*$ denote the non-zero subset of θ^*. Denote $\mathcal{N}^s$ as a maximal $\varepsilon/2$-separated subset of the Euclidean sphere $\mathbb{S}^{s-1}$ with radius of 1 . The set $\mathcal{N}^s$ satisfies that $\|x - y\|_2 \geq \varepsilon/2$, for all $x, y \in \mathcal{N}^s$, and no subset of $\mathbb{S}^{s-1}$ containing $\mathcal{N}^s$

satisfies this condition. Thus, the size of $\mathcal{N}^s$ is

$$|\mathcal{N}^s| \le \left(\frac{4}{\varepsilon} + 1\right)^s. \tag{4.2}$$

For a set $\mathcal{M}$, we denote an estimator as $\hat{\theta}_{\mathcal{M}} \in \mathcal{N}^s$ to indicate the estimator which has only non-zero coordinates at $\mathcal{M}$.

For $\forall w \in \mathcal{N}^s$, we collect all $x \in \text{rows}(\Phi)$ close to w by the measurement $|\hat{\theta}_{\mathcal{M}}^\top(x_{\mathcal{M}} - w)|$ where $x_{\mathcal{M}} \in \mathbb{R}^s$ is the sub-vector of $x \in \mathbb{R}^d$ restricted to the index set $\mathcal{M}$ and define the set as

$$\mathcal{R}_{\mathcal{M}}^w(\hat{\theta}_{\mathcal{M}}) := \{x \in \text{rows}(\Phi) : |\hat{\theta}_{\mathcal{M}}^\top(x_{\mathcal{M}} - w)| \le \frac{\varepsilon}{2}\}. \tag{4.3}$$

The above set is simply denoted as $\mathcal{R}_{\mathcal{M}}^w$ in the following proof if $\hat{\theta}_{\mathcal{M}}$ is clear from the context. In each round of the algorithm, we find $x \in \mathcal{R}_{\mathcal{M}}^w$ and a set $\mathcal{M}'$ ($\mathcal{M}' \ne \mathcal{M}$) such that $\hat{\theta}_{\mathcal{M}'}^\top x_{\mathcal{M}'}$ deviates from $\hat{\theta}_{\mathcal{M}}^\top w$ (at least $\Omega(\varepsilon)$). Then, we query such x and receive the corresponding reward r_x. By comparing the difference between r_x and $\hat{\theta}_{\mathcal{M}}^\top w$, we can know whether the subset $\mathcal{M}$ or $\mathcal{M}'$ of x is more likely to determine the reward r_x and rule out the incorrect parameters. For $x \in \mathcal{R}_{\mathcal{M}}^w$, let $[x]_{\mathcal{N}^s}$ denote the vector $v = \arg\min_{w \in \mathcal{N}^s} \|w - x_{\mathcal{M}}\|_2$ where $x_{\mathcal{M}} \in \mathbb{R}^s$ is the sub-vector of x. Let $(\sim, \mathcal{M}, \hat{\theta}_{\mathcal{M}}) \in \mathcal{S}$ denote all of the elements involving the index set $\mathcal{M}$ and $\hat{\theta}_{\mathcal{M}} \in \mathcal{N}^s$. We present the full algorithm in Algorithm 2.

Theorem 2 *After*

$$O\left(\left(\frac{1}{\varepsilon}\right)^s \cdot \binom{d}{s}\right)$$

number of queries, the outputs of Algorithm 2, $\hat{\theta}_{\mathcal{L}}$ and $\mathcal{L}$, satisfy $|r_a - \langle a_{\mathcal{L}}, \hat{\theta}_{\mathcal{L}}\rangle| \le O(\varepsilon)$ for all $a \in \text{rows}(\Phi)$.

***Proof**: We first prove the correctness of the algorithm. Suppose for some $(w, \mathcal{M}, \hat{\theta}_{\mathcal{M}}) \in \mathcal{S}$, there is $x \in \mathcal{R}_{\mathcal{M}}^w$ such that $([x_{\mathcal{M}'}]_{\mathcal{N}^s}, \mathcal{M}', \hat{\theta}_{\mathcal{M}'}) \in \mathcal{S}$ and $|\langle x_{\mathcal{M}'}, \hat{\theta}_{\mathcal{M}'}\rangle - \langle w, \hat{\theta}_{\mathcal{M}}\rangle| > 5\varepsilon/2$ and $\mathcal{M}' \ne \mathcal{M}$. Consider two cases in Lines 4-7 in Algorithm 2.*

- *Case 1: Suppose $|r_x - \langle w, \hat{\theta}_{\mathcal{M}}\rangle| \le 3\varepsilon/2$, then we have that $|r_x - \langle x_{\mathcal{M}}, \hat{\theta}_{\mathcal{M}}\rangle| \le 2\varepsilon$ and $|r_x - \langle x_{\mathcal{M}'}, \hat{\theta}_{\mathcal{M}'}\rangle| \ge |\langle x_{\mathcal{M}'}, \hat{\theta}_{\mathcal{M}'}\rangle - \langle w, \hat{\theta}_{\mathcal{M}}\rangle| - |r_x - \langle w, \hat{\theta}_{\mathcal{M}}\rangle| > \varepsilon$. Thus after the iterations, for*

Algorithm 2: Parameter Elimination

```
1: Input: feature matrix $\Phi \in \mathbb{R}^{k\times d}$.
2: Initialize: $\mathcal{S} := \{(w,\mathcal{M},\hat{\theta}_{\mathcal{M}}) : w \in \mathcal{N}^s, \mathcal{M} \subseteq [d], |\mathcal{M}| = s, \hat{\theta}_{\mathcal{M}} \in \mathcal{N}^s\}$.
3: For each $(w,\mathcal{M},\hat{\theta}_{\mathcal{M}}) \in \mathcal{S}$, establish $\mathcal{R}^w_{\mathcal{M}}$ as (4.3).
4: while there exit $(w,\mathcal{M},\hat{\theta}_{\mathcal{M}}) \in \mathcal{S}$, $\mathcal{M}' \subseteq [d], |\mathcal{M}'| = s$, $\mathcal{M} \neq \mathcal{M}'$, and $x \in \mathcal{R}^w_{\mathcal{M}}$ such that $(\sim,\mathcal{M}',\hat{\theta}_{\mathcal{M}'}) \in \mathcal{S}$, $|\langle x_{\mathcal{M}'},\hat{\theta}_{\mathcal{M}'}\rangle - \langle w,\hat{\theta}_{\mathcal{M}}\rangle| > 5\varepsilon/2$ do
5:     Query the action $x$ and receive a reward $r_x = \langle x,\theta^*\rangle + \nu_x$ where $\nu_x \in [-\varepsilon,\varepsilon]$.
6:     If $|r_x - \langle w,\hat{\theta}_{\mathcal{M}}\rangle| > 3\varepsilon/2$ then $\mathcal{S} = \mathcal{S}\backslash(\sim,\mathcal{M},\hat{\theta}_{\mathcal{M}})$, otherwise $\mathcal{S} = \mathcal{S}\backslash(\sim,\mathcal{M}',\hat{\theta}_{\mathcal{M}'})$.
7: end while
8: Find a certain set $\mathcal{L} \subseteq [d], |\mathcal{L}| = s$ and corresponding $\hat{\theta}_{\mathcal{L}} \in \mathcal{N}^s$ such that $(\sim,\mathcal{L},\hat{\theta}_{\mathcal{L}}) \in \mathcal{S}$.
9: Output: $\hat{\theta}_{\mathcal{L}}$ and $\mathcal{L}$
```

some $(w,\mathcal{M},\hat{\theta}_{\mathcal{M}}) \in S$ and $x \in \mathcal{R}^w_{\mathcal{M}}$, we have $|r_x - \langle x_{\mathcal{M}},\hat{\theta}_{\mathcal{M}}\rangle| \leq 2\varepsilon$. We remove the elements $(\sim,\mathcal{M}',\hat{\theta}_{\mathcal{M}'})$ from $\mathcal{S}$ since there exists an $x \in \mathrm{rows}(\Phi)$ such that $|r_x - \langle x_{\mathcal{M}'},\hat{\theta}_{\mathcal{M}'}\rangle| > \varepsilon$.

- *Case 2: Assume that $|r_x - \langle w,\hat{\theta}_{\mathcal{M}}\rangle| > 3\varepsilon/2$ for some $x \in \mathcal{R}^w_{\mathcal{M}}$. Then the elements $(\sim,\mathcal{M},\hat{\theta}_{\mathcal{M}})$ get removed from $\mathcal{S}$ since there exists an $x \in \mathrm{rows}(\Phi)$ such that $|r_x - \langle x_{\mathcal{M}},\hat{\theta}_{\mathcal{M}}\rangle| \geq |r_x - \langle w,\hat{\theta}_{\mathcal{M}}\rangle| - |\langle x_{\mathcal{M}},\hat{\theta}_{\mathcal{M}}\rangle - \langle w,\hat{\theta}_{\mathcal{M}}\rangle| > \varepsilon$.*

Moreover, Algorithm 2 guarantees that

- *The elements $(\sim,\mathcal{M}^*,[\theta^*{}_{\mathcal{M}^*}]_{\mathcal{N}^s})$ maintain in the set $\mathcal{S}$, which involves the ground-truth index set $\mathcal{M}^*$ and $[\theta^*{}_{\mathcal{M}^*}]_{\mathcal{N}^s} \in \mathcal{N}^s$ such that $|r_x - \langle x_{\mathcal{M}^*},[\theta^*{}_{\mathcal{M}^*}]_{\mathcal{N}^s}\rangle| \leq \varepsilon$. Algorithm 2 only eliminates elements $(\sim,\mathcal{M},\hat{\theta}_{\mathcal{M}})$ involving the index set $\mathcal{M}$ and $\hat{\theta}_{\mathcal{M}}$ such that $|r_x -$*

$\langle x_{\mathcal{M}}, \hat{\theta}_{\mathcal{M}}\rangle| > \varepsilon$ *for some* $x \in \mathrm{rows}(\Phi)$.

- *If no more pairs in the remaining set* $\mathcal{S}$ *satisfies the conditions on Line 4 in Algorithm 2, then it must be the case that, for all* $(w, \mathcal{M}, \hat{\theta}_{\mathcal{M}}) \in \mathcal{S}$ *with the remaining set* $\mathcal{S}$ *and* $\forall\, x \in \mathcal{R}^w_{\mathcal{M}}$*,* $|\langle x_{\mathcal{M}^*}, [\theta^*{}_{\mathcal{M}^*}]_{\mathcal{N}^s}\rangle - \langle w, \hat{\theta}_{\mathcal{M}}\rangle| \le 5\varepsilon/2$*, and hence*

$$\begin{aligned}&|r_x - \langle w, \hat{\theta}_{\mathcal{M}}\rangle| = |\langle x, \theta^*\rangle + \nu_x - \langle w, \hat{\theta}_{\mathcal{M}}\rangle| \\ \le&|\langle x_{\mathcal{M}^*}, [\theta^*{}_{\mathcal{M}^*}]_{\mathcal{N}^s}\rangle - \langle w, \hat{\theta}_{\mathcal{M}}\rangle| + \varepsilon \le 7\varepsilon/2,\end{aligned} \tag{4.4}$$

Moreover,

$$|r_x - \langle x_{\mathcal{M}}, \hat{\theta}_{\mathcal{M}}\rangle| \le |r_x - \langle w, \hat{\theta}_{\mathcal{M}}\rangle| + |\hat{\theta}^\top_{\mathcal{M}}(x_{\mathcal{M}} - w)| \le 4\varepsilon.$$

In summary, for a set $\mathcal{L} \subseteq [d], |\mathcal{L}| = s$ *and corresponding* $\hat{\theta}_{\mathcal{L}} \in \mathcal{N}^s$ *such that* $(\sim, \mathcal{L}, \hat{\theta}_{\mathcal{L}})$ *in the remaining set* $\mathcal{S}$*, we can guarantee that*

$$|r_x - \langle x_{\mathcal{L}}, \hat{\theta}_{\mathcal{L}}\rangle| \le 4\varepsilon,$$

for $\forall\, x \in \mathrm{rows}(\Phi)$.

We arrive at the sample complexity analysis of the algorithm. If we find $(w, \mathcal{M}, \hat{\theta}_{\mathcal{M}}) \in \mathcal{S}$*,* $\mathcal{M}' \neq \mathcal{M}$*,* $x \in \mathcal{R}^w_{\mathcal{M}}$ *satisfying the condition on Line 4 in Algorithm 2, we remove either the elements either* $(\sim, \mathcal{M}, \hat{\theta}_{\mathcal{M}})$ *or* $(\sim, \mathcal{M}', \hat{\theta}_{\mathcal{M}'})$ *after querying one action. The loop stops when the condition on Line 4 is not satisfied. Thus, at most* $|\mathcal{N}^s|\binom{d}{s}$ *queries are needed for the algorithm. Recall* $|\mathcal{N}^s|$ *(4.2), the number of queries in* s*-sparsity case can be bounded by*

$$O\left(\left(\frac{1}{\varepsilon}\right)^s \cdot \binom{d}{s}\right).$$

□

When s is a fixed constant, the above theorem demonstrates that $\mathrm{poly}(d/\varepsilon)$-queries are sufficient to learn an $O(\varepsilon)$-optimal action. This is in stark contrast to the $\Omega(\exp(d))$ lower-bound provided in [65] and [66]. When s is not fixed, the dependence on $\exp(s)$ is undesirable. One may ask, whether it is possible to achieve $\mathrm{poly}(s)$-dependence for some cases, e.g., relaxed error $s^\delta\varepsilon$ for some $\delta > 0$. Unfortunately, the next section provides a lower bound that rules out the possibility for $\delta < 1$.

4.4.2 Lower Bound

In this section, we establish an information-theoretical lower bound to show that our upper bound is nearly tight. The basic idea is by reduction to the INDEX-QUERY problem [65, 112] using statistical analysis on sub-exponential random variables. More formally, it is shown [65] that if one is given a vector of dimension n with only one non-zero entry, then it is necessary to query $\Omega(pn)$ entries of the vector to output the index of the entry with probability p. In what follows, we can show that for any algorithm that solves an s-sparse ε-misspecified linear bandit problem, we can use it to solve the INDEX-QUERY problem of size $\Omega(\exp(s))$. The idea is to establish a set of sparse vectors with sub-exponential random variables, such that the vector input to the INDEX-QUERY problem can be embedded into the bandit instance (without any queries to the vector).

The next lemma is the key tool that will be useful in our lower-bound arguments. It shows that there exists a sparse matrix $\Phi \in \mathbb{R}^{k\times d}$ with sufficiently large k where rows have unit norm and sparsity s, and all non-equal rows are almost orthogonal.

Lemma 1 *For* $0 < \delta < 1$*,* $c > 1$ *and* $C' = \frac{2c^3}{(1+\tau)\sqrt{c^2-1}}$ *with sufficiently small* $0 \le \tau < 1$*,*

- *if* $0 < \varepsilon \le \frac{C's}{d}$*, by choosing* $k \ge \sqrt{\delta}\exp\left(\frac{d(1+\tau)\varepsilon^2}{4C'}\right)$*,*
- *if* $\varepsilon > \frac{C's}{d}$*, by choosing* $k \ge \sqrt{\delta}\exp\left(\frac{s(1+\tau)\varepsilon}{4}\right)$*,*

there exists a feature matrix $\Phi \in \mathbb{R}^{k\times d}$ *with rows such that for all* $a, b \in \mathrm{rows}(\Phi)$ *with* $a \neq b$*,* $\|a\|_2 = 1$*,* $\|a\|_0 \le s$*, and* $|\langle a, b\rangle| \le \varepsilon$*.*

Proof*:[Proof Sketch] The matrix is established by choosing each entry of the matrix* Φ *a small probability (*$\sim s/d$*) to be non-zero and if it is non-zero, the entry follows a Gaussian distribution. The formal proof is provided in Appendix C.1.* □

As we will show shortly, the matrix in Lemma 1 can be used to *agnostically* embed an arbitrary INDEX-QUERY problem to a sparse misspecified instance. To start with the formal reduction, we introduce the definition of (η, Δ)-sound algorithm for linear bandit problem, where the algorithm returns an estimated optimal action $\hat{a} \in \mathrm{rows}(\Phi)$ and an estimation vector $\hat{\theta} \in \mathbb{R}^d$.

Definition 1 *For any $0 < \eta < 1$ and the approximation error $\Delta \geq \varepsilon$, an algorithm $\mathcal{A}$ solving linear bandit problem is called sound for (η, Δ) if with probability at least $1 - \eta$, algorithm $\mathcal{A}$ returns the estimated optimal action $\hat{a}$ such that $r_{\hat{a}} \geq \max_x r_x - \Delta$.*

For any input vector v to the INDEX-QUERY problem (of dimension k) with some unknown index j to be non-zero, we can simply take Φ as the feature matrix, and the j-th row of Φ to be the ground-truth θ^*. Then we would have $\|v - \Phi\theta^*\|_\infty \leq \varepsilon$. Thus any (η, Δ)-sound algorithm for some appropriate Δ would identify the non-zero index in v with good probability and thus inherits the lower bound of INDEX-QUERY. The formal lower bound is presented in the following theorem.

Theorem 3 *For any (η, Δ)-sound linear bandit algorithm $\mathcal{A}$, there exists an s-sparse ε-misspecified linear bandit instance such that algorithm $\mathcal{A}$ takes at least*

$$(1-\eta)\exp\left(c_0 d \cdot \left(\frac{\varepsilon}{\Delta}\right)^2\right), \textit{ if } 0 < \frac{\varepsilon}{\Delta} \leq \frac{C's}{d}, \tag{4.5}$$

$$(1-\eta)\exp\left(\frac{c_1 s(1+\tau)\varepsilon}{\Delta}\right), \textit{ if } \frac{\varepsilon}{\Delta} > \frac{C's}{d}, \tag{4.6}$$

actions to halt, where c_0, c_1, C' are absolute constants.

***Proof**: We begin with the construction of the hard s-sparsity instances. Consider an INDEX-QUERY problem with dimension k. Suppose the input vector with the i^*-index (unknown to the algorithm) is non-zero, i.e., e_{i^*}. Here, e_i is the standard unit vector with the i-th coordinate equaling 1. In our hard instance, we choose reward $r_x = 2\Delta$ when $x = a_{i^*}$ with $i^* \in [k]$, otherwise is 0. Now we show that there exists a linear feature representation that approximates the reward vector $\Delta e_{i^*} \in \mathbb{R}^k$ with a uniform error. Based on Lemma 1, let Φ be the matrix $\mathrm{rows}(\Phi) = (a_i)_{i=1}^k$ such that for all $a_i, a_j \in \mathrm{rows}(\Phi)$ with $i \neq j$, $\|a_i\|_2 = 1$ and $|\langle a_i, a_j\rangle| \leq \varepsilon/(2\Delta)$. With $\theta^* = 2\Delta a_{i^*}$, we have $\Phi\theta^* = (2\Delta a_1^\top a_{i^*}, \ldots, 2\Delta a_{i^*}^\top a_{i^*}, \ldots, 2\Delta a_k^\top a_{i^*})^\top$. By choice of Φ, the i^*-th component of $\Phi\theta^*$*

is Δ and the others are all less than ε in absolute value. Hence, we can represent the reward vector $2\Delta e_{i^*}$ by $2\Delta e_{i^*} = \Phi\theta^* + \nu$ for some $\nu \in [-\varepsilon, \varepsilon]^k$.

Then an (η, Δ)-sound algorithm would identify an action a, such that with probability at least $1 - \eta$, $a^\top \theta^* \geq 2\Delta - \Delta = \Delta$, which is only possible if $a = a_{i^*}$. Hence the algorithm would output i^* with probability at least $1 - \eta$. By the lower bound of the INDEX query problem (e.g., Theorem A1 in [65]), the algorithm takes at least $\Omega((1 - \eta)k)$ queries in the worst-case.

In the construction, we only need Lemma 1 to hold for k with the correct parameters. Hence we have

- *if $0 < \varepsilon \leq \frac{C's}{d}$, then $k \geq \sqrt{\delta} \exp\left(\frac{d(1+\tau)\varepsilon^2}{16C'\Delta^2}\right)$, and*
- *if $\varepsilon > \frac{C's}{d}$, then $k \geq \sqrt{\delta} \exp\left(\frac{s(1+\tau)\varepsilon}{8\Delta}\right)$,*

for constant τ, δ, and C', completing the proof.

$\square$

Remark 1 *The above theorem shows that even if we relax the approximation error to $s^\delta \varepsilon$ for some $0 < \delta < 1$, the $\exp(s)$ dependence is unavoidable. Hence our upper bound in the previous section is nearly tight. However, this lower bound does not rule out the improvement in terms of ε^{-s} and efficient regimes when $\Delta = \Omega(s^\delta \varepsilon)$ for some $\delta \geq 1$. We will explore both settings in the rest of the paper.*

4.5 Improvement on the ε^{-s} Dependence

Even though the dependence of d^s is unavoidable, we can improve the upper bound in Theorem 2 by eluding the dependence of ε. The fundamental idea of the improved algorithm is based on a mix of G-optimal design and elimination argument. Instead of guessing an estimator $\hat{\theta}$ for θ^*, we use G-optimal design to estimate $\hat{\theta}$ concerning an index set $\mathcal{M} \subset [d]$. Then for each estimator $\hat{\theta}$ and $\mathcal{M}$, we check the actions that have similar features restricting to $\mathcal{M}$. The rest of the elimination argument is similar to Section 4.4.1. Yet the optimal G-optimal design only gives an error guarantee of $O(\varepsilon\sqrt{s})$, which worsens our error guarantee. Below, we present the algorithm more formally.

We start with an essential theorem in G-optimal design which shows that there exists a near-optimal design with a small core set.

Theorem 4 ([113]) *Given a matrix $A \in \mathbb{R}^{k\times s}$ and a probability distribution $\rho : \mathrm{rows}(A) \to [0,1]$, let $G(\rho) \in \mathbb{R}^{s\times s}$[1] and $g(\rho) \in \mathbb{R}$ be given by*

$$G(\rho) = \sum_{a\in\mathrm{rows}(A)} \rho(a)aa^\top , \qquad g(\rho) = \max_{a\in\mathrm{rows}(A)} \|a\|^2_{G(\rho)^{-1}} .$$

There exists a probability distribution ρ such that $g(\rho) \le 2m$ and the size of the support of ρ is at most $4s\log\log(s) + 16$.

Remark 2 *The distribution satisfying the results in Theorem 4 can be computed by the Frank-Wolfe algorithm introduced in [113][Chapter 3] after $O(ks^2)$ computations.*

Let $\mathcal{S} \subset [d]^s$ be all the subsets of cardinality s. For each $\mathcal{M} \in \mathcal{S}$, suppose that $\rho_{\mathcal{M}}$ is a probability distribution over $\mathrm{rows}(\Phi_{\mathcal{M}})$ satisfying the results of Theorem 4, where $\Phi_{\mathcal{M}} \in \mathbb{R}^{k\times s}$ is the sub-matrix of $\Phi \in \mathbb{R}^{k\times d}$. In the following, we use $G_{\mathcal{M}}(\rho_{\mathcal{M}})$ to present $G(\rho)$ defined in Theorem 4 with respect to $\mathcal{M}$. We begin with querying actions to estimate $\hat{\theta}_{\mathcal{M}}$ based on the support of $\rho_{\mathcal{M}}$ and obtain rewards:

$$\hat{\theta}_{\mathcal{M}} = G_{\mathcal{M}}(\rho_{\mathcal{M}})^{-1} \sum_{a\in\mathrm{rows}(\Phi_{\mathcal{M}}),\rho_{\mathcal{M}}(a)\neq 0} \rho_{\mathcal{M}}(a)r_a a, \tag{4.7}$$

With Theorem 4, we can show that, for all $b \in \mathrm{rows}(\Phi)$ and $\lceil 4s\log\log(s) + 16\rceil$ queries, we have

$$|\langle b_{\mathcal{M}^*}, \hat{\theta}_{\mathcal{M}^*}\rangle - \langle b, \theta^*\rangle| \le \varepsilon\sqrt{2s}, \tag{4.8}$$

[1]*Without loss of generality, we assume $G(\rho)$ is invertible in the rest of the paper. If not, we can discard columns in Φ until the Φ is full column rank.*

where $b_{\mathcal{M}^*} \in \mathbb{R}^s$ is the sub-vector of $b \in \mathbb{R}^d$. For $\mathcal{M}, \mathcal{M}' \in \mathcal{S}$, we try to find some $x \in \mathrm{rows}(\Phi)$ making $\hat{\theta}_{\mathcal{M}'}^{\top} x_{\mathcal{M}'}$ deviate from $\hat{\theta}_{\mathcal{M}}^{\top} x_{\mathcal{M}}$. We query such x and receive the corresponding reward r_x. By comparing the difference between r_x and $\hat{\theta}_{\mathcal{M}}^{\top} x_{\mathcal{M}}, \hat{\theta}_{\mathcal{M}'}^{\top} x_{\mathcal{M}'}$, we can know whether the subset $\mathcal{M}$ or $\mathcal{M}'$ of x is more likely to determine the reward r_x, and hence eliminate the incorrect parameter-set. The full algorithm is presented in Algorithm 3.

Algorithm 3: (ε^{-s})-Free Algorithm

```
1: **Input:** feature matrix $\Phi \in \mathbb{R}^{k \times d}$.
2: **Initialize:** $\mathcal{S} := \{\mathcal{M} : \mathcal{M} \subseteq [d], |\mathcal{M}| = s\}$.
3: For each $\mathcal{M} \in \mathcal{S}$, estimate $\hat{\theta}_{\mathcal{M}}$ based on (4.7).
4: **while** there exit $\mathcal{M}, \mathcal{M}' \in \mathcal{S}$, $\mathcal{M} \neq \mathcal{M}'$, and $x \in \mathrm{rows}(\Phi)$ such
   that $|\langle x_{\mathcal{M}'}, \hat{\theta}_{\mathcal{M}'} \rangle - \langle x_{\mathcal{M}}, \hat{\theta}_{\mathcal{M}} \rangle| > 2\varepsilon(1 + \sqrt{2s})$ **do**
5:     Query the action $x$ and receive a reward $r_x = \langle x, \theta^* \rangle + \nu_x$
       where $\nu_x \in [-\varepsilon, \varepsilon]$.
1
6:     If $|r_x - \langle x_{\mathcal{M}}, \hat{\theta}_{\mathcal{M}} \rangle| \leq \varepsilon(1 + \sqrt{2s})$ then $\mathcal{S} = \mathcal{S} \backslash \mathcal{M}'$.
7:     Otherwise $\mathcal{S} = \mathcal{S} \backslash \mathcal{M}$, if $|r_x - \langle x_{\mathcal{M}'}, \hat{\theta}_{\mathcal{M}'} \rangle| > \varepsilon(1 + \sqrt{2s})$ then
       $\mathcal{S} = \mathcal{S} \backslash \mathcal{M}'$.
8: **end while**
9: Find a certain set $\mathcal{L} \subseteq [d], |\mathcal{L}| = s$ such that $\mathcal{L} \in \mathcal{S}$ and estimate
   $\hat{\theta}_{\mathcal{L}} \in \mathbb{R}^s$.
10: **Output:** $\hat{\theta}_{\mathcal{L}}$ and $\mathcal{L}$
```

Theorem 5 *After*

$$O\left(s \log s \cdot \binom{d}{s}\right)$$

number of queries, the outputs of Algorithm 3, $\hat{\theta}_{\mathcal{L}}$ and $\mathcal{L}$, satisfy $|r_a - \langle a_{\mathcal{L}}, \hat{\theta}_{\mathcal{L}} \rangle| \leq O(\varepsilon\sqrt{s})$ for all $a \in \mathrm{rows}(\Phi)$.

Proof: *We first prove the correctness of the algorithm. Suppose we find some $\mathcal{M}, \mathcal{M}' \in \mathcal{S}$,*

$\mathcal{M} \neq \mathcal{M}'$, and $x \in \mathrm{rows}(\Phi)$ $|\langle x_{\mathcal{M}'}, \hat{\theta}_{\mathcal{M}'}\rangle - \langle x_{\mathcal{M}}, \hat{\theta}_{\mathcal{M}}\rangle| > 2\varepsilon(1+\sqrt{2s})$. Consider two cases in Lines 4-8 in Algorithm 3.

- *Case 1: Suppose we have $|r_x - \langle x_{\mathcal{M}}, \hat{\theta}_{\mathcal{M}}\rangle| \leq \varepsilon(1+\sqrt{2s})$. We remove the element $\mathcal{M}'$ from $\mathcal{S}$ since there exists an $x \in \mathrm{rows}(\Phi)$ such that $|r_x - \langle x_{\mathcal{M}'}, \hat{\theta}_{\mathcal{M}'}\rangle| \geq |\langle x_{\mathcal{M}'}, \hat{\theta}_{\mathcal{M}'}\rangle - \langle x_{\mathcal{M}}, \hat{\theta}_{\mathcal{M}}\rangle| - |r_x - \langle x_{\mathcal{M}}, \hat{\theta}_{\mathcal{M}}\rangle| > \varepsilon(1+\sqrt{2s})$.*

- *Case 2: Assume that $|r_x - \langle x_{\mathcal{M}}, \hat{\theta}_{\mathcal{M}}\rangle| > \varepsilon(1+\sqrt{2s})$, then the element $\mathcal{M}$ gets removed from $\mathcal{S}$. We can also remove the other index set $\mathcal{M}'$ from $\mathcal{S}$ if $|r_x - \langle x_{\mathcal{M}'}, \hat{\theta}_{\mathcal{M}'}\rangle| > \varepsilon(1+\sqrt{2s})$.*

Moreover, Algorithm 3 guarantees that

- *The ground-truth index set $\mathcal{M}^*$ maintains in the set $\mathcal{S}$. According to (4.8), for all $x \in \mathrm{rows}(\Phi)$, we have $|r_x - \langle x_{\mathcal{M}^*}, \hat{\theta}_{\mathcal{M}^*}\rangle| \leq \varepsilon(1+\sqrt{2s})$. Algorithm 3 only eliminates $\mathcal{M}$ such that $|r_x - \langle x_{\mathcal{M}}, \hat{\theta}_{\mathcal{M}}\rangle| > \varepsilon(1+\sqrt{2s})$ for some $x \in \mathrm{rows}(\Phi)$. After each query, Algorithm 3 removes at least one element from $\mathcal{S}$.*

- *If no more pair in the remaining set $\mathcal{S}$ satisfies the conditions on Line 4 in Algorithm 3, then it must be the case that, for all $\mathcal{M} \in \mathcal{S}$ with the remaining set $\mathcal{S}$ and $\forall\, x \in \mathrm{rows}(\Phi)$, $|\langle x_{\mathcal{M}^*}, \hat{\theta}_{\mathcal{M}^*}\rangle - \langle x_{\mathcal{M}}, \hat{\theta}_{\mathcal{M}}\rangle| \leq 2\varepsilon(1+\sqrt{2s})$. According to (4.8), we have*

$$\begin{aligned}
&|r_x - \langle x_{\mathcal{M}}, \hat{\theta}_{\mathcal{M}}\rangle| \\
\leq &|r_x - \langle x_{\mathcal{M}^*}, \hat{\theta}_{\mathcal{M}^*}\rangle| + |\langle x_{\mathcal{M}^*}, \hat{\theta}_{\mathcal{M}^*}\rangle - \langle x_{\mathcal{M}}, \hat{\theta}_{\mathcal{M}}\rangle| \\
\leq &3\varepsilon(1+\sqrt{2s}). \qquad (4.9)
\end{aligned}$$

In summary, for a set $\mathcal{L} \subseteq [d], |\mathcal{L}| = s$ in the remaining set $\mathcal{S}$ and the estimation $\hat{\theta}_{\mathcal{L}} \in \mathbb{R}^s$, we can guarantee that

$$|r_x - \langle x_{\mathcal{L}}, \hat{\theta}_{\mathcal{L}}\rangle| \leq 3\varepsilon(1+\sqrt{2s}),$$

for $\forall\, x \in \mathrm{rows}(\Phi)$.

We arrive at the sample complexity analysis of the algorithm. The estimation on Line 3 in Algorithm 3 takes $\lceil 4s\log\log(s)+16\rceil\binom{d}{s}$ queries. If we find $\mathcal{M},\mathcal{M}'\in\mathcal{S}$, $\mathcal{M}\neq\mathcal{M}'$, and $x\in\mathrm{rows}(\Phi)$ satisfying the condition on Line 4 in Algorithm 3, we remove at least one element from $\mathcal{M},\mathcal{M}'$ after querying one action. The loop stops when the condition on Line 4 is not satisfied. Thus, the number of queries in the s-sparsity case can be bounded by

$$O\left(s\log s\cdot\binom{d}{s}\right).$$

□

4.6 A $\mathrm{poly}(s)$-Query Algorithm for Benign Features

The lower bound derived in Section 4.4.2 does not rule out the possibility of $\exp(s)$-free bound when $\Delta = O(s^{\delta}\varepsilon)$ for $\delta\geq 1$, which we attempt to achieve in this section. The core idea of our algorithm is based on feature compression followed by action-elimination bandit learning. Specifically, we compress the feature vectors and the sparse parameter vector to a lower dimensional vector space, thus converting the sparse linear bandits to a dense case with much lower dimensional features. Note that this compression is agnostic to the ground-truth parameters. Then we implement action-elimination learning in compressed linear bandits. The detailed algorithm is provided in the following.

We here consider the finite setting where the number of rows, k, in the feature matrix Φ is finite (recall the definition in (4.1)). This argument is without loss of generality as we can always find an ε-net to cover the actions if there are infinitely many. By Johnson-Linderstrauss lemma [114], we have that for some $p=\Theta(\log(k)/\upsilon^2)$, there is a function $f:\mathbb{R}^d\to\mathbb{R}^p$ that preserves inner product, i.e., for each $a\in\mathrm{rows}(\Phi)$,

$$\langle f(a), f(\theta^*)\rangle = \langle a,\theta^*\rangle \pm 2\upsilon, \tag{4.10}$$

for some error $\upsilon>0$. Such a function can be found efficiently using techniques in, e.g., [115]. Hence, we transform the previous sparse linear model $\langle a,\theta^*\rangle$ where $a,\theta^*\in\mathbb{R}^d$ to a new linear model $\langle f(a), f(\theta^*)\rangle$ where $f(a), f(\theta^*) \in \mathbb{R}^p$ with $p < d$. We apply G-optimal design mentioned in (4.7) to get an estimation of $f(\theta^*)$, i.e., $\hat{\theta}_f$. The detailed algorithm is illustrated in Algorithm 4 where $C > 0$ is a suitable large constant.

Algorithm 4: $\mathrm{poly}(s)$-Query Algorithm for Benign Features

```
1: **Input:** feature matrix $\Phi \in \mathbb{R}^{k\times d}$, function $f : \mathbb{R}^d \to \mathbb{R}^p$ (4.10),
   the total time steps $n$.
2: **Initialize:** $\mathcal{S} := \{f(a) \in \mathbb{R}^p : a \in \mathrm{rows}(\Phi)\}$.
3: **while** number of queries is no greater than $n$ **do**
4:     Compute the probability distribution $\rho : \mathcal{S} \to [0, 1]$ satisfying
       the results of Theorem 4.
5:     Compute $\hat{\theta}_f = G(\rho)^{-1} \sum_{a\in\mathcal{S}} \rho(a) r_a a$ where the reward $r_a$ is
       received by querying the action $a$.
6:     Update active action set:
       $$\mathcal{S} \leftarrow \left\{ a \in \mathcal{S} : \max_{b\in\mathcal{S}} \langle \hat{\theta}_f, b - a\rangle \le C \cdot (\log k)^{\frac{1}{4}} \sqrt{\varepsilon} \right\}.$$
7: **end while**
8: **Output:** $\hat{\theta}_f \in \mathbb{R}^p$
```

Theorem 6 *Suppose there is a function $f : \mathbb{R}^d \to \mathbb{R}^p$ satisfied (4.10). After $n \ge O\left(\sqrt{\log k}/\varepsilon\right)$ number of queries, the output of Algorithm 4 satisfies $|r_a - \langle f(a), \hat{\theta}_f\rangle| \le O((\log k))^{1/4}\sqrt{\varepsilon} + \varepsilon)$ for all $a \in \mathrm{rows}(\Phi)$.*

Proof:*[Proof Sketch] We start with the approximation error of $f(\theta^*)$.*

Similar to the G-optimal design in Section 4.5, we have

$$\begin{aligned} &|\langle f(a), \hat{\theta}_f\rangle - \langle a, \theta^*\rangle| \\ \le &|\langle f(a), \hat{\theta}_f\rangle - \langle f(a), f(\theta^*)\rangle| + |\langle f(a), f(\theta^*)\rangle - \langle a, \theta^*\rangle|, \end{aligned} \tag{4.11}$$

for $\forall\ a \in \mathrm{rows}(\Phi)$.

The first term in (4.11) can be termed as misspecified linear bandits in $\mathbb{R}^p$*. Similar to (4.8), the first term in (4.11) can be bounded by*

$$|\langle f(a), \hat{\theta}_f\rangle - \langle f(a), f(\theta^*)\rangle| \leq C\left(\varepsilon\sqrt{p}\right), \tag{4.12}$$

with $O(p \log p)$ *number of queries, where* $C > 0$ *is a suitably universal large constant. The second term in (4.11) can be bounded by* 2υ*. Hence, we have*

$$|\langle f(a), \hat{\theta}_f\rangle - \langle a, \theta^*\rangle| \leq C\left(\varepsilon\sqrt{p} + \upsilon\right), \tag{4.13}$$

Recall that $p = \Theta(\log(k)/\upsilon^2)$*, thus* $C(\varepsilon\sqrt{p} + \upsilon)$ *can be presented as an function with respect to* υ *(*$\upsilon > 0$*), given by*

$$g(\upsilon) = C(\varepsilon\sqrt{\log(k)/\upsilon^2} + \upsilon).$$

By optimizing $g(\upsilon)$ *with respect to* υ*, we have the approximation error of* $O((\log k)^{\frac{1}{4}}\sqrt{\varepsilon})$ *achieved by the number of queries* $O(\sqrt{\log k}/\varepsilon)$*.*

We can derive the final approximate error as

$$|\langle f(a), \hat{\theta}_f\rangle - \langle a, \theta^*\rangle| \leq C\left((\log k)^{\frac{1}{4}}\sqrt{\varepsilon}\right). \tag{4.14}$$

□

Corollary 1 *Based on the notations in Theorem 6, if setting* $\upsilon = O(s^\delta \varepsilon)$ *for* $\delta \geq 1$*, the number of queries* $O(s^{1+\delta})$ *can be achieved whenever* $\log k \leq \varepsilon^2 s^{2(1+\delta)}$*. Additionally, the regret of Algorithm 4 is bounded by* $O(s^\delta \varepsilon n \log n)$*.*

According to Corollary 1, when the coefficient $s^\delta < \sqrt{d}$, Algorithm 4 can break the $\varepsilon\sqrt{d}$ barrier with polynomial queries in all parameters if $\log k$ is small, which is achievable if the feature space possesses certain benign structures. For instance, the features are (close to) sparse as the instance in our lower bound construction. This also demonstrates that this result may not admit additional improvement as it resolves the lower bound instance.

All results above focus on the noiseless case. We further give a discussion on the noisy cases in Appendix C.3.

4.7 A $\mathrm{poly}(s)$-Query Algorithm for General Features

Theorem 6 presents an algorithm with sample complexity dependent on $\log k$ where k is the number of actions. Corollary 1 shows that it is possible to achieve sample complexity of $\mathrm{poly}(s)$ when k satisfies the condition $\log k \leq \varepsilon^2 s^{2(1+\delta)}$ for $\delta \geq 1$. However, to accommodate a wider range of scenarios, we aim for a sample complexity with a better dependence. In the following section, we will describe the method for achieving a sample complexity dependent on $\mathrm{poly}(s)$ for general features leveraging the ideas in Section 4.6.

The core idea of our algorithm is to select a submatrix $\Psi \in \mathbb{R}^{k' \times d}$ from the feature matrix $\Phi \in \mathbb{R}^{k \times d}$ where $k' < k$. The submatrix Ψ should contain enough representative actions, which we obtain by using G-optimal design concerning all possible s-subset $\mathcal{M}' \subset [d]$ as (4.7) and collecting the corresponding actions $a \in \mathbb{R}^d$. Then, we apply the same compression process as Section 4.6 to reduce the dimensionality of the feature matrix Ψ and the sparse parameter vector θ^*. Finally, we estimate the s-sparsity parameter θ^* based on the above information. This method consists of three main steps:

1. **G-optimal design:** For each $\mathcal{M}' \subset [d]$ such that $|\mathcal{M}'| = s$, we first find a probability distribution $\rho_{\mathcal{M}'}$ over $\mathrm{rows}(\Phi_{\mathcal{M}'})$ that meets the conditions of Theorem 4. We then use this distribution $\rho_{\mathcal{M}'}$ to generate $z := 4s \log\log(s) + 16$ distinct feature vectors $a_{\mathcal{M}'} \in \mathbb{R}^s$. We collect all the corresponding actions $a \in \mathbb{R}^d$ and denote them as $\Psi \in \mathbb{R}^{\binom{d}{s} \cdot z \times d}$.

2. **Compression:** By Johnson-Linderstrauss lemma [114], we have that for some $q = \Theta(\log(\binom{d}{s} \cdot z)/\varphi^2)$, there is a function $h : \mathbb{R}^d \to \mathbb{R}^q$ that preserves inner product, i.e., for each $a \in \mathrm{rows}(\Psi)$ there is

$$\langle h(a), h(\theta^*) \rangle = \langle a, \theta^* \rangle \pm 2\varphi, \tag{4.15}$$

for some error $\varphi > 0$.

3. **Estimation:** After inputting Algorithm 4 with the feature matrix Ψ, function h and the total time steps z, we can estimate the compressed parameter $\hat{\theta}_h \in \mathbb{R}^q$. Based on $\hat{\theta}_h \in \mathbb{R}^q$ and $\Psi_h \in \mathbb{R}^{\binom{d}{s}\cdot z \times q}$ whose rows are $h(a)$ for all $a \in \mathrm{rows}(\Psi)$, the s-sparsity estimator $\hat{\theta} \in \mathbb{R}^d$ can be computed via the convex optimization problem (4.16).

Using the above notations, the proposed algorithm is illustrated in Algorithm 5. The following theorem presents the sample complexity of our method.

Algorithm 5: $\mathrm{poly}(s)$-Query Algorithm for General Features

```
1
1: Input: feature matrix $\Phi \in \mathbb{R}^{k\times d}$.
2: Compute $\rho_{\mathcal{M}'} : \mathrm{rows}(\Phi_{\mathcal{M}'}) \to [0,1]$ satisfying the results of
   Theorem 4 for each sub-index-set $\mathcal{M}' \subseteq [d]$ with $|\mathcal{M}'| = s$.
3: Based on $\{\rho_{\mathcal{M}'}\}$, we collect representative action features as
   $\Psi \in \mathbb{R}^{\binom{d}{s}\cdot z\times d}$ where $z := 4s\log\log(s) + 16$.
4: Find a function $h : \mathbb{R}^d \to \mathbb{R}^q$ satisfying (4.15).
5: Get $\hat{\theta}_h \in \mathbb{R}^q$ and $\Psi_h \in \mathbb{R}^{\binom{d}{s}\cdot z\times q}$ via inputting ($\Psi$,$h$,$z$) to Algo-
   rithm 4.
6: Estimate the parameter $\hat{\theta} \in \mathbb{R}^d$ via the convex optimization prob-
   lem
       $$\min_{\theta\in\mathbb{R}^d} \left\| \Psi\theta - \Psi_h\hat{\theta}_h \right\|_\infty \text{ subject to } \|\theta\|_0 \le s. \qquad (4.16)$$
7: Output: $\hat{\theta} \in \mathbb{R}^d$
```

Theorem 7 *Suppose there is a function $h : \mathbb{R}^d \to \mathbb{R}^q$ satisfied (4.15) for $q = \Theta(\log(\binom{d}{s}\cdot z)/\varphi^2)$ and $\varphi = (s\log d)^{1/4}\sqrt{\varepsilon}$. Then after $n \ge O(\sqrt{s\log d}/\varepsilon)$ number of queries, the output of Algorithm 5 satisfies $|r_a - \langle a, \hat{\theta}\rangle| \le O((s\log d)^{1/4}\sqrt{s\varepsilon} + \varepsilon)$ for all $a \in \mathrm{rows}(\Phi)$.*

***Proof:**[Proof Sketch] We start with the approximation error of $\theta^* \in \mathbb{R}^d$ based on the feature matrix $\Psi \in \mathbb{R}^{k' \times d}$. According to (4.13) in Section 4.6, we can apply G-optimal design to compute the estimator $\hat{\theta}_h \in \mathbb{R}^q$ satisfying*

$$|\langle h(a), \hat{\theta}_h \rangle - \langle a, \theta^* \rangle| \le C' \left(\varepsilon \sqrt{q} + \varphi \right), \tag{4.17}$$

where $C' > 0$. Recall that $q = \Theta(\log(\binom{d}{s} \cdot z)/\varphi^2)$ with $z := 4s \log\log(s) + 16$, thus $C'(\varepsilon\sqrt{q} + \varphi)$ can be presented as a function with respect to φ ($\varphi > 0$), given by

$$g(\varphi) = C''(\varepsilon \sqrt{s \log(d)/\varphi^2} + \varphi),$$

where $C'' > 0$. The minimum of $g(\varphi)$ achieves when $\varphi^* = (s \log d)^{1/4} \sqrt{\varepsilon}$ such that

$$g(\varphi^*) = C \left((s \log d)^{1/4} \sqrt{\varepsilon} \right),$$

where $C > 0$. Hence, we have the approximation error, $\forall\, b \in \mathrm{rows}(\Psi)$

$$|\langle h(b), \hat{\theta}_h \rangle - \langle b, \theta^* \rangle| \le C \left((s \log d)^{1/4} \sqrt{\varepsilon} \right), \tag{4.18}$$

which is achieved by the number of queries

$$O\left(\log\left(\binom{d}{s} \cdot z \right) / (v^*)^2 \right) = O\left(\sqrt{s \log(d)} / \varepsilon \right).$$

For $a \in \mathrm{rows}(\Psi)$, the approximate error of estimator $\hat{\theta}$ in (4.16) can be bounded by

$$\begin{aligned}
\left| \langle a, \hat{\theta} \rangle - \langle a, \theta^* \rangle \right| &\le \left| \langle a, \hat{\theta} \rangle - \langle h(a), \hat{\theta}_h \rangle \right| + \left| \langle h(a), \hat{\theta}_h \rangle - \langle a, \theta^* \rangle \right| \\
&\overset{(a)}{\le} \max_{a \in \mathrm{rows}(\Psi)} \left| \langle a, \hat{\theta} \rangle - \langle h(a), \hat{\theta}_h \rangle \right| + C \left((s \log d)^{1/4} \sqrt{\varepsilon} \right) \\
&\overset{(b)}{\le} \max_{a \in \mathrm{rows}(\Psi)} \left| \langle a, \theta^* \rangle - \langle h(a), \hat{\theta}_h \rangle \right| + C \left((s \log d)^{1/4} \sqrt{\varepsilon} \right) \\
&= O\left((s \log d)^{1/4} \sqrt{\varepsilon} \right),
\end{aligned} \tag{4.19}$$

where step (a) and the last step come from (4.18) and step (b) derives due to the estimator $\hat{\theta}$ of the problem (4.16).

From the estimator $\hat{\theta}$, we can get an index set $\hat{\mathcal{M}} := \{i \mid \hat{\theta}_i \neq 0,\ i \in [d]\}$. Let $\mathcal{M}' = \hat{\mathcal{M}} \cup \mathcal{M}^*$, we then have[2]

$$\hat{\theta}_{\mathcal{M}'} = G_{\mathcal{M}'}^{-1} G_{\mathcal{M}'} \hat{\theta}_{\mathcal{M}'} = G_{\mathcal{M}'}^{-1} \sum_{a \sim \rho_{\mathcal{M}'}} \langle a_{\mathcal{M}'}, \hat{\theta}_{\mathcal{M}'} \rangle a_{\mathcal{M}'}, \tag{4.20}$$

which helps us to bound the approximate error of θ^* for $\forall\ b \in \mathrm{rows}(\Phi)$. Thus, there is

$$\begin{aligned}
\langle b, \hat{\theta} \rangle - \langle b, \theta^* \rangle &= \langle b_{\mathcal{M}'}, \hat{\theta}_{\mathcal{M}'} - \theta^*_{\mathcal{M}'} \rangle \\
&\overset{(a)}{=} \left\langle b_{\mathcal{M}'}, G_{\mathcal{M}'}^{-1} \sum_{a_{\mathcal{M}'} \sim \rho_{\mathcal{M}'}} \langle a_{\mathcal{M}'}, \hat{\theta}_{\mathcal{M}'} \rangle a_{\mathcal{M}'} - \theta^*_{\mathcal{M}'} \right\rangle \\
&= \left\langle b_{\mathcal{M}'}, G_{\mathcal{M}'}^{-1} \sum_{a_{\mathcal{M}'} \sim \rho_{\mathcal{M}'}} |\langle a_{\mathcal{M}'}, \hat{\theta}_{\mathcal{M}'} \rangle - \langle a_{\mathcal{M}'}, \theta^*_{\mathcal{M}'} \rangle| a_{\mathcal{M}'} \right\rangle \\
&= \sum_{a_{\mathcal{M}'} \sim \rho_{\mathcal{M}'}} |\langle a_{\mathcal{M}'}, \hat{\theta}_{\mathcal{M}'} \rangle - \langle a_{\mathcal{M}'}, \theta^*_{\mathcal{M}'} \rangle| \langle b_{\mathcal{M}'}, G_{\mathcal{M}'}^{-1} a_{\mathcal{M}'} \rangle \\
&\overset{(b)}{\leq} C \left((s \log d)^{1/4} \sqrt{\varepsilon} \right) \sum_{a_{\mathcal{M}'} \sim \rho_{\mathcal{M}'}} |\langle b_{\mathcal{M}'}, G_{\mathcal{M}'}^{-1} a_{\mathcal{M}'} \rangle| \\
&\overset{(c)}{\leq} C \left((s \log d)^{1/4} \sqrt{\varepsilon} \right) \sqrt{\sum_{a_{\mathcal{M}'} \sim \rho_{\mathcal{M}'}} \langle b_{\mathcal{M}'}, G_{\mathcal{M}'}^{-1} a_{\mathcal{M}'} \rangle^2} \\
&= C \left((s \log d)^{1/4} \sqrt{\varepsilon} \right) \sqrt{\|b_{\mathcal{M}'}\|^2_{G_{\mathcal{M}'}^{-1}}} \\
&\overset{(d)}{\leq} C \left((s \log d)^{1/4} \sqrt{\varepsilon} \right) \sqrt{g_{\mathcal{M}'}(\rho_{\mathcal{M}'})} \\
&= O \left((s \log d)^{1/4} \sqrt{s\varepsilon} \right),
\end{aligned} \tag{4.21}$$

where step (a) comes from the definition of $G_{\mathcal{M}'}$ (4.20), step (b) derives from Holder's inequality and the inequality (4.19), step (c) is due to Cauchy–Schwarz inequality and step (d) follow the definition of $g_{\mathcal{M}'}(\rho_{\mathcal{M}'})$ in Theorem 4. Hence, we ensure that $|r_b - \langle b_{\mathcal{M}'}, \hat{\theta}_{\mathcal{M}'} \rangle| \leq O((s \log d)^{1/4} \sqrt{s\varepsilon} + \varepsilon)$ for all $b \in \mathrm{rows}(\Phi)$. □

The results in Theorem 7 do not depend on the number of actions k, unlike Theorem 6. This is achieved by selecting representative actions and applying compression to get the submatrix Ψ_h,

[2] *Same as Theorem 4, we assume $G_{\mathcal{M}'}$ is invertible. If not, we can discard columns in Φ until the $\Phi_{\mathcal{M}'}$ is full column rank.*

followed by estimation based on the submatrix. In other words, this method works for general features, not just benign ones introduced in Section 4.6. The following corollary restates Theorem 6. It shows the relaxed requirements on the sparse linear bandit model to achieve $O(s\varepsilon)$-optimal actions within $O(s)$ queries, which present more general results compared to Corollary 1.

Corollary 2 *Based on the notations in Theorem 7, if $s = \Omega(\log(d)/\varepsilon^2)$, $O(s\varepsilon)$-optimal actions can be achieved with the number of queries $O(s)$.*

4.8 Conclusions

We aim to utilize the sparsity in linear bandits to remove the $\varepsilon\sqrt{d}$ barrier in the approximation error in existing results [66] about the misspecified setting. We provide a thorough investigation of how sparsity helps in learning misspecified linear bandits.

We establish novel algorithms that obtain $O(\varepsilon)$-optimal actions by querying $O(\varepsilon^{-s}d^s)$ actions, where s is the sparsity parameter. For fixed sparsity s, the algorithm finds an $O(\varepsilon)$-optimal action with $\mathrm{poly}(d/\varepsilon)$ queries, removing the dependence of $O(\varepsilon\sqrt{d})$. The ε^{-s} dependence in the sample bound can be further improved to $\tilde{O}(s)$ if we instead find an $O(\varepsilon\sqrt{s})$ suboptimal actions. We establish information-theoretical lower bounds to show that our upper bounds are nearly tight. In particular, we show that any algorithms that can obtain $O(\Delta)$-optimal actions need to query $\Omega(\exp(s\varepsilon/\Delta))$ samples from the bandit environment. We further break the $\exp(s)$ sample barrier by showing an algorithm that achieves $O(s\varepsilon)$ sub-optimal actions while only querying $\mathrm{poly}(s/\varepsilon)$ samples.

Starting from our results on the general bound in misspecified sparse linear bandits, it is interesting to explore results in different bandit learning settings, e.g., contextual bandit problems, RL problems, and distributed/federated learning settings.

CHAPTER 5

Investigation into Partial Observation in Delay Bandit Problem

5.1 Introduction

In reinforcement learning (RL) environments, the agent aims to make sequential decisions based on state and action spaces, to maximize its expected cumulative reward [116], which is known as the Markov Decision Process (MDP). In real-world applications of modern RL, such as robotic control [117], autonomous vehicles [118], and financial trading [119], the agent cannot make proper actions solely based on the observation state due to the delayed feedback in the environment. In this chapter, we consider the delay in receiving observations after taking actions, also known as observation delay. At the current state of the environment, the agent possesses the observational state, i.e., a previous environment state, and an action sequence. The inference of the current environment state from the agent's observation state and the action sequence inevitably meets the challenges of expansive state and action spaces, commonly known as the curse of large state and action spaces. Although there have been multiple methods based on feature mapping [120] targeting addressing the curse of large state and action spaces in standard MDP, it is not clear how to address this problem in the delayed MDP.

In this chapter, we develop provable RL algorithms for delayed MDPs (m time-step delay) with feature mapping. Our proposed method enjoys regret bound that is independent of the size of the state and action spaces, thereby addressing the curse of large state and action spaces. To achieve this, we introduce a parameterized transition model depending on an observation state and an action sequence. The connection between delayed MDP and standard MDP is established particularly in scenarios where two continuous action sequences have $m-1$ 'overlapping' actions.

However, these 'overlapping' actions bring challenges in regret analysis. This is due to the intricate statistical dependence between model parameter estimation based on action sequences and feature mapping disrupts the assumption of i.i.d. random variables crucial for establishing generic bounds in random matrix theory. To rigorously establish the regret bound based on the estimated value functions, we develop a novel mechanism to address the statistical issue of 'overlapping' actions. Specifically, we introduce a sequence of auxiliary parameter iterations that are close surrogates of the original and are independent of the feature mapping vectors. In such cases, it is much easier to control the estimation iterations' incoherence w.r.t. feature mapping vectors to the desired level, yielding a lower regret bound.

The regret of the proposed algorithm is bounded by $\tilde{O}((d+2-\gamma^m)\sqrt{T}/(1-\gamma)^2)$. In particular, this regret bound is independent of the number of states and actions and is close to a lower bound $\Omega(d\sqrt{T}/(1-\gamma)^{\frac{3}{2}})$ where d is the dimension of feature, T is the total time steps and γ is the discount factor.

Notation We introduce several notations used throughout this chapter. For a vector $\mathbf{x} \in \mathbb{R}^d$, we denote by $\|\mathbf{x}\|_2$ the Euclidean norm. For two sequences $\{f(n)\}$ and $\{g(n)\}$, $f(n) = O(g(n))$ denotes that there exists a constant $c > 0$ such that $|f(n)| \leq c|g(n)|$. $\tilde{O}(\cdot)$ is used to hide the logarithmic factors. Additionally, $f(n) = \Omega(g(n))$ means that there exists a constant $c > 0$ such that $|f(n)| \geq c|g(n)|$.

5.2 Related Work

MDP in delayed environments Previous work on delayed environments can be traced back to the control theory with linear time-invariant systems [121, 122, 123, 124, 125]. Generally, the system to control theory formulations can be represented by some known diffusion or stochastic differential equation. Similarly, a line of literature has considered the MDP model with structural assumption on the transition function or reward [84, 126, 65].

There are different types of delayed feedback in delayed environments, e.g., reward delay, observation delay, and execution delay. Note that observation delay and execution delay are actually equivalent and can thus be treated with the same models [127, 128]. Therein, the paper [129] studied multi-armed bandits for deterministic and stochastic latencies to deal with reward delay. The paper [130] proposed a Q-learning variant for solving MDPs with reward-delay that satisfies a Poisson distribution. The paper [127] investigated three types of delay in MDP: observation, execution, and reward. The paper [131] considered execution delay on multi-agent reinforcement learning problem. The aforementioned papers on MDPs utilize an augmented approach to evaluate the empirical model. Specifically, these empirical evaluations involve the degradation caused due to the delay. In this augmentation approach, all pending action is integrated with the original state, thereby mitigating the partial observability introduced by the delay. The main disadvantage of this augmented approach is the exponential growth of the state-space concerning the delay value [128, 131, 131].

To address the exponential-growth issue, the paper [128] proposed an efficient planning approach, i.e., model-based simulation (MBS) algorithm, to solve MDPs with delayed observation delay. Under the assumption of *deterministic* or *mild stochastic* probability transition function, the MBS algorithm [128] plans an optimal policy based on a most likely present state estimate. The sample complexity of MBS is $\tilde{O}(|\mathcal{S}|^2|\mathcal{A}|/((1-\gamma)^6\epsilon^3))$. MBS is an offline algorithm that requires the state space to be finite or discretized, which is inapplicable to large continuous domains. To address the limitations of MBS, the paper [132] has recently proposed a Delayed-Q model-based algorithm for learning and planning in MDPs with delayed execution without resorting to state-augmentation. It proves that Markovian non-stationary policies in the original state space are sufficient for achieving an optimal reward. Recently, the paper [133] proposed a delayed OPPO algorithm to learn adversarial MDPs with delayed feedback.

Another line of work studied a concurrent control problem where a single action selection occurs between two consecutive observations [134, 135]. Therefore, this control problem can be termed as an MDP with an execution delay of $m = 1$. [131] further extended it to a generalized setting where there are a multiple number of actions between two observations. [136] proposed a policy-based method to handle a relatively low execution delay, i.e., $m \leq 3$ in the braking control of autonomous vehicles.

Although recent papers like [132, 133] offer experimental insights, their work lacks theoretical results on regret bounds. Our work seeks to address this gap by providing theoretical analyses.

MDP with feature mapping Recent years have seen rapid growth of research on solving MDP using reinforcement learning with feature mapping, especially linear function approximation, e.g., [91, 84, 137, 126, 78, 65]. There is a line of literature focusing on finite-horizon MDP with feature mapping. For instance, [84] studied linear MDPs where both the transition probability and the rewards are linear functions concerning a feature mapping $\boldsymbol{\phi} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$. The authors proposed an efficient reinforcement learning algorithm to solve the linear MDPs with $\tilde{O}(\sqrt{d^3 H^3 T})$, where H is the length of an episode. [126] assumed the probability transition kernel is bilinear in two feature mappings in dimension d and d', and proposed an algorithm with $\tilde{O}(dH^2\sqrt{T})$ regret. Both [65] and [88] provide a general view of solving linear MDPs under the episodic reinforcement learning setting. Therein, [65] provided sharp thresholds for reinforcement learning, which implies the conditions for constituting good function approximation. A parallel line of literature investigates linear kernel MDPs of which probability transition function can be represented by a ternary feature mapping $\psi : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^d$ [138, 85]. Besides linear function approximation, there are some other function approximation settings of interest in finite-horizon MDP, such as general function approximation with Eluder dimension [139, 79, 138] and kernel approximation [140].

Apart from finite-horizon MDP, significant efforts have been recently made to study discounted MDP with feature mapping [141, 66, 142]. To be specific, [141] assumed that the probability transition function can be parameterized by a d-dimensional feature mapping. They proposed a phased parametric Q-Learning algorithm which achieves an ϵ-suboptimal policy with the optimal sample complexity, i.e., $\tilde{O}(d/((1-\gamma)^3\epsilon^2))$. Furthermore, [66] studied stochastic bandits and reinforcement learning with a generative model. Under this setting, the learner can approximate the action-value functions for all policies via d-dimensional linear features. To solve this problem,

[66] proposed a phased elimination algorithm with $\tilde{O}(d/((1-\gamma)^4\epsilon^2))$ sample complexity. Both [141] and [66] assumed the learner could access a generative model. Recently, based on the feature mapping, [142] proposed an upper-confidence linear kernel reinforcement learning (UCLK) algorithm without accessing the generative model. This algorithm solves RL for discounted MDPs with a $\tilde{O}(d\sqrt{T}/(1-\gamma)^2)$ regret.

Before [142], there is also a line of literature focusing on learning discounted MDP without accessing the generative model even though utilizing other technical arguments instead of feature mapping [143, 144, 145, 146]. [143] proposed a delay-Q-learning algorithm that achieves near optimal performance, i.e., $\tilde{O}(|\mathcal{S}||\mathcal{A}|/((1-\gamma)^8\epsilon^4))$ sample complexity of exploration. MoRmax algorithm was proposed by [144] requiring $\tilde{O}(|\mathcal{S}||\mathcal{A}|/((1-\gamma)^6\epsilon^2))$ sample complexity of exploration. [145] proposed upper confidence reinforcement learning algorithm (UCRL) algorithm which obtains $\tilde{O}(|\mathcal{S}|^2|\mathcal{A}|/((1-\gamma)^3\epsilon^2))$ sample complexity of exploration. [146] proposed Infinite Q-learning with UCB that achieves $\tilde{O}(|\mathcal{S}||\mathcal{A}|/((1-\gamma)^7\epsilon^2))$ sample complexity of exploration.

However, existing state-of-the-art approaches have not considered MDP environments with delayed feedback. Analyzing delayed MDPs poses theoretical challenges due to their intricate statistical dependencies between actions in the pending action sequence.

To achieve the regret bound independent of the size of the state and action spaces in delayed settings, we first introduce a parameterized transition model that serves as a bridge between delayed MDP and standard MDP. This connection is established particularly in scenarios where two consecutive action sequences have $m-1$ overlapping actions. To further address the statistical challenges brought by 'overlapping' action, we develop an effective mechanism to decouple the dependence between the model parameter estimate and the feature mapping. This mechanism enables us to derive desirable regret bounds.

5.3 Preliminaries

An infinite-horizon discounted MDP is denoted by a tuple $\langle \mathcal{S}, \mathcal{A}, \gamma, r, \mathbb{P} \rangle$, where $\mathcal{S}$ is a countable state space (could be finite or infinite), $\mathcal{A}$ is the action space, $\gamma : 0 \leq \gamma < 1$ is the discount factor, $r : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the reward function. For simplicity, we assume the reward function r is deterministic. $\mathbb{P}(s'|s, a)$ is the transition probability function which denotes the probability for state s to transfer to state s' given action a. A policy $\pi : \mathcal{S} \to \mathcal{A}$ maps states s to actions a. Let $\{s_t, a_t\}_{t=1}^{\infty}$ be states and actions deduced by $\mathbb{P}$ and π. We denote the action-value function $Q_t^{\pi}(s, a)$ and value function $V_t^{\pi}(s)$ as follows

$$\begin{aligned} Q_t^{\pi}(s, a) &= r(s, a) + \gamma \mathbb{E}_{s' \sim \mathbb{P}(\cdot|s,a)} V_t^{\pi}(s'), \\ V_t^{\pi}(s) &= \mathbb{E}\bigg[\sum_{i=0}^{\infty} \gamma^i r(s_{t+i}, \pi(s_{t+i})) \bigg| s_t = s \bigg]. \end{aligned}$$

We define the optimal value function V^* and the optimal action-value function Q^* as $V^*(s) = \sup_{\pi} V^{\pi}(s)$ and $Q^*(s, a) = \sup_{\pi} Q^{\pi}(s, a)$. For simplicity, for any function $V : \mathcal{S} \to \mathbb{R}$, we denote $[\mathbb{P}V](s, a) = \mathbb{E}_{s' \sim \mathbb{P}(\cdot|s,a)} V(s')$. Therefore we have the following Bellman equation, as well as the Bellman optimality equation:

$$Q_t^*(s_t, a_t) = r(s_t, a_t) + \gamma [\mathbb{P}V^*](s_t, a_t).$$

Based on the above notations, we introduce a constant-delay MDP as a tuple $\langle \mathcal{S}, \mathcal{A}, \gamma, r, \mathbb{P}, m \rangle$ where m is a non-negative integer indicating the number of timesteps between the current state of the environment and the last state observed by the agent. When $m = 0$, the CDMDP becomes a regular MDP; otherwise, we assume that the agent observes its initial state in response to each of its first m actions. This assumption is based on the intuition that one would not expect an agent in a delayed environment to act before making at least a starting observation.

To avoid the sample complexity depending on the size of the state and action spaces, we develop the structured CDMDP. In this case, the unknown probability transition function is modeled with a linear parametrization $\mathbb{P} = \sum_i \theta_i \mathbb{P}_i$, where $\mathbb{P}_1, \mathbb{P}_2, \cdots, \mathbb{P}_d$ represent known basis models, and $\boldsymbol{\theta} = (\theta_1, ..., \theta_d)$ represents the unknown parameters.

5.4 Problem Formulation

In the subsequent discussion, we employ the term 'current observation state' to denote the last state known by the agent. The environment state is considered to be the state that is m-time-step following the current observation state. At t-th time step, define s_t as the current observation state and the pending action sequence $\vec{\boldsymbol{a}}_t = (a_{t-m}, \cdots, a_{t-1})$. The action a_t is selected according to $a_t \leftarrow \pi(s_t)$. At t-th time step, action a_{t-m} will be executed on state s_t independently of the present action selection, thereby yielding a new observation state s_{t+1}. It arrives at the new pending action sequence $\vec{\boldsymbol{a}}_{t+1} = (a_{t-m+1}, \cdots, a_t)$. Given the current observation state s_t and the action sequence $\vec{\boldsymbol{a}}_t$, the action-value function with action sequence $Q^{\pi}_m(s_t, a_t, \vec{\boldsymbol{a}}_t)$ and value function with action sequence $V^{\pi}_m(s_t, \vec{\boldsymbol{a}}_t)$ are denoted as follows,

$$\begin{aligned} Q^{\pi}_m(s_t, a_t, \vec{\boldsymbol{a}}_t) &= r(s_t, a_{t-m}) + \gamma \mathbb{E}_{s'} V^{\pi}_m(s', \vec{\boldsymbol{a}}_{t+1}), \\ V^{\pi}_m(s_t, \vec{\boldsymbol{a}}_t) &= \mathbb{E}\Bigg[\sum_{i=0}^{m-1} \gamma^i r(s_i, a_{t+i-m}) + \\ &\qquad \sum_{i=m}^{\infty} \gamma^i r(s_i, \pi(s_{i-m}))\Bigg| s_0 = s_t\Bigg]. \end{aligned} \tag{5.1}$$

Thus, the Bellman optimality equation is given as:

$$Q^{*}_m(s_t, a_t, \vec{\boldsymbol{a}}_t) = r(s_t, a_{t-m}) + \gamma[\mathbb{P}V^{*}_m](s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}),$$

where $[\mathbb{P}V_m](s, a, \vec{\boldsymbol{a}}) = \mathbb{E}_{s' \sim \mathbb{P}(\cdot|s,a)} V_m(s', \vec{\boldsymbol{a}})$ and the optimal value function is given by $V^{*}_m(s, \vec{\boldsymbol{a}}) = \max_{\pi} V^{\pi}_m(s, \vec{\boldsymbol{a}})$.

We aim to estimate the optimal value function with an efficient algorithm in which sample complexity is independent of the size of the state and action spaces. To achieve this goal, the use of a structured probability transition model becomes essential. This approach is widely adopted in real-world applications, where different state-action features may demonstrate inherent correlations with each other, influencing the determination of the probability transition function. In this work, we consider the *parameterized-transition CDMDPs*, where the transition probability function can be represented as a linear function of a given feature mapping $\boldsymbol{\phi} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^d$.

Definition 2 $\langle \mathcal{S}, \mathcal{A}, \gamma, r, \mathbb{P}, m \rangle$ *is called a parameterized-transition CDMDP if there exist a* known *feature mapping* $\boldsymbol{\phi}(s'|s,a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^d$ *and an* unknown *vector* $\boldsymbol{\theta} \in \mathbb{R}^d$ *with* $\|\boldsymbol{\theta}\|_2 \leq \sqrt{d}$*, such that*

- *For any state-action-state triplet* $(s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$*, we have* $\mathbb{P}(s'|s,a) = \langle \boldsymbol{\phi}(s'|s,a), \boldsymbol{\theta} \rangle$*;*
- *For any augmented state,* $\mathbf{I}_m = (s, a_{-m}, \cdots, a_{-1}) \in S \times A^m$*, the transition probabilities used in the augmented approach can be defined:*

$$\mathbb{P}'\left(\mathbf{I}'_m | \mathbf{I}_m, a\right) = \begin{cases} \langle \boldsymbol{\phi}(s'|s, a_{-m}), \boldsymbol{\theta} \rangle, \\ \textit{if } \mathbf{I}'_m = (s', a_{-m+1}, \cdots, a_{-1}, a)\,; \\ 0, \textit{ otherwise} \end{cases}$$

- *For any bounded function* $V_m : \mathcal{S} \times \mathcal{A}^m \to [0,1]$ *and the tuple* $(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}) \in \mathcal{S} \times \mathcal{A} \times \mathcal{A}^m$*, we have* $\boldsymbol{\phi}_{V_m}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}) = \sum_{s'} \boldsymbol{\phi}(s'|s_i, a_{i-m}) V_m(s', \vec{\boldsymbol{a}}_{i+1}) \in \mathbb{R}^d$*, where* $\vec{\boldsymbol{a}}_{i+1} = (a_{i-m+1}, \cdots, a_{i-1}, a_i)$*.*

Note that if the next augmented state $\mathbf{I}'_m$ have overlap action sequence, i.e., $(a_{-m+1}, \cdots, a_{-1})$ as the current augmented state $\mathbf{I}_m$, we can simply the augmented transition probability model to $\mathbb{P}(s'|s, a_{-m})$. This means that the next state s' is determined by the current state s and previous action a_{-m}, and we put the new action a into the action sequence, i.e., $(a_{-m+1}, \cdots, a_{-1}, a)$. This intuition establishes a connection between delayed MDP and standard MDP, playing a crucial role in our method. However, this 'overlap' introduces significant challenges in theoretical analysis, particularly in distinguishing the i.i.d. random variable in the proof. More details on technical novelties will be explained in the next section.

In online learning, the agent observes the starting state s_1 along with pending action sequence $(a_{1-m}, \cdots, a_0)$ at the beginning. The goal is to design a policy π such that the expected discounted return at step t is close to the optimal expected return $V^*_m(s_t)$. We formalize this goal as minimizing the regret, which can be defined as follows.

Definition 3 *For any policy π, we define its regret on CDMDP $\langle \mathcal{S}, \mathcal{A}, \gamma, r, \mathbb{P}, m \rangle$ in the first T rounds as*

$$\textit{Regret}(T) = \sum_{t=1}^{T} V_m^*(s_t, \vec{\boldsymbol{a}}_t) - V_m^\pi(s_t, \vec{\boldsymbol{a}}_t), \tag{5.2}$$

where the action sequence is given by $\vec{\boldsymbol{a}}_t = (a_{t-m}, \cdots, a_{t-1})$.

The regret (Definition 3) in the delayed environment reflects the difference between the cumulative reward obtained by the learning algorithm and the cumulative reward that could have been achieved by an optimal policy. In contrast to regret definitions without considering delays, this formulation acknowledges that the algorithm must deal with the uncertainty and delayed consequences of its actions.

5.5 Value-Targeted Model Regression for Delayed MDP

Our algorithm, i.e., Algorithm 6, solves the discounted Markov Decision Processes with delayed feedback. The proposed algorithm is a model-based algorithm that exploits the empirical value-target model concerned with delayed feedback. The key insight is to embed the effect of delayed feedback into the value-target model, followed by estimating the model based on experiences.

5.5.1 Confidence Set Construction for Value-Targeted Model with Delayed Feedback

Confidence set construction is essential in value-targeted model estimation because it quantifies uncertainty, ensures robustness, aids optimal decision-making, controls errors, and allows for the incorporation of prior information. In this section, under the delayed feedback, we first present the close form of the model parameter $\hat{\boldsymbol{\theta}}_k$ estimation, followed by the confidence set construction.

At t-th time step, it occurs a transition involving (s_t, a_{t-m}, s_{t+1}) where $s_{t+1} \sim \mathbb{P}(\cdot|s_t, a_{t-m})$, thus we receive information on the probability transition function $\mathbb{P}$. Instead of regression onto a fixed target, e.g., probabilities, we focus on estimating the model via regression and taking the estimated value functions concerned with delayed feedback as the target.

Algorithm 6: Value-Targeted Model Regression for Delayed MDP

```
Require: Regularization parameter $\lambda$, confidence radius $\beta_k$, $\forall\, k$ (5.6), number of
    inner iteration rounds $\alpha$ (5.7), time horizon $T$
 1: Observe $s_1$ and the pending action sequence $\vec{\boldsymbol{a}}_1 = (a_{1-m}, \cdots, a_0)$
 2: Set $t \leftarrow 1$, $\boldsymbol{\Sigma}_1 \leftarrow \lambda\mathbf{I}$, $\mathbf{b}_1 = \mathbf{0}$, $Q_m^k(\cdot, \cdot, \vec{\boldsymbol{a}}_1) = 1/(1-\gamma), \forall\, k$
 3: for $k = 0, \ldots$ do
 4:     Set $t_k \leftarrow t$, $\hat{\boldsymbol{\theta}}_k \leftarrow \boldsymbol{\Sigma}_{t_k}^{-1}\mathbf{b}_{t_k}$
 5:     Set the confidence set as $\mathcal{D}_k = \{\boldsymbol{\theta} : \|\boldsymbol{\Sigma}_{t_k}^{1/2}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_k)\|_2^2 \leq \beta_k\}$
 6:     if det$(\boldsymbol{\Sigma}_t) \leq 2$det$(\boldsymbol{\Sigma}_{t_k})$ and $t - t_k \leq \alpha$ then
 7:         Set $Q_m^k(\cdot, \cdot, \vec{\boldsymbol{a}}_t)$ based on the action sequence $\vec{\boldsymbol{a}}_t = (a_{t-m}, \cdots, a_{t-1})$ as
            follows:
                $V_m(\cdot) \leftarrow \max_{a \in \mathcal{A}} Q_m^k(\cdot, a, \vec{\boldsymbol{a}}_t),$
                $Q_m^k(s_t, a, \vec{\boldsymbol{a}}_t) \leftarrow r(s_t, a_{t-m}) +$
                $\max_{\boldsymbol{\theta} \in \mathcal{D}_k} \sum_{s'} \langle \boldsymbol{\phi}(s'|s_t, a_{t-m}), \boldsymbol{\theta} \rangle V_m^k(s', \vec{\boldsymbol{a}}_{t+1})$        (5.3)
 8:         Set $a_t = \mathrm{argmax}_{a \in \mathcal{A}}\, Q_m^k(\cdot, a, \vec{\boldsymbol{a}}_t)$ and update a new action sequence as
            $\vec{\boldsymbol{a}}_{t+1} = (a_{t-m+1}, \cdots, a_{t-1}, a_t)$, we have
            $V_m^k(\cdot, \vec{\boldsymbol{a}}_{t+1}) \leftarrow \max_{a \in \mathcal{A}} Q_m^k(\cdot, a, \vec{\boldsymbol{a}}_t)$,
 9:         Receive $s_{t+1} \sim \mathbb{P}(\cdot|s_t, a_{t-m})$ where $\mathbb{P}(\cdot|s, a) = \langle \boldsymbol{\phi}(\cdot|s, a), \hat{\boldsymbol{\theta}}_k \rangle$
10:         Set $\boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) = \sum_{s'} \boldsymbol{\phi}(s'|s_t, a_{t-m}) V_m^k(s', \vec{\boldsymbol{a}}_{t+1})$
11:         Set $\boldsymbol{\Sigma}_{t+1} \leftarrow \boldsymbol{\Sigma}_t + \boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) \boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1})^\top$
12:         Set $\mathbf{b}_{t+1} \leftarrow \mathbf{b}_t + V_m^k(s_{t+1}, \vec{\boldsymbol{a}}_{t+1}) \boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1})$
13:         $t \leftarrow t + 1$
14:     end if
15: end for
```

Denote the action sequence as $\vec{\boldsymbol{a}}_i = (a_{i-m}, \cdots, a_{i-1})$. Based on the above idea, the model parameter $\hat{\boldsymbol{\theta}}_k$ can be estimated by

$$\begin{aligned} \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \sum_{j=0}^{k-1} \sum_{i=t_j}^{t_{j+1}-1} \big[\big\langle \boldsymbol{\theta}, \boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1})\big\rangle - \\ V_m^k(s_{i+1}, \vec{\boldsymbol{a}}_{i+1})\big]^2 + \lambda\|\boldsymbol{\theta}\|_2^2. \end{aligned} \tag{5.4}$$

The above regression problem (5.4) has a closed-form solution. To calculate the solution of problem (5.4), we first present $\boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1})$ based on Definition 2 (see line 10 in Algorithm 6) and recursively calculate estimation of the ground-truth $\boldsymbol{\theta}^\natural$ via line 11 and line 12 in Algorithm 6). Thus, the estimated $\hat{\boldsymbol{\theta}}_k$ can be easily obtained by $\hat{\boldsymbol{\theta}}_k = \boldsymbol{\Sigma}_{t_k}^{-1}\mathbf{b}_{t_k}$ (refer to line 4). Note that during the recursive iterations of regression, as the accuracy of value estimations increases, the regret target is still dynamic. This is opposed to general supervised learning for estimating models.

To establish a confidence set, we consider a set, of which the center is $\hat{\boldsymbol{\theta}}_k$. Given a parameter $\boldsymbol{\theta}$, the distance between $\hat{\boldsymbol{\theta}}_k$ and $\boldsymbol{\theta}$ is denoted as

$$\begin{aligned} &\mathcal{L}_k(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}_k) \\ =&\sum_{j=0}^{k-1} \sum_{i=t_j}^{t_{j+1}-1} \Big(\big\langle\langle \boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_k, \boldsymbol{\phi}(\cdot|s_i, a_{i-m})\rangle, V_m^k(\cdot, \vec{\boldsymbol{a}}_{t+1})\big\rangle\Big)^2 \\ &+ \lambda\|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_k\|_2^2 \\ =&(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_k)^\top \boldsymbol{\Sigma}_{t_k}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_k). \end{aligned} \tag{5.5}$$

Based on (5.5) and the confidence radius parameter β, Algorithm 6 constructs the confidence set of $\boldsymbol{\theta}^\natural$ as $\mathcal{D}_k := \{\boldsymbol{\theta} : \|\boldsymbol{\Sigma}_{t_k}^{1/2}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_k)\|_2^2 \le \beta\}$.

5.5.2 Value Iteration

During k-th iteration in Algorithm 6, the confidence set $\mathcal{D}_k$ induces various plausible MDPs with delayed feedback. Then Algorithm 6 searches for the action-value function for the near-optimal MDP among these MDPs. At each iteration of Algorithm 6, given the observation state s_t and the action sequence $\vec{\boldsymbol{a}}_t = (a_{t-m}, \cdots, a_{t-1})$, we perform one-step optimal value iteration to obtain the new action-value function $Q_m^k(s_t, \cdot, \vec{\boldsymbol{a}}_t)$. This is achieved by selecting the best possible MDP among all plausible MDPs induced by $\mathcal{D}_k$ to maximize the Bellman optimality equation over the value function V_m^k. This is represented as line 7 in Algorithm 6.

After calculating Q_m^k, Algorithm 6 arrives at selecting a policy, taking action, collecting observations, and updating parameters. To be specific, at t-time step, the algorithm selects the optimal action a_t induced by Q_m^k and update the action sequence as $\vec{\boldsymbol{a}}_{t+1} = (a_{t-m+1}, \cdots, a_{t-1}, a_t)$. The action a_{t-m} is executed on the environment followed by observing the new state s_{t+1}. Then Algorithm 6 computes vector $\boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1})$ according to Definition 2 and the value function at s_{t+1}, i.e., $V_m^k(s_{t+1}, \vec{\boldsymbol{a}}_{t+1})$. The parameter $\boldsymbol{\Sigma}_t$ and $\mathbf{b}_t$ are updated based on $\boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1})$. Corresponding to the stopping rule in the [147], the inner loop in k-th iteration of Algorithm 6 repeats until det$(\boldsymbol{\Sigma}_t) >$ 2det$(\boldsymbol{\Sigma}_{t_k})$ or $t - t_k \leq \alpha$ where α is given by (5.7).

5.6 Theoretical Analysis

In this section, we develop the theoretical analysis of Algorithm 6, which efficiently solves reinforcement learning with value-targeted regression for discounted parameterized CDMDP.

The following theorem provides an upper bound of the regret for Algorithm 6.

Theorem 8 *Consider a parameterized-transition CDMDP $\langle \mathcal{S}, \mathcal{A}, \gamma, r, \mathbb{P}, m \rangle$ with the underlying vector $\boldsymbol{\theta}^\natural$. Set β_k and α in Algorithm 6 as follows:*

$$\begin{aligned}\beta_k = &\frac{8}{(1-\gamma)^2} \log\left(d\delta^{-1}\right) \\ &+ 4\sqrt{\frac{2}{(1-\gamma)^2} t_k^2 \log(2t_k(t_k+1)\delta^{-1})} \\ &+ m^2 t_k + 4dt_k + \lambda d. \end{aligned} \tag{5.6}$$

and

$$\alpha = \left\lceil \frac{\log\left[(mT)/(1-\gamma)\right])}{1-\gamma} \right\rceil + m, \tag{5.7}$$

then with probability exceeding $1-\delta$*, we have*

$$\begin{aligned} \textit{Regret}(T) \leq 1 &+ \frac{6}{1-\gamma}\sqrt{d\beta_T T \log\frac{\lambda + T/(1-\gamma)^2}{\lambda}} \\ &+ \frac{(2-\gamma^m)\sqrt{T\log\delta^{-1}}}{(1-\gamma)^2} + \frac{\gamma(1+\gamma^m)}{(1-\gamma)^2} \\ &\cdot \min\left\{2d\log\frac{\lambda+Td}{\lambda(1-\gamma)^2}, \alpha\right\}. \end{aligned} \tag{5.8}$$

Theorem 8 suggests that the regret of Algorithm 6 is in the order of $\tilde{O}((d+2-\gamma^m)\sqrt{T}/(1-\gamma)^2)$. The regret bound enjoys its independence from the size of the state and action space. Notably, Algorithm 6 has been wisely designed for handling delayed feedback. This strategic design empowers the agent to continuously improve decision-making, ensuring that the regret bound remains reasonably bound even in the face of delayed information. Under this approach, the regret bound of the algorithm not only avoids the dependence on the size of the state and action spaces but also mitigates the adverse effects of delayed feedback

Additionally, we analyze a lower bound to learn a parameterized CDMDP in the following theorem.

Theorem 9 *Consider a parameterized-transition CDMDP* $\langle \mathcal{S}, \mathcal{A}, \gamma, r, \mathbb{P}, 1\rangle$ *with the underlying vector* $\boldsymbol{\theta}^{\natural}$*. Choosing* $\psi = d\sqrt{1-\gamma}/(66\sqrt{2T})$ *and* $\delta = 1-\gamma$ *such that* $3\psi/2 \leq \delta \leq 1/5$*, then the regret with respect to any policy* π *satisfies that*

$$\mathbb{E}[\textit{Regret}(T)] \geq \frac{\gamma d\sqrt{T}}{2640\sqrt{\log 2}(1-\gamma)^{\frac{3}{2}}} - \frac{\gamma}{(1-\gamma)^2}. \tag{5.9}$$

Importantly, it is noteworthy that our upper bound regret, as dictated by the regret bound $\tilde{O}((d+2-\gamma^m)\sqrt{T}/(1-\gamma)^2)$, closely aligns with the lower bound $\Omega(d\sqrt{T}/(1-\gamma)^{\frac{3}{2}})$. This proximity underscores the efficacy of our proposed method, affirming its ability to not only navigate the intricacies of large state and action spaces but also substantially narrow the gap between the upper bound and the lower bound.

5.7 Proof of the Main Theorem

In this section, we provide the proof sketch of the main theorem. Let $N_T - 1$ be the number of epochs when Algorithm 6 executes $t = T$ rounds, and $t_{N_T} = T + 1$. The outline of proof is summarized as follows. To begin with, we construct the confidence set $\mathcal{D}_k$ for each k iteration. In Lemma 2, we show that the underlying vector $\boldsymbol{\theta}^{\natural}$ satisfies $\boldsymbol{\theta}^{\natural} \in \mathcal{D}_k$ for all $0 \leq k \leq N_T - 1$. Based on this condition, Lemma 3 further shows that the estimated Q-value functions corresponding to CDMDP, i.e., Q_m^k, are optimistic estimates of optimal Q-value functions corresponding to CDMDP, i.e., Q_m^*. Utilizing this fact, we can bound Regret(T) (5.2) via the sum of $V_m^k(s_t, \vec{\boldsymbol{a}}_t) - V_m^{\pi}(s_t, \vec{\boldsymbol{a}}_t)$, which can be decomposed into different terms and bounded separately. The proof of Theorem 8 is completed. Finally, we can transform the constructed regret bound to the sample complexity of exploration, which is illustrated in Theorem 10.

5.7.1 Confidence Set Construction

We first introduce several notations and definitions. $\hat{\boldsymbol{\theta}}_k$ in Algorithm 6 can be represented as

$$\hat{\boldsymbol{\theta}}_k = \bigg(\lambda \mathbf{I} + \sum_{j=0}^{k-1} \sum_{i=t_j}^{t_{j+1}-1} \boldsymbol{\Phi}\boldsymbol{\Phi}^\top\bigg)^{-1} \cdot \\ \bigg(\sum_{j=0}^{k-1} \sum_{i=t_j}^{t_{j+1}-1} V_m^k(s_{i+1}, \vec{\boldsymbol{a}}_{i+1})\boldsymbol{\Phi}\bigg),$$

where $\boldsymbol{\Phi} = \boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1})$. Additionally, for any $0 \le j \le k-1$ and $t_j \le i \le t_{j+1}-1$ and the pending action sequence $\vec{\boldsymbol{a}}_{i+1} = (a_{i+1-m}, \cdots, a_i)$,

$$\begin{aligned}
&[\mathbb{P}V_m^k](s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}) \\
=&\sum_{s'} \mathbb{P}(s'|s_i, a_{i-m}) V_m^k(s', \vec{\boldsymbol{a}}_{i+1}) \\
=&\sum_{s'} \langle \boldsymbol{\phi}(s'|s_i, a_{i-m}), \boldsymbol{\theta}^{\natural} \rangle V_m^k(s', \vec{\boldsymbol{a}}_{i+1}) \\
=&\Big\langle \sum_{s'} \boldsymbol{\phi}(s'|s_i, a_{i-m}) V_m^k(s', \vec{\boldsymbol{a}}_{i+1}), \boldsymbol{\theta}^{\natural} \Big\rangle \\
=&\langle \boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}), \boldsymbol{\theta}^{\natural} \rangle. \qquad (5.10)
\end{aligned}$$

For $\beta_k > 0$, we construct the confidence set as

$$\mathcal{D}_k(\beta_k) = \Bigg\{ \boldsymbol{\theta} : \sum_{j=0}^{k-1} \sum_{i=t_j}^{t_{j+1}-1} \Big(\langle \boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}), \boldsymbol{\theta} \rangle - \langle \boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}), \hat{\boldsymbol{\theta}}_k \rangle \Big)^2 + \lambda \|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_k\|_2^2 \le \beta_k \Bigg\}. \qquad (5.11)$$

Lemma 2 *Set β_k in Algorithm 6 as*

$$\beta_k = \frac{8}{(1-\gamma)^2} \log\left(d\delta^{-1}\right) + 4\sqrt{\frac{2}{(1-\gamma)^2} t_k^2 \log(2t_k(t_k+1)\delta^{-1}) + m^2 t_k} + 4dt_k + \lambda d, \qquad (5.12)$$

then with probability exceeding $1-\delta$, for all $0 \le k \le N_T - 1$ where $0 \le t_k \le T$, there is $\boldsymbol{\theta}^{\natural} \in \mathcal{D}_k$ with non-empty $\mathcal{D}_k$.

Lemma 2 suggests that in every epoch of Algorithm 6, $\boldsymbol{\theta}^{\natural}$ is contained in the confidence sets $\{\mathcal{D}_k\}_{k=0}^{N_T-1}$ with a high probability.

5.7.2 Optimism

Lemma 3 *Assume that the conditions in Lemma 2 hold. Then for all $0 \le k \le N_T - 1$, we have $1/(1-\gamma) \ge Q_m^k(s, a, \vec{\boldsymbol{a}}_t) \ge Q_m^*(s, a, \vec{\boldsymbol{a}}_t)$ for any $(s, a) \in \mathcal{S} \times \mathcal{A}$ with the action pending sequence*

$\vec{\boldsymbol{a}}_t = (a_{t-m}, \cdots, a_{t-1})$.

Lemma 3 suggests that in every epoch of Algorithm 6, $Q_m^k(s, a)$ found by (5.3) is an upper bound for the optimal action-value function $Q_m^*(s, a, \vec{\boldsymbol{a}}_t)$. Recall that the goal is to find the action-value function Q_k corresponding to the optimal MDP in $\mathcal{M}_k$, which satisfies the following optimality condition

$$\begin{aligned} Q_m^k(s, a, \vec{\boldsymbol{a}}_t) = r(s_t, a_{t-m}) + \\ \max_{\boldsymbol{\theta} \in \mathcal{D}_k} \sum_{s'} \langle \boldsymbol{\phi}(s'|s_t, a_{t-m}), \boldsymbol{\theta} \rangle V_m^k(s', \vec{\boldsymbol{a}}_{t+1}), \end{aligned}$$

where the action pending sequence is $\vec{\boldsymbol{a}}_{t+1} = (a_{t-m+1}, \cdots, a_{t-1}, a)$.

However, it is impossible to find the exact optimal value function since there is constant delay and only a finite number of iterations are performed. The following lemma characterizes the error to the constant delay m and the number of iterations.

5.7.3 Regret Decomposition

The regret can be decomposed as follows:

$$\begin{aligned} \text{Regret}(T) &= \sum_{k=0}^{N_T-1} \sum_{t=t_k}^{t_{k+1}-1} \left[V_m^*(s_t, \vec{\boldsymbol{a}}_t) - V_m^\pi(s_t, \vec{\boldsymbol{a}}_t) \right] \\ &\leq \sum_{k=0}^{N_T-1} \sum_{t=t_k}^{t_{k+1}-1} \left[V_m^k(s_t, \vec{\boldsymbol{a}}_t) - V_m^\pi(s_t, \vec{\boldsymbol{a}}_t) \right], \end{aligned} \tag{5.13}$$

where the inequality holds due to Lemma 3 and the update rule in line 8 of Algorithm 6.

$$\sum_{t=t_k}^{t_{k+1}-1} \left[V_m^k(s_t, \vec{\boldsymbol{a}}_t) - V_m^\pi(s_t, \vec{\boldsymbol{a}}_t) \right]$$

can be further bounded as follows by Bellman equation and Lemma 11.

$$\begin{aligned}
&\sum_{t=t_k}^{t_{k+1}-1} \left[V_m^k(s_t, \vec{\boldsymbol{a}}_t) - V_m^\pi(s_t, \vec{\boldsymbol{a}}_t)\right] \\
\leq &\left[\sum_{t=t_k}^{t_{k+1}-1} \frac{1}{1-\gamma}\left\langle \boldsymbol{\theta}_t - \boldsymbol{\theta}^\natural, \boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1})\right\rangle + \xi_t\right] \\
&+ \frac{m}{1-\gamma}(t_{k+1}-t_k)\gamma^{t_{k+1}-t_k-m+1} + 1 + \frac{1}{(1-\gamma)^2},
\end{aligned} \tag{5.14}$$

where $\xi_t = \left[\mathbb{P}(V_m^k - V_m^\pi)\right](s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) - \left(V_m^k(s_{t+1}, \vec{\boldsymbol{a}}_{t+1}) - V_m^\pi(s_{t+1}, \vec{\boldsymbol{a}}_{t+1})\right)/(1-\gamma)$. Computing the summation of (5.14) from $k = 0$ to $N_T - 1$, we derive the upper bound as follows

$$\begin{aligned}
&\sum_{t=t_k}^{t_{k+1}-1} \left[V_m^k(s_t, \vec{\boldsymbol{a}}_t) - V_m^\pi(s_t, \vec{\boldsymbol{a}}_t)\right] \\
\leq &\sum_{k=0}^{N_T-1}\sum_{t=t_k}^{t_{k+1}-1} \frac{1}{1-\gamma}\left\langle \boldsymbol{\theta}_t - \boldsymbol{\theta}^\natural, \boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1})\right\rangle + \\
&\sum_{k=0}^{N_T-1}\sum_{t=t_k}^{t_{k+1}-1} \xi_t + \frac{[2 + (1-\gamma)m\gamma^{\alpha-m}]N_T}{(1-\gamma)^2},
\end{aligned}$$

where α (5.7) is defined in Theorem 8. The upper bound of the first term on the right-hand side can be derived from Azuma-Hoeffding inequality and the definition of the value function (5.1), i.e., $\tilde{O}((2-\gamma^m)\sqrt{T}/(1-\gamma)^2)$, the second term can be bounded by $\tilde{O}(d\sqrt{T}/(1-\gamma)^2)$ by Lemma 2, the last term can be bounded by $\min\{\tilde{O}(d/(1-\gamma)^2), O(\alpha/(1-\gamma)^2)\}$ by Lemma 16 and the selection of α (5.7). Combining the upper bounds above, the regret of Algorithm 6 is bounded in the order of $\tilde{O}((d+2-\gamma^m)\sqrt{T}/(1-\gamma)^2)$. In analyzing regret bounds using random matrix theory such as Azuma-Hoeffding inequality, undelayed scenarios typically rely on i.i.d. random variables. However, delays introduce complex statistical dependence among pending actions, causing overlaps in sequences. To address the problem caused by the statistical dependence, we construct a variant of the target random variable, asymptotically approaching to the original variable by excluding a single datum. This "decoupling" effectively mitigates intricate dependencies induced by the pending action sequence. Our method excels in overcoming delay-related challenges, yielding a near-optimal regret bound and eliminating dependence on the state and action space size.

5.7.4 Transform between Sample Complexity of Exploration and Regret

A popular quantity used for measuring discounted MDPs is *sample complexity of exploration* $N(\epsilon, \delta)$ [145, 128, 144]. In our case, the sample complexity of exploration is defined as the number of time steps t where $V_m^*(s_t, \vec{\boldsymbol{a}}_t) - V_m^\pi(s_t, \vec{\boldsymbol{a}}_t)$ is greater than ϵ with probability at least $1 - \delta$. We transform the regret derived in Theorem 8 to the sample complexity of exploration for constant-delay MDP, presented in the following Theorem.

Theorem 10 *Consider a parameterized-transition CDMDP $\langle \mathcal{S}, \mathcal{A}, \gamma, r, \mathbb{P}, m \rangle$ with the underlying vector $\boldsymbol{\theta}^\natural$. For some $\epsilon, \delta > 0$ and $d < \epsilon^{-1}$, the non-stationary policy determined by Algorithm 6 in environments with delayed feedback is ϵ-optimal except in*

$$\tilde{O}\left(\frac{d + 2 - \gamma^m}{\epsilon^2 (1-\gamma)^3}\right). \tag{5.15}$$

time steps during the whole run of the algorithm, with probability exceeding $1 - \delta$.

***Proof**: Assume that an algorithm has $\tilde{O}(C_d \epsilon^{-a})$ sample complexity of exploration, where C_d is a constant that depends on the parameters of parameterized-CDMDP model, i.e., γ, d, m. Then with probability at least $1 - \delta$, it requires at least $\tilde{O}(C_d \epsilon^{-a})$ number of time steps t to arrive at $V_m^*(s_t, \vec{\boldsymbol{a}}_t) - V_m^\pi(s_t, \vec{\boldsymbol{a}}_t) \le \epsilon$. Denote the set $\mathcal{B}_k$ as the collection of time step t_k such that $\mathcal{B}_k = \{t_k | V_m^*(s_{t_k}, \vec{\boldsymbol{a}}_{t_k}) - V_m^\pi(s_{t_k}, \vec{\boldsymbol{a}}_{t_k}) \ge \epsilon\}$. Thus, for T time steps, with probability exceeding $1 - \delta$, we have*

$$\begin{aligned}
\textit{Regret}(T) &= \sum_{k=0}^{N_T-1} \sum_{t=t_k}^{t_{k+1}-1} \left[V_m^*(s_t, \vec{\boldsymbol{a}}_t) - V_m^\pi(s_t, \vec{\boldsymbol{a}}_t)\right] \\
&= \sum_{k=0}^{N_T-1} \Bigg[\sum_{t \in \mathcal{B}_k} \left[V_m^*(s_t, \vec{\boldsymbol{a}}_t) - V_m^\pi(s_t, \vec{\boldsymbol{a}}_t)\right] + \\
&\quad \sum_{t \notin \mathcal{B}_k} \left[V_m^*(s_t, \vec{\boldsymbol{a}}_t) - V_m^\pi(s_t, \vec{\boldsymbol{a}}_t)\right] \Bigg] \\
&\overset{(a)}{\le} \sum_{k=0}^{N_T-1} |\mathcal{B}_k| \cdot 1/(1-\gamma) + (t_{k+1} - t_k) \cdot \epsilon \\
&\overset{(b)}{=} \tilde{O}(C_d \epsilon^{-a}/(1-\gamma) + \epsilon T),
\end{aligned} \tag{5.16}$$

where the inequality (a) *holds due to the fact that* $0 \leq V^*(s_t, \vec{\boldsymbol{a}}_t), V_t^\pi(s_t, \vec{\boldsymbol{a}}_t) \leq 1/(1-\gamma)$*, and the equality* (b) *derives based on the definition of* $\mathcal{B}_k$ *and* N_T*. The regret (5.16) can be minimized by Regret*$(T) = \tilde{O}(C_d^{1/(a+1)}(1-\gamma)^{-1/(a+1)}T^{a/(a+1)})$*, which is achieved by choosing* $\epsilon = T^{-1/(a+1)}(1-\gamma)^{1/(a+1)}C_d^{-1/(a+1)}$.

To be specific, the regret of Algorithm 6 provided in Theorem 8 is $\tilde{O}((d+2-\gamma^m)\sqrt{T}/(1-\gamma)^2)$ *and it implies a sample complexity of exploration of*

$$\tilde{O}\left(\frac{(d+2-\gamma^m)^2}{\epsilon(1-\gamma)^3}\right). \tag{5.17}$$

As presented in Theorem 10, the sample complexity becomes $\tilde{O}((d+2-\gamma^m)/(\epsilon^2(1-\gamma)^3))$ *when* $d < \epsilon^{-1}$.

□

5.8 Conclusion

We proposed a provably efficient algorithm for solving parameterized-transition CDMDPs called value-targeted model regression for delayed MDP. The regret is proved to be upper bounded by $\tilde{O}((d+2-\gamma^m)\sqrt{T}/(1-\gamma)^2)$. We also proved a lower bound $\Omega(d\sqrt{T}/(1-\gamma)^{\frac{3}{2}})$ for solving parameterized-transition CDMDPs, which is close to our upper regret bound. Theoretical analysis demonstrates the effectiveness of our proposed algorithm in addressing the negative influence of delayed feedback on the agent's decision-making. Additionally, it results in a regret bound that remains independent of the size of the state and action spaces.

While our regret bound is indicative of the algorithm's efficiency, there is potential for further research to explore tighter regret bounds. Investigating refinements or alternative approaches that can yield regret bounds with smaller constants or dependencies on the problem parameters may contribute to a deeper understanding of the algorithm's performance. Additionally, extending the algorithm's capabilities to generalize across different types of MDPs or relaxing certain assumptions on the delay structure could enhance its versatility and widen its scope of application.

CHAPTER 6

SGD-based Method for Partial Gradient Information

6.1 Introduction

Online stochastic bandits represent a class of sequential decision-making problems where an agent makes actions and receives uncertain rewards. The applications in wireless communication range from client scheduling [148] to channel selection [149, 150, 151]. The goal of the agent is to maximize the cumulative rewards over n time steps by strategically selecting actions based on streaming data. A line of literature has developed effective algorithms for online stochastic bandits. Compared with the common methods, such as the upper confidence bound (UCB) bandit algorithm [147, 81] and online mirror descent (OMD) [152, 153], the SGD-based methods [154, 80, 155] can effectively reduce computational complexity by avoiding the matrix inverse operations when estimating the model parameter. However, several limitations persist in the current SGD-based methods for online stochastic bandits.

Firstly, existing online algorithms predominantly focus on linear models, while the general parametric model is unexplored. Secondly, prior SGD-based approaches only focused on expected regret bounds and did not tackle high probability bounds, leaving uncertainties about their algorithms in achieving desirable regret bounds when involving too many sub-optimal actions. Thirdly, current SGD-based approaches introduce bias in their estimators. This bias arises from a greedy action selection strategy at each time step, deviating from the conventional SGD approach that uniformly samples from all available data points. The presence of bias implies a larger divergence between the estimation and the ground truth, potentially compromising result robustness. In other words, different datasets may yield significantly different estimation results.

To address the above limitations, we consider SGD-based stochastic bandit problems with a general parametric model, emphasizing performance guarantees that hold high probability, an aspect lacking in current literature due to the considerable technical effort and modifications required to establish such guarantees. Specifically, the general parametric models usually involve complex optimization problems. It is vital to make reasonable but not strict assumptions about the model to guarantee feasible solutions. Furthermore, the statistical analysis associated with general parametric models requires precisely establishing corresponding i.i.d. random variables, necessitating the utilization of random matrix theory thorough analysis.

The contributions can be summarized as follows:

1. **General framework.** Our proposed method applies to stochastic bandits with a general parametric reward functions.

2. **High probability bound.** The proposed algorithm endows with a high probability regret-bound guarantee.

3. **Unbiased Action-elimination strategy.** We design a strategy to eliminate sufficiently sub-optimal actions gradually. The proposed algorithm uniformly selects the action from the current action subset at random. This strategy guarantees an unbiased estimation of model parameters, yielding a robust and desirable upper regret bound.

6.1.1 Related Work

UCB for linear bandit Auer et al. [156] provided the first analysis of linear bandit problem. They established a confidence set for the model parameter of the linear function. At each time step, the algorithm chooses an estimate from the confidence set and selects an action to maximize the cumulated reward. In other words, this estimate-action pair is chosen optimistically. The work

[156] achieves a regret bound of $\tilde{O}((\log |\mathcal{A}|)^{3/2}\text{poly}(d)\sqrt{n})^1$ where d is the feature dimension, n is the number of time steps, and $|\mathcal{A}|$ is the number of feasible actions. Dani et al. [82] developed variants of the algorithm based on upper confidence bounds (UCB) in [156]. It improves the regret bound to $\tilde{O}(d\sqrt{n})$. The main idea is to calculate the estimator of the model parameter at each time step, and then find the upper confidence bound of linear reward-function estimates. The same idea has been applied to web advertisement by Li et al. [157, 92] and Chu et al. [83]. Although UCB can effectively solve linear bandit problems, the time complexity of the algorithm depends quadratically on both d and n, i.e., $O(nd^2)$, because it needs to calculate the matrix inverse at every time step to estimate the model parameter. It turns out to be impractical when dealing with high-dimensional features.

TS for linear bandit Thompson Sampling (TS) has been a popular scheme for logistic bandit [158, 159, 160, 161], where the basic idea is to estimate the posterior of the model parameter based on new observation at each time step. TS is also successfully applied to linear bandit problem [80, 81]. Recently, Kveton et al. [93] proposed TS-based algorithms for the generalized linear model (GLM) of which linear bandit is a special case, enjoying $\tilde{O}(d\sqrt{n})$ total regret, where d is the feature dimension, n is the number of time steps. The essential idea in [93] is to draw a random sample from the approximated posterior distribution and select the action with the best estimates of this posterior. For the linear bandit problem, this algorithm has the time complexity of $O(nd^2)$ since it needs to calculate the matrix inverse to estimate the posterior and the covariance matrix of the posterior requires to be reweighed based on previous observations.

Online linear optimization with bandit feedback Online linear optimization with bandit feedback was originally introduced in [162, 163]. Dani et al. [164] firstly obtain optimal $O(d\sqrt{n})$ regret bounds by using the Exp2 (Expanded Exp) strategy. However, it still requires finding the estimates via the matrix inverse calculations. Following the Exp2 strategy, algorithms based on online mirror

$^1\tilde{O}$ ignores poly-logarithmic factors.

descent were investigated to reduce the time complexity. Mirror descent was pioneered in [165] as an off-line convex optimization method. The first application of mirror descent to online linear bandit was proposed by Abernethy et al. [166]. Abernethy et al. developed the first computationally efficient strategy and obtained a $O\left(d^2\sqrt{n}\right)$ regret. Bubeck et al. [153] further improved the regret bound to $O\left(d\sqrt{n}\right)$. Unfortunately, matrix inverse operations are still required in OMD-based algorithms. For more details, please refer to [167, 100].

SGD-based algorithm for bandit problem To overcome the time complexity issue of the above algorithms, online stochastic gradient descent (SGD) is verified as an efficient optimization algorithm for the model parameter estimation in stochastic linear bandit model by Kordaet et al. [154]. Even though the algorithm in [154] achieves an improvement of order $O(d)$ in time complexity, the regret performance suffers a loss of $O(\log^4 n)$, which is precisely given by $O(d \log^4 n \cdot \sqrt{n})$.

Besides stochastic linear bandit, a line of literature has applied SGD-based algorithm in contextual bandit problem [168, 80, 155, 169]. Although the above algorithms can be practically modified to solve stochastic bandits without contexts, the theoretical analysis of contextual bandits cannot directly be extended to the setting concerned in this chapter. The primary distinction lies in the fact that contextual bandits incorporate context information for optimizing decision-making problems. In contrast, stochastic bandits without contexts operate in a different version of action-reward feedback. This substantial difference involves varied forms of optimization when estimating the reward function. Moreover, these papers lack consideration for high probability bounds, leaving uncertainty about the achievability of their proposed regret bound with high confidence. Our work aims to explore the high probability bound of stochastic bandit problems.

Concentration analysis for SGD For the SGD-based bandit problem, concentration analysis for SGD plays an essential role in regret bound. In other words, the tight concentration analysis leads to a tight regret bound. Most of the literature (see e.g., [170, 171, 172, 173, 174]) on the performance of SGD focuses on the expected error rate which supports the regret analysis of SGD-based bandit problem in [80, 155]. Herein, the averaged SGD (ASGD) proposed by Polyak and Juditsky [172] is a popular method that averages iterates and establishes the asymptotic normality of the estimate. Under certain regularity assumptions, it can achieve the optimal rate $O(1/\sqrt{n})$ due to the central limit theorem (CLT), after n time steps of SGD.

Apart from the guarantees in the expected error rate, practitioners usually prefer high confidence guarantees, i.e., high-probability error bounds in the form of $\mathbb{P}(\|\hat{\theta} - \theta_\star\|_2 \geq \epsilon) \leq \delta$, where $\epsilon > 0, \delta \in (0, 1)$ can be arbitrarily small, and $\hat{\theta}$ is the estimator of $\theta_\star$. As pointed out by Lou et al. [175], bounds in expected error rate are typically too conservative to derive high-probability guarantees. Additionally, the confidence intervals derived from the CLT only remain asymptotically when the number of samples goes to infinity and cannot be used to provide rigorous sample complexity when δ goes to zero. Hence, further tail probability analyses (non-asymptotic) are needed.

Some widely known high-probability results include [173] by Rakhlin et al. who showed that for the strongly convex setting under sub-Gaussian noise assumptions, the estimator of T-round averaged SGD achieves $\|\widehat{\theta}_T - \theta_\star\|_2 \leq O(\sqrt{\log(\log(T)/\delta)/T})$ with high probability. Recently, the above bound has been improved to $O(\sqrt{\log(1/\delta)/T})$ by a line of literature, e.g., [176, 177, 178, 179, 180] and [181]. Most recently, Lou et al. [175] provided a similar bound under heavy-tailed noise assumptions, which is a more general case. Moreover, the best known state-of-the-art high probability bound of SGD-based algorithm is $O(d\sqrt{n \log(n \log n/\delta)})$ for contextual bandit problem [155]. However, these investigations are confined to the linear setting, lacking generalizability. This chapter seeks to fill this gap by undertaking a comprehensive examination of SGD-based algorithms within a broader bandit problem framework, delving into both theoretical and empirical dimensions. The comparison between our result and state-of-the-art is illustrated in Table 6.1. The first four papers lack consideration for high probability bounds. In contrast to [147], our bound demonstrates an improvement by a multiplicative factor of approximately $\sqrt{\log(n)}$. Notably, our method not only attains a near-optimal bound akin to [81] but also enjoys low computational complexity.

Table 6.1: Comparison of our main result and state-of-the-art.

Paper	Model	Algorithm	Regret	Computational complexity
[153]	Linear	OMD-based	$O(d\sqrt{n\log(n)})$	$O(nd^2)$
[80]	Linear	SGD-TS	$O(d\sqrt{n\log(n)})$	$O(nd)$
[155]	Linear	SGD-based	$O(d\sqrt{n\log(n\log(n))})$	$O(nd)$
[154]	Linear	SGD-based	$O(d\log^4 n\cdot\sqrt{n})$	$O(nd)$
[147]	Linear	UCB	$O\left(d\log(n)\sqrt{n} + \sqrt{dn\log\left(\frac{n}{\delta}\right)}\right)$	$O(nd^2)$
[81]	General parametric	UCB	$O(d\sqrt{n\log(n/\delta)})$	$O(n^2d^2)$
Theorem 11	**General parametric**	**SGD-based**	$\mathbf{O(d\sqrt{n\log(n/\delta)})}$	$\mathbf{O(nd)}$

6.2 Preliminaries

Let $\mathcal{D} \subset \mathbb{R}^d$ be a compact set of decisions the environment decides. At each time t, the learning algorithm determines a subset $\mathcal{A}_t \subseteq \mathcal{D}$ and the agent selects an action $x_t \in \mathcal{A}_t$, after which the agent observes a reward y_t.

We denote $\mathcal{H}_t$ as the history $(\mathcal{A}_1, x_1, y_1, \ldots, \mathcal{A}_{t-1}, x_{t-1}, y_{t-1}, \mathcal{A}_t)$ of observations available to the agent when choosing an action x_t. After choosing the action x_t, the agent receives a reward y_t that is a function with respect to a certain parameter $\theta_\star \in \mathbb{R}^d$, i.e., for all $x_t \in \mathcal{D}$, $y_t = r(\theta_\star, x_t) + \epsilon_t$ where ϵ_t denotes the noise. Generally, the vector $\theta_\star$ is unknown, though fixed.

We begin with two standard assumptions for most bandit problems [156]. The first assumption sets the range of the reward function.

Assumption 2 (Reward function) *Define a function $r : \mathbb{R}^d \times \mathcal{D} \to \mathbb{R}$. The reward function for bandit problem is represented as $r(\theta_\star, x)$ for all $x \in \mathcal{D}$ and a certain parameter $\theta_\star \in \mathbb{R}^d$ where*

$\|\theta_\star\|_2 \le S$ for $S > 0$.

Our second assumption ensures that observation noise is light-tailed. A wide range of noise, e.g., Gaussian and sub-Gaussian noise, is covered by this assumption.

Assumption 3 (Noise assumption) *For all $t \in [n], \epsilon_t = y_t - r(\theta_\star, x)$ conditioned on $\mathcal{H}_t$ is σ-sub-Gaussian, i.e., $\mathbb{E}[\exp(\lambda\epsilon_t)] \le \exp(\lambda^2\sigma^2/2)$ almost surely for all λ.*

We let $x^* \in \arg\max_{x\in\mathcal{D}}\ r(\theta_\star, x)$ denote the optimal action. The n period cumulative regret is $\mathrm{Reg}(n) = \sum_{t=1}^n [r(\theta_\star, x^*) - r(\theta_\star, x_t)]$ where $\{x_t : t \in [n]\}$ denote the actions.

6.3 Algorithm

We present our proposed algorithm in this section. To begin with, we need to estimate the model parameter in the bandit problem. The efficient estimation is intractable for a general function $r(\theta, x)$ unless we consider the bandit problem under some reasonable and mild assumptions. These assumptions ensure that stochastic gradient descent can efficiently and effectively apply to model parameter estimation. Before presenting assumptions, we start with the representation of the loss function in estimation.

Suppose that decisions $x_1, \dots x_n \in \mathcal{D}$ have been made, corresponding rewards are $y_1, \dots, y_n \in \mathbb{R}$. The loss function at i-th step is defined as $\ell_i(r(\theta, x_i), y_i),\ \forall\ \theta \in \mathbb{R}^d,\ i \in [n]$, which depends on data pair (x_i, y_i) and the reward function $r(\theta, x_i)$. To guarantee efficient and effective parameter estimation, we consider the problem of minimizing a smooth and convex function by stochastic gradient descent. Specifically, we make the following assumptions on the loss function $\ell_i(r(\theta, x_i), y_i)$. We abuse the notation $\ell_i(\theta)$ to represent $\ell_i(r(\theta, x_i), y_i)$ in the following.

Assumption 4 (Loss function) *We assume that the loss function $\ell_i(\theta)$ (6.3) with $i \in [n]$ satisfies, for some Lipschitz constants $L, L_G, L_H > 0$, convexity constant $\mu > 0$ and any points $\theta, \theta' \in \mathbb{R}^d$*

- *Convexity and smoothness:*

$$\ell_i(\theta') - \ell_i(\theta) - \langle \nabla \ell_i(\theta), \theta' - \theta \rangle \geq \frac{\mu}{2} \|\theta' - \theta\|_2^2,$$
$$\|\nabla \ell_i(\theta') - \nabla \ell_i(\theta)\|_2 \leq L_G \|\theta' - \theta\|_2,$$
$$|r(\theta', x_i) - r(\theta, x_i)| \leq L \|\theta' - \theta\|_2.$$

  In other words, $\ell_i(\theta)$ is L_G-smooth with respect to θ, and $r(\theta, x_i)$ is L-smooth with respect to θ.

- *Hessian-Smoothness: $\|\nabla^2 \ell_i(\theta') - \nabla^2 \ell_i(\theta)\|_2 \leq L_H \|\theta' - \theta\|_2$, which is equivalent to*

$$\begin{aligned} &\left\|\nabla \ell_i(\theta') - \nabla \ell_i(\theta) - \nabla^2 \ell_i(\theta)(\theta' - \theta)\right\|_2 \\ \leq &\frac{L_H}{2} \|\theta' - \theta\|_2^2. \end{aligned} \tag{6.1}$$

- *Bounded gradient: For the model parameter $\theta_\star$ in Assumption 2, we have $\mathbb{E}[\|\nabla \ell_i(\theta_\star)\|_2] \leq \mu \|\theta_\star\|_2$.*

Assumption 4 can be easily satisfied by a wide range of loss functions under various scenarios. An example of stochastic linear bandit problems that satisfies Assumption 4 is presented in Example 1 in Section 6.4.

Based on Assumption 4, we update the estimator of the model parameter via the mini-batch averaged SGD [172]. Additionally, we design an action-elimination strategy to gradually eliminate sufficiently sub-optimal actions in the learning process and maintain near-optimal actions. At each round of mini-batch averaged SGD, we uniformly and randomly select the action from the current action subset to guarantee the corresponding feature vectors are i.i.d. The details on designing proper action subsets are presented in Section 6.4, which is our main argument to guarantee near-optimal regret bound.

According to the above discussions, the algorithm is presented as follows. An estimate $\hat{\theta}$ to the ground-truth vector $\theta_\star$ can be constructed by

$$\hat{\theta} := \arg\min_{\theta} \mathcal{L}(\theta), \text{ where } \mathcal{L}(\theta) := \frac{1}{n} \sum_{i=1}^{n} \ell_i(\theta) \tag{6.2}$$

where $\ell_i(\theta)$ is the loss function.

We apply the mini-batch SGD to estimate the model parameter. Initialized at θ_0, the t-th iteration is computed by the mini-batch SGD with step size η_t. We define B as the mini-batch size and step sizes η_t will be discussed in our proposed algorithm. In each round t, the agent randomly selects an action $x_t \in \mathcal{A}_t$. The proposed SGD for general bandit is illustrated in Algorithm 7.

Our algorithm (Algorithm 7) is an exploration-exploitation modification of mini-batch averaged SGD, where the action-elimination strategy realizes the exploration-exploitation balancing. At a high level, each round consists of one inner loop over all mini-batch sizes B. Before initiating the inner loop, an action subset (referenced in line 4) is established. In line 4, the objective is to filter out actions from $\mathcal{A}_{t-1}$ whose rewards deviate significantly from the maximum action reward within $\mathcal{A}_{t-1}$ by solving the optimization problem (6.3). The remaining actions are then defined as $\mathcal{A}_t$. In the case of a nonconvex reward function, the problem (6.3) becomes intractable in general. However, when the reward function adheres to suitable statistical models (e.g., low-rank matrix), straightforward first-order methods are assured to discover a local minimum with a minimal number of iterations. This approach can still achieve low computational and sample complexities.

After getting an action set $\mathcal{A}_t$, the inner loop (line 5-9) uniformly and randomly selects an action from the action set $\mathcal{A}_t$ to guarantee the i.i.d. property of the action x_i and the reward y_i. It ensures the unbiased estimation of stochastic gradient. Subsequently, the inner loop (line 5-9) updates the stochastic gradient g_t that is used to form the iterate, i.e., θ_t. The analysis of our proposed algorithm is provided in the next section.

6.4 Main Theory

In this section, we present our main result, Theorem 11, which provides the sample complexity guarantee for Algorithm 7 in the general bandit problem under mild assumptions.

Algorithm 7: SGD-based Algorithm for General Stochastic Bandits

```
**Require:** Neighborhood radius $\beta_t, \forall\, t \in \{1, 2, \cdots, T\}$, number of
    outer iteration rounds $T$, mini-batch size $B$, step-size $\eta_t = \eta_0 t^{-\alpha}$
    where $\alpha \in (0, 1)$.
1: Initialize $\theta_0 \in \mathbb{R}^d$ such that $\|\theta_0\|_2 \le S$, $\bar{\theta}_0 = 0 \in \mathbb{R}^d$, $\mathcal{A}_0 = \mathcal{D}$,
   $\beta_0 = S$.
2: **for** $t = 1$ **to** $T$ **do**
3:   Initial $g_t = 0 \in \mathbb{R}^d$.
4:   Update action set as
       $$\begin{aligned}\mathcal{A}_t :=& \{x \in \mathcal{A}_{t-1} | \max_{a \in \mathcal{A}_{t-1}} r(\bar{\theta}_{t-1}, a) \\ & - r(\bar{\theta}_{t-1}, x) \le 2L\beta_{t-1}\}, \qquad (6.3)\end{aligned}$$
1
     where $L$ is defined in Assumption 4.
5:   **for** $i = (t-1)B + 1$ **to** $tB$ **do**
6:     Uniformly at random select an action $x_i \in \mathcal{A}_t$.
7:     Observe the reward $y_i$.
8:     Update $g_t = g_t + \frac{1}{B}\nabla \ell_i(\theta_{t-1})$.
9:   **end for**
10:  Update $\theta_t = \theta_{t-1} - \eta_t g_t$.
11:  Compute $\bar{\theta}_t = t^{-1}(\theta_t + (t-1)\bar{\theta}_{t-1})$
12: **end for**
13: **Output:** $\bar{\theta}_T$
```

We begin with notations used in the main theorem. Let

$$\Psi_{\chi,\alpha} = \int_1^\infty \exp\left(-\chi \int_1^z x^{-\alpha} dx\right) dz \leq C_{\chi,\alpha}, \tag{6.4}$$

where $C_{\chi,\alpha} > 0$, $\chi > 0$, and $0 < \alpha \leq 1$. We define $\lambda_* = \max_t \lambda_{\max}(\nabla^2 \ell_t(\theta_\star)),\ t \in [T]$ in the following. In Theorem 11, let $\sigma > 0$ denote the standard deviation of noise for the gradient caused by the noise $\{\epsilon_t\}$ under Assumption 3.

Theorem 11 *Under Assumptions 2, 3, and 4, we set*

$$\beta_t = C''_{\chi,\alpha} \sqrt{\frac{16 d L_H^2}{t\mu^3} \log\left(\frac{t}{\delta}\right)},\ t \in [T], \tag{6.5}$$

$\alpha = 1/2$, the mini-batch size $B = \mu\sigma^2 d / L_H^2$, and the initial step-size $\eta_0 \leq 1/\lambda_*$, Algorithm 7 achieves the following regret with probability at least $1 - \delta$,

$$\mathrm{Reg}(n) \leq \frac{2\mu\sigma^2 SLd}{L_H^2} + 32 \cdot \sigma C''_{\chi,\alpha} dL \sqrt{\frac{n}{\mu^2} \log\left(\frac{n}{\delta}\right)}, \tag{6.6}$$

where $n = TB$ and $C''_{\chi,\alpha} \asymp C_{\chi,\alpha}$ with $C_{\chi,\alpha}$ derived from (6.4) with $\chi = \mu\eta_0/2$ and $\alpha = 1/2$.

The first term of the regret bound (6.6) comprises a constant determined by the parameters. Specifically, a more concentrated tail distribution, reflected by a smaller σ^2, results in a diminished upper regret bound. If the slope (first-order derivative) and the curvature (second-order derivative) of the loss function don't change too rapidly across its domain, reflected by a smaller L and a larger L_H, the upper regret bound will be reduced. The dominant factor in the regret bound (6.6) is the second part. If the loss function's curvature changes more gradually (rapidly) across its domain, reflected by a larger μ, this leads to a reduced upper bound on regret. Thus, there is a trade-off between the first and second terms of the regret bound (6.6).

Example 1 (Linear Bandits) *Consider linear bandits where the function $r(\theta, x) = \theta^\top x$ is 2-smooth with respect to $\theta \in \mathbb{R}^d$ for a fixed $x \in \mathcal{D}$ since $\nabla_\theta r(\theta, x) = x$. The loss function of estimating θ can be given by*

$$\ell_i(\theta) = \frac{1}{2n}\left(\theta^\top x_i - y_i\right)^2 + \frac{\mu}{2}\|\theta\|_2^2 \tag{6.7}$$

for $i \in [n]$ and $\mu > 0$. Let $\mu = 1$, Algorithm 7 solves linear bandits and achieves the following regret with probability at least $1 - \delta$,

$$\mathrm{Reg}(n) \leq \frac{4\sigma^2 S d}{L_H^2} + 64 \cdot \sigma C''_{\chi,\alpha} d \sqrt{n \log\left(\frac{n}{\delta}\right)},$$

where $n = TB$ and $C''_{\chi,\alpha} \asymp C_{\chi,\alpha}$ with $C_{\chi,\alpha}$ defined in (6.4), which is comparable to the result for linear bandit in [147] $O(d\sqrt{n \log(n/\delta)})$.

6.5 Proof Sketch

This section overviews several key mechanisms behind the regret bound in Theorem 11. We defer the full proof of Theorem 11 to Appendix E.2. Our analysis depends on the following three key steps.

Step 1: Confident set construction. Define the confidence region at round t to be

$$\mathcal{B}_t := \left\{\nu : \left\|\nu - \bar{\theta}_t\right\|_2 \leq \beta_t.\right\} \tag{6.8}$$

According to the choice of β_t (6.5) in $\mathcal{B}_t$ (6.8), we show that $\theta_\star$ always remains inside this region for all times t, with high probability. Please refer to Lemma 4 for details of choosing proper β_t. It ensures that the averaged iterate $\bar{\theta}_t$ converges to the ground truth $\theta_\star$. The value of β_t (6.5) plays an important role in bounding the regret.

Step 2: Action set selection. We further show that $\forall t \in \mathbb{N}$ the set $\mathcal{A}_t$ (6.3) contains the optimal action x^* with high probability. Please refer to Lemma 5 for details. It guarantees that our proposed algorithm chooses near-optimal action and eliminates actions that are sufficiently suboptimal as time goes by. At each t-round of Algorithm 7, the regret of action $x_i \in \mathcal{A}_t$ is bounded by β_{t-1} with a positive constant, which facilitates regret analysis in the next step.

Step 3: Regret bound. Combining the above two steps, the total regret can be bounded by the sum of the regrets for each selected action. At each t-round of SGD-Ridge, an action is selected from the action set $\mathcal{A}_t$ that is updated at the beginning of the inner loop. Herein, the regret of action $x_i \in \mathcal{A}_t$ is bounded by $4L\beta_{t-1}$.

Confident set construction. To prove the main theorem, the first step is to show that the confidence region is appropriate. To be specific, $\theta_\star$ always remains inside this region for all times t, with high probability.

Lemma 4 *Let $\delta > 0$ and the choice of β_t (6.5) in $\mathcal{B}_t$ (6.8), we have $\mathbb{P}\left(\forall t \in \mathbb{N}, \theta_\star \in \mathcal{B}_t\right) \geq 1 - \delta$.*

Lemma 4 implies that

$$\left\|\bar{\theta}_t - \theta_\star\right\|_2 \leq \beta_t, \tag{6.9}$$

with probability at least $1 - \delta$, which facilitates to regret analysis in Proposition 1.

***Proof**: Please refer to Appendix E.2 for details.* □

Action set selection. The following lemma shows that $\forall t \in \mathbb{N}$ the set $\mathcal{A}_t$ (6.3) contains the optimal action x^* with high probability.

Lemma 5 *Let $\delta > 0$ and recall the action set $\mathcal{A}_t$ (6.3), we have*

$$\mathbb{P}\left(\forall t \in \mathbb{N}, \quad x^* \in \mathcal{A}_t\right) \geq 1 - \delta.$$

***Proof**: Please refer to Appendix E.3 for details.*

□

Regret bound. We provide regret analysis in the following proposition.

Proposition 1 *Set the batch-size as $B = 16L^4d/\lambda$. Let $\mathrm{reg}_i = \theta_\star^\top x^* - \theta_\star^\top x_i$ denote the instantaneous regret of $x_i \in \mathcal{A}_t$ acquired by the Algorithm 7 on round t. With probability at least $1 - \delta$, the regret achieves $\mathrm{Reg}(TB) = \sum_{t=1}^{T}\sum_{i=(t-1)B+1}^{tB} \mathrm{reg}_i \leq 16 \cdot \left(\frac{2dSL^5}{\lambda} + C_\alpha \cdot dSL^3\sqrt{\frac{6TB\log(TB/\delta)}{\lambda}}\right)$, where $C > 0$ depends on α where $\alpha \in (0, 1)$.*

***Proof**: Please refer to Appendix E.8 for details.*

□

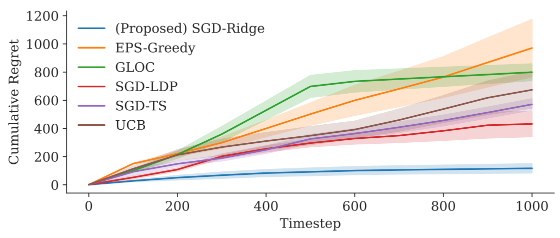
6.6 Simulation Results

In this section, we provide the experimental results with industry-standard synthetic datasets for both linear and logistic bandit.

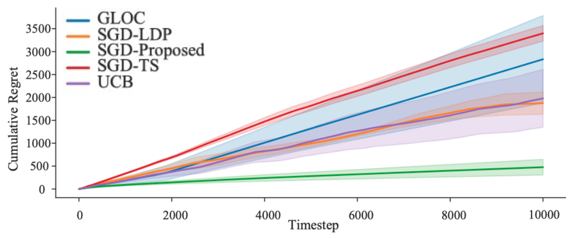
- Linear bandit: We set the number of rounds $T = 1000$ and conduct simulations on the parameter: $K = 30$ (K is the number of action) and $d = 2$. We build linear bandit models, where the feature vectors $\{x_i\}$ and the true model parameter $\theta_\star$ are drawn i.i.d. from Gaussian distribution $\mathcal{N}(0, I_d)$ and normalize to $\|x_i\|_2 = 1,\ \|\theta_\star\|_2 = 1$. The loss function for a linear bandit is the form of (6.7) with the regularization parameter μ.
- Logistic bandit: We set the number of rounds $T = 10000$ and conduct simulations on the parameter: $K = 40$ and $d = 2$. We draw $\{x_i\}$ and the true model parameter $\theta_\star$ iid from uniform distribution in the interval of $[-\frac{1}{\sqrt{d}}, \frac{1}{\sqrt{d}}]$. We build a logistic model on the dataset and draw random rewards y_t from a Bernoulli distribution with mean $1/(1 + \exp(x_i^\top \theta^*))$. The loss function for logistics is the form of

$$\ell_i(\theta) = \frac{1}{2n}\left(\left(1 + \exp(\theta^\top x_i)\right)^{-1} - y_i\right)^2 + \frac{\mu}{2}\|\theta\|_2^2 \tag{6.10}$$

for $i \in [n]$ and μ, which satisfies Assumption 4.



(a) Linear bandit



(b) Logistic bandit

Figure 6.1: The cumulative regret vs. time-step of different algorithms.

To ensure a fair comparison, we evaluate SGD-Ridge (for linear bandit) and SGD-Proposed (for logistic bandit) alongside established methods including ϵ-greedy [156, 182], GLOC [95],

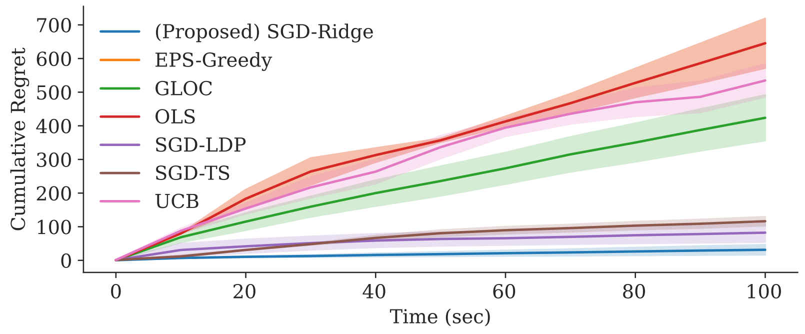


Figure 6.2: The cumulative regret vs. computational time of different algorithms.

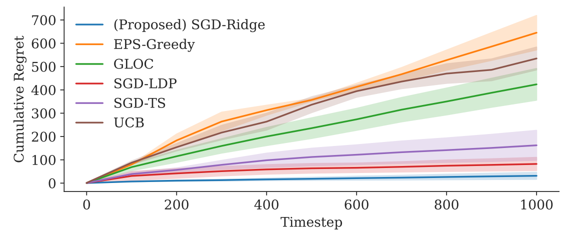
SGD-LDP [155], SGD-TS [80], and UCB [92], with their codes available publicly. We standardize noise levels, considering both privacy noise [155] and reward noise.

Parameter tuning is conducted uniformly across all algorithms. For GLOC and UCB-GLM, we explore exploration rates in $0.01, 0.1, 1, 5, 10$. The exploration probability of ϵ-greedy is set as $\frac{c}{\sqrt{t}}$ at the t-th iteration, with c selected from $0.01, 0.1, 1, 5, 10$. For SGD-based algorithms, we set mini-batch size $B = 16d/\lambda$, with λ tuned in $0.01, 0.1, 1, 5, 10$. The parameter β_t (6.5) is set as $\sqrt{4d\log(\frac{t}{10^{-3}})/t}$. Step size η_t is $\eta_0/\sqrt{t}$, where η_0 is chosen from $0.01, 0.05, 0.1, 0.5, 1, 5, 10$. Regularization parameter μ for each algorithm is searched from $0.01, 0.05, 0.1, 0.15$.

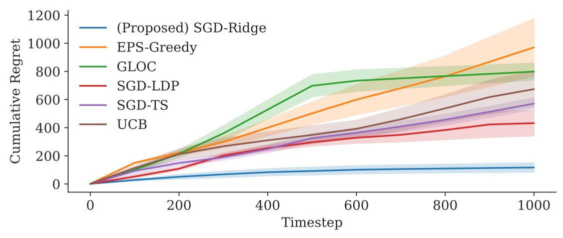
We perform experiments 30 times and plot the mean and standard deviation of their regrets, which are illustrated in Fig 6.1 and Fig 6.2. It shows that our proposed algorithms outperform state-of-the-art approaches. The beneficial performance is due to a good balance between exploitation and exploration via an action-elimination strategy and efficient estimation via the mini-batch SGD method. Moreover, random selection during each mini-batch guarantees an unbiased gradient, which outperforms greedy selection used by Han et al. [155].

All the algorithms are required to solve similar optimization problems as (6.3) which aims to find the proper arm to pull. The advantage of our proposed algorithm on computational time mostly comes from the efficiency of estimating parameters via SGD.

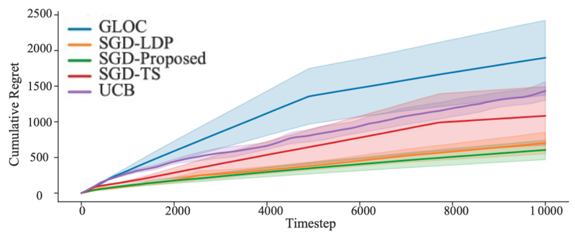
We further provide simulated experiments in industry-standard synthetic datasets for both linear



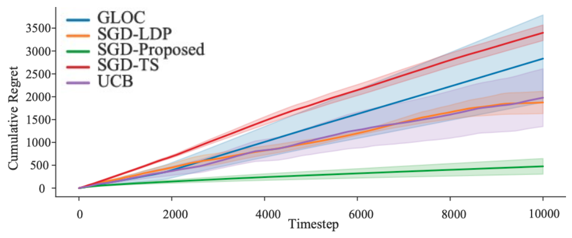
(a) Linear bandit: $K = 10,\ d = 2$



(b) Linear bandit: $K = 30,\ d = 2$



(c) Logistic bandit: $K = 20,\ d = 2$



(d) Logistic bandit: $K = 40,\ d = 2$

Figure 6.3: Cumulative regret of different algorithms for linear and logistic bandits for timestep. (a) and (b) illustrate linear bandits with $K = 10$ and $K = 30$, respectively. (c) and (d) illustrate logistic bandits with $K = 20$ and $K = 40$, respectively.

and logistic bandit. We plot the mean and standard deviation of their regrets, which are illustrated in Fig 6.3. Our proposed algorithms outperform state-of-the-art approaches and maintain advantages with larger K in both linear and logistic bandits. The beneficial performance is due to a good balance between exploitation and exploration via an action-elimination strategy. Moreover, random selection during each mini-batch guarantees an unbiased gradient, which outperforms greedy selection used by Han et al. [155].

6.7 Conclusion

In this chapter, we present the SGD-based algorithm for generalized stochastic bandits, focusing on regret-bound guarantees that hold with high probability. In addition to theoretical validation, we conducted experiments to showcase the practical effectiveness of our proposed algorithm. The results highlight the improved performance and versatility of our approach in handling stochastic bandits with general parametric reward functions. By addressing the identified limitations of current works, our algorithm presents a promising advancement in the application of SGD to stochastic bandit problems, paving the way for more robust and efficient solutions in real-world scenarios. In addition to the stochastic bandit problems addressed in this chapter, it is interesting to investigate more complex and general reward models, such as neural networks in the future.

APPENDIX A

Proofs for Chapter 2

A.1 Proof of Theorem 1

By extending Lemma 4 in [13], we conclude that $\mathbb{E}[\|\nabla J(\hat{\boldsymbol{\theta}})\|^2]$ depends on $\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla \widetilde{J}_{\mathcal{B}}(\boldsymbol{\theta}_t) - \nabla J(\boldsymbol{\theta}_t)\|^2]$. We first present a few vital lemmas to complete the proof of theorem 1. To be specific, Lemma 6 characterizes the bounds on $\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla \widetilde{J}_{\mathcal{B}}(\boldsymbol{\theta}_t) - \nabla J(\boldsymbol{\theta}_t)\|^2]$.

Lemma 6 *Denote N_ℓ^i as the number of the neighborhood nodes concerning $i \in \mathcal{V}$ sampling uniformly at random at ℓ-th layer. The cached nodes in the set $\mathcal{C}$ with the size of $|\mathcal{C}|$ are sampled without replacement according to p_v^{cache}. The dimension of the node feature is denoted as n and the size of the min-batch is denoted as B. Define $\widetilde{C} = |\mathcal{C}|/|\mathcal{V}|$ and $C_d = \sum_{v_i\in\mathcal{V}}\deg(v_i)/|\mathcal{V}|$ with the constant $c > 0$. Under Assumption 1, the expected mean-square error of stochastic gradient $\nabla\widetilde{J}_{\mathcal{B}}(\boldsymbol{\theta})$ derived from Algorithm 1 to the full gradient is bounded by*

$$\begin{aligned}\mathrm{MSE} :=& \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left\|\nabla \widetilde{J}_{\mathcal{B}}(\boldsymbol{\theta}_t) - \nabla J(\boldsymbol{\theta}_t)\right\|^2\right]\\ \leq& O\left(L_f'^2\frac{\log(4n/\delta)+1/2}{B}\right) + O\left(L_f'^2 L_g^4\frac{\log(4n/\delta)+1/2}{c\widetilde{C}C_d N_1^j N_2^i}\right)\\ &+ O\left(L_g'^2 L_f^2\frac{\log(4n/\delta)}{c\widetilde{C}C_d N_1^j N_2^i}\right). \end{aligned} \tag{A.1.1}$$

Proof: *According to the inequality, $\|\boldsymbol{a}+\boldsymbol{b}\|^2 \leq 2\|\boldsymbol{a}\|^2+2\|\boldsymbol{b}\|^2$, MSE (A.1.1) can be decomposed* *into two parts:*

$$\begin{aligned}\mathrm{MSE} :=& \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left\|\nabla \widetilde{J}_{\mathcal{B}}(\boldsymbol{\theta}_t) - \nabla J(\boldsymbol{\theta}_t)\right\|^2\right] \\ \leq& \frac{2}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left\|\nabla \widetilde{J}_{\mathcal{B}}\left(\boldsymbol{\theta}_t\right) - \nabla J_{\mathcal{B}}\left(\boldsymbol{\theta}_t\right)\right\|^2\right] \\ &+ \frac{2}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left\|\nabla J_{\mathcal{B}}\left(\boldsymbol{\theta}_t\right) - \nabla J\left(\boldsymbol{\theta}_t\right)\right\|^2\right] \end{aligned} \tag{A.1.2}$$

Two terms in (A.1.2) are bounded by Lemma 7 and 8, respectively. □

Lemma 7 *Based on the notations in Lemma 6, with probability exceeding* $1-\delta$ *we have*

$$\begin{aligned}&\mathbb{E}[\|\nabla \widetilde{J}_{\mathcal{B}}(\boldsymbol{\theta}) - \nabla J_{\mathcal{B}}(\boldsymbol{\theta})\|^2] \\ \leq& 128 L_f^{'2} L_g^4 \frac{\log(4n/\delta)+1/2}{c\widetilde{C}C_d N_1^j N_2^i} + 64 L_g^{'2} L_f^2 \frac{\log(4n/\delta)}{c\widetilde{C}C_d N_1^j N_2^i}.\end{aligned} \tag{A.1.3}$$

***Proof**: The proof of Lemma 7 is provided in Appendix A.2.* □

Lemma 8 *Based on the notations in Lemma 6, with probability exceeding* $1-\delta$ *we have*

$$\mathbb{E}[\|\nabla J_{\mathcal{B}}(\boldsymbol{\theta}) - \nabla J(\boldsymbol{\theta})\|^2] \leq 128 L_f^{'2} \frac{\log(4n/\delta)+1/2}{B}. \tag{A.1.4}$$

***Proof**: Besides the definition of* $\nabla J_{\mathcal{B}}(\boldsymbol{\theta})$ *(A.2.1),* $\nabla J(\boldsymbol{\theta})$ *is given as*

$$\nabla J(\boldsymbol{\theta}) = \frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}} A_1^i A_2^i, \tag{A.1.5}$$

where A_1^i, A_2^i *are represented by (A.2.2) and (A.2.3), respectively.*

For simplicity, we denote

$$\mathbb{E}_{\mathcal{V}_{\mathcal{B}}\sim\mathcal{V}}\left[\mathbb{E}_{j\sim\mathcal{N}(i),\forall i\in\mathcal{V}_{\mathcal{B}}}\left[\mathbb{E}_{k\sim\mathcal{N}(j),\forall j\in\mathcal{N}(i)}[\cdot]\right]\right]$$

as $\mathbb{E}[\cdot]$.

$$\begin{aligned}&\mathbb{E}[\|\nabla J_{\mathcal{B}}(\boldsymbol{\theta}) - \nabla J(\boldsymbol{\theta})\|^2] \\ \leq& \mathbb{E}\left[\left\|\frac{1}{B}\sum_{i\in\mathcal{V}_{\mathcal{B}}} A_1^i A_2^i - \frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}} A_1^i A_2^i\right\|^2\right], \\ \leq& 128 L_f^{'2} \frac{\log(4n/\delta)+1/2}{c\widetilde{C}C_d N_1^j N_2^i},\end{aligned} \tag{A.1.6}$$

where the last inequality comes from Lemma 9. □

A.2 Proof of Lemma 7

To bound $\mathbb{E}[\|\nabla \widetilde{J}_{\mathcal{B}}(\boldsymbol{\theta}) - \nabla J_{\mathcal{B}}(\boldsymbol{\theta})\|^2]$, we begin with the definition of $\nabla \widetilde{J}_{\mathcal{B}}(\boldsymbol{\theta})$ and $\nabla J_{\mathcal{B}}(\boldsymbol{\theta})$, which is given by

$$\nabla J_{\mathcal{B}}(\boldsymbol{\theta}) = \frac{1}{B} \sum_{i \in \mathcal{V}_{\mathcal{B}}} A_1^i A_2^i, \tag{A.2.1}$$

where

$$A_1^i = \nabla f_i \left(\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \frac{1}{|\mathcal{N}(j)|} \sum_{k \in \mathcal{N}(j)} g_{jk}(\boldsymbol{\theta}) \right), \tag{A.2.2}$$

$$A_2^i = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \frac{1}{|\mathcal{N}(j)|} \sum_{k \in \mathcal{N}(j)} \nabla g_{jk}(\boldsymbol{\theta}). \tag{A.2.3}$$

$$\nabla \widetilde{J}_{\mathcal{B}}(\boldsymbol{\theta}) = \frac{1}{B} \sum_{i \in \mathcal{V}_{\mathcal{B}}} B_1^i B_2^i \tag{A.2.4}$$

where

$$B_1^i = \nabla f_i \left(\frac{1}{|\mathcal{N}_2^{\mathcal{C}}(i)|} \sum_{j \in \mathcal{N}_2^{\mathcal{C}}(i)} \frac{1}{|\mathcal{N}_1^{\mathcal{C}}(j)|} \sum_{k \in \mathcal{N}_1^{\mathcal{C}}(j)} p_k^{(1)} g_{jk}(\boldsymbol{\theta}) \right), \tag{A.2.5}$$

$$B_2^i = \frac{1}{|\mathcal{N}_2^{\mathcal{C}}(i)|} \sum_{j \in \mathcal{N}_2^{\mathcal{C}}(i)} \frac{1}{|\mathcal{N}_1^{\mathcal{C}}(j)|} \sum_{k \in \mathcal{N}_1^{\mathcal{C}}(j)} p_k^{(1)} \nabla g_{jk}(\boldsymbol{\theta}). \tag{A.2.6}$$

For simplicity, we denote

$$\mathbb{E}_{\mathcal{V}_{\mathcal{B}} \sim \mathcal{V}} \left[\mathbb{E}_{j \sim \mathcal{N}(i), \forall i \in \mathcal{V}_{\mathcal{B}}} [\mathbb{E}_{k \sim \mathcal{N}(j), \forall j \in \mathcal{N}(i)} [\cdot]] \right]$$

as $\mathbb{E}[\cdot]$.

Based on the inequalities $\left\| \frac{1}{n} \sum_{i=1}^n \boldsymbol{a}_i \right\| \leq \frac{1}{n} \sum_{i=1}^n \|\boldsymbol{a}_i\|$, $\|\boldsymbol{a} + \boldsymbol{b}\|^2 \leq 2\|\boldsymbol{a}\| + 2\|\boldsymbol{b}\|$, and $\|\boldsymbol{a}\boldsymbol{b}\| \leq$

$\|\boldsymbol{a}\|\|\boldsymbol{b}\|$, we arrive

$$\begin{aligned}
&\mathbb{E}[\|\nabla \widetilde{J}_{\mathcal{B}}(\boldsymbol{\theta}) - \nabla J_{\mathcal{B}}(\boldsymbol{\theta})\|^2] \\
=&2\mathbb{E}\left[\left\|B_1^i\right\|^2\right]\mathbb{E}\left[\left\|B_2^i - A_2^i\right\|^2\right] \\
&+ 2\mathbb{E}\left[\left\|B_1^i - A_1^i\right\|^2\right]\mathbb{E}\left[\left\|A_2^i\right\|^2\right].
\end{aligned} \tag{A.2.7}$$

We shall bound two terms in (A.2.7).

1. The first term: for $\mathbb{E}\left[\left\|B_1^i\right\|^2\right]$, we have

$$\mathbb{E}\left[\left\|B_1^i\right\|^2\right] \le L_f^2. \tag{A.2.8}$$

In terms of $\mathbb{E}\left[\left\|B_2^i - A_2^i\right\|^2\right]$, we have

$$\begin{aligned}
&\mathbb{E}\left[\left\|B_2^i - A_2^i\right\|^2\right] \\
=&L_f^{\prime 2}\mathbb{E}\left[\left\| \sum_{j\in\mathcal{N}_2^{\mathcal{C}}(i)} \sum_{k\in\mathcal{N}_1^{\mathcal{C}}(j)} \frac{p_k^{(1)}}{N_{\mathcal{C}_2}^i N_{\mathcal{C}_1}^j} \nabla g_{jk}(\boldsymbol{\theta}) \right.\right. \\
&\left.\left. - \sum_{j\in\mathcal{N}(i)} \sum_{k\in\mathcal{N}(j)} \frac{1}{(N^i)^2} \nabla g_{jk}(\boldsymbol{\theta}) \right\|^2\right] \\
\le&64 L_g^{\prime 2} \frac{\log(4n/\delta)}{c\widetilde{C}C_d N_1^j N_2^i},
\end{aligned} \tag{A.2.9}$$

where the last inequality comes from Lemma 9.

2. The second term: for $\mathbb{E}\left[\left\|A_2^i\right\|^2\right]$, we have

$$\mathbb{E}\left[\left\|A_2^i\right\|^2\right] \le L_g^2. \tag{A.2.10}$$

In terms of $\mathbb{E}\left[\|B_1^i - A_1^i\|^2\right]$, we have

$$\begin{aligned}
&\mathbb{E}\left[\|B_1^i - A_1^i\|^2\right] \\
=&L_f'^2\mathbb{E}\Bigg[\Bigg\| \sum_{j\in\mathcal{N}_2^{\mathcal{C}}(i)} \sum_{k\in\mathcal{N}_1^{\mathcal{C}}(j)} \frac{p_k^{(1)}}{N_{\mathcal{C}_2}^i N_{\mathcal{C}_1}^j} g_{jk}(\boldsymbol{\theta}) \\
&- \sum_{j\in\mathcal{N}(i)} \sum_{k\in\mathcal{N}(j)} \frac{1}{(N^i)^2} g_{jk}(\boldsymbol{\theta})\Bigg\|^2\Bigg] \\
\leq&128L_g^2L_f'^2\frac{\log(4n/\delta)+1/2}{c\widetilde{C}C_dN_1^jN_2^i},
\end{aligned} \tag{A.2.11}$$

where the last inequality comes from Lemma 9.

By integrating inequalities (A.2.8), (A.2.11), (A.2.10), (A.2.9), it yields

$$\begin{aligned}
&\mathbb{E}[\|\nabla\widetilde{J}_{\mathcal{B}}(\boldsymbol{\theta}) - \nabla J_{\mathcal{B}}(\boldsymbol{\theta})\|^2] \\
\leq&128L_f'^2L_g^4\frac{\log(4n/\delta)+1/2}{c\widetilde{C}C_dN_1^jN_2^i} + 64L_g'^2L_f^2\frac{\log(4n/\delta)}{c\widetilde{C}C_dN_1^jN_2^i}.
\end{aligned} \tag{A.2.12}$$

Lemma 9 *Based on the notations in Lemma 6, with probability* $1-\delta$ *we have*

$$\begin{aligned}
&\mathbb{E}\Bigg[\Bigg\| \sum_{j\in\mathcal{N}_2^{\mathcal{C}}(i)} \sum_{k\in\mathcal{N}_1^{\mathcal{C}}(j)} \frac{p_k^{(1)}}{N_{\mathcal{C}_2}^i N_{\mathcal{C}_1}^j} g_{jk}(\boldsymbol{\theta}) \\
&- \mathbb{E}_{j\sim\mathcal{N}(i),k\sim\mathcal{N}(j)}[g_{jk}(\boldsymbol{\theta})]\Bigg\|^2\Bigg] \\
\leq&8\sqrt{2}L_g\sqrt{\frac{\log(4n/\delta)+1/2}{c\widetilde{C}C_dN_1^jN_2^i}} \quad \forall\, i\in\mathcal{V}
\end{aligned} \tag{A.2.13}$$

$$\begin{aligned}
&\mathbb{E}\Bigg[\Bigg\| \sum_{j\in\mathcal{N}_2^{\mathcal{C}}(i)} \sum_{k\in\mathcal{N}_1^{\mathcal{C}}(j)} \frac{p_k^{(1)}}{N_{\mathcal{C}_2}^i N_{\mathcal{C}_1}^j} \nabla g_{jk}(\boldsymbol{\theta}) \\
&- \mathbb{E}_{j\sim\mathcal{N}(i),k\sim\mathcal{N}(j)}[\nabla g_{jk}(\boldsymbol{\theta})]\Bigg\|^2\Bigg] \\
\leq&8L_g'\sqrt{\frac{\log(4n/\delta)}{c\widetilde{C}C_dN_1^jN_2^i}} \quad \forall\, i\in\mathcal{V},
\end{aligned} \tag{A.2.14}$$

where $C_d = \sum_{v_i\in\mathcal{V}}\deg(v_i)/|\mathcal{V}|$ *with the constant* $c>0$.

Proof: *The proof is derived from the proof of Lemma 6 in [183] and Lemma 10 and 11 in [13]. Based on the definition of $p_k^{(1)}$, for $i \in \mathcal{V}$, we have*

$$\begin{aligned}&\mathbb{E}\left[\sum_{j\in\mathcal{N}_2^{\mathcal{C}}(i)}\sum_{k\in\mathcal{N}_1^{\mathcal{C}}(j)}\frac{p_k^{(1)}}{N_{\mathcal{C}_2}^i N_{\mathcal{C}_1}^j}g_{jk}(\boldsymbol{\theta})\right]\\ =&\mathbb{E}_{j\sim\mathcal{N}(i),k\sim\mathcal{N}(j)}[g_{jk}(\boldsymbol{\theta})]. \qquad \text{(A.2.15)}\end{aligned}$$

Similarly, there is

$$\begin{aligned}&\mathbb{E}\left[\sum_{j\in\mathcal{N}_2^{\mathcal{C}}(i)}\sum_{k\in\mathcal{N}_1^{\mathcal{C}}(j)}\frac{p_k^{(1)}}{N_{\mathcal{C}_2}^i N_{\mathcal{C}_1}^j}\nabla g_{jk}(\boldsymbol{\theta})\right]\\ =&\mathbb{E}_{j\sim\mathcal{N}(i),k\sim\mathcal{N}(j)}[\nabla g_{jk}(\boldsymbol{\theta})]. \qquad \text{(A.2.16)}\end{aligned}$$

The value of $N_{\mathcal{C}_1}^j$ and $N_{\mathcal{C}_2}^j$ depend on the predefined number of neighborhood nodes sampled uniformly at random in each layer, i.e., N_2^i, N_1^j, the ratio of cached nodes to the whole graph nodes, i.e., $\widetilde{C} = |\mathcal{C}|/|\mathcal{V}|$ and the sampling probability $\mathcal{P}$. Thus, $N_{\mathcal{C}_1}^j$ can be approximated by $N_{\mathcal{C}_1}^j = c\widetilde{C}C_dN_1^j$ where $C_d = \sum_{v_i\in\mathcal{V}}\deg(v_i)/|\mathcal{V}|$ with the constant $c > 0$. Based on the above, the proof can be completed by extending the proof of Lemma 10 and 11 in [13].

Given the sampled set $\mathcal{S}_1, \mathcal{S}_2$ and

$$V_S(\boldsymbol{\theta}) = \frac{1}{|S_1|\cdot|S_2|}\sum_{i\in\mathcal{S}_1}\sum_{j\in\mathcal{S}_2}V_{ij}(\boldsymbol{\theta}),$$

where $V_{ij}(\boldsymbol{\theta}) \in \mathbb{R}^n$ is L_v-Lipschitz continuous for all i, j, the proof can be completed via Bernstein's bound with a sub-Gaussian tail, given by we have

$$\begin{aligned}&\mathbb{P}\left(\|V_S(\boldsymbol{\theta}) - \mathbb{E}[V_S(\boldsymbol{\theta})]\| \geq \epsilon\right)\\ \leq&4n\cdot\exp\left(-\frac{\epsilon^2|S_1|\cdot|S_2|}{64L_v^2}+\frac{1}{2}\right), \qquad \text{(A.2.17)}\end{aligned}$$

where $\epsilon \leq 2L_v$.

Finally, let δ as the upper bound of Bernstein's inequality

$$\delta = 4n\cdot\exp\left(-\frac{\epsilon^2|S_1|\cdot|S_2|}{64L_v^2}+\frac{1}{2}\right). \qquad \text{(A.2.18)}$$

Therefore, we have

$$\epsilon = 8\sqrt{2}L_v\sqrt{\frac{\log(4n/\delta) + 1/2}{|S_1| \cdot |S_2|}}. \tag{A.2.19}$$

The inequality (A.2.14) can be clarified similarly. □

APPENDIX B

Supplementary for Chapter 3

B.1 Dataset Statistics

	Train	Dev	Test
Natural Questions	79168	8757	3610
TriviaQA	78785	8837	11313

Table B.1: Dataset Statistics.

In Fig. B.1, we illustrate the AMR graph statistics in Natural Questions (NQ) and TriviaQA datasets. To better illustrate the structure of the shortest path, we also conduct some experiments to show the statistic of the shortest path in the AMR graph, see Fig B.2. We analyze the shortest single source paths (SSSPs) in the AMR graphs of documents and try to establish the connection between question contexts and document contexts. The analysis reveals a notable trend in the AMR graphs of documents, indicating that certain negative documents cannot establish adequate connections to the question context within their text. This pattern brings insights into the encoding process to enhance reranking performance.

B.2 Simulation Results with Different GNN Models.

Besides the GCN [53] model considered in the main manuscript, we compare the simulation results with different GNN models in this section. Specifically, under the same setting as the GCN model in

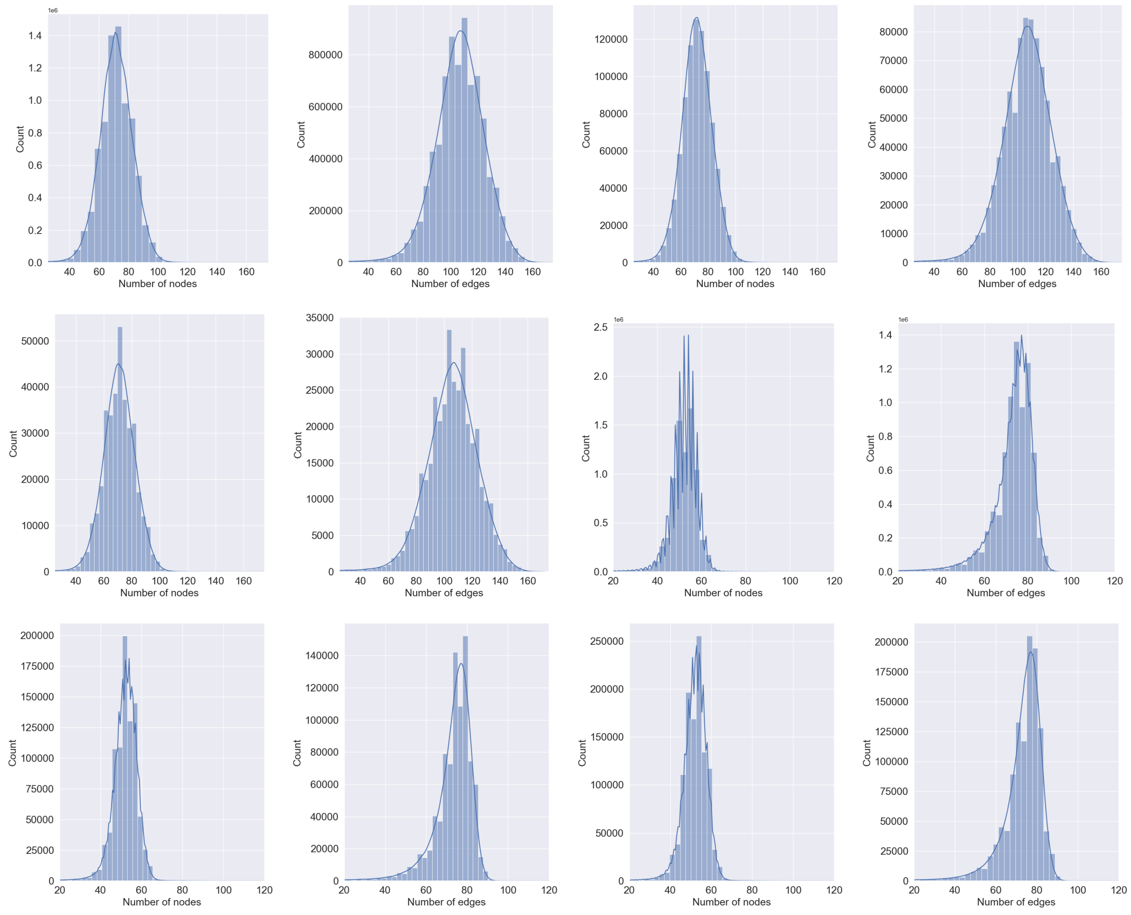


Figure B.1: Number of nodes and edges in AMR graphs in train/dev/test set of dataset NQ and TQA.

Ember (HPs-T) from Table 3.2, we use GAT [184] with additional parameter number of heads being 8, GraphSage [185] with the aggregation choice being 'lstm', and GIN [186] with the aggregation choice being 'mean'. The comparison results are illustrated in Table B.2.

For the convenience of comparison, we directly add two results from Section 3.4.2, i.e., BART-GST and GCN (i.e., Ember (HPs-T) in Table 3.2). It shows that the GCN model still outperforms in most cases. This may be due to the document graphs considered in our paper being very small, while the advanced GNN model usually targets handling thousands, or millions of nodes in the graph. Besides, our model has already taken the edge feature into consideration, which may lead to

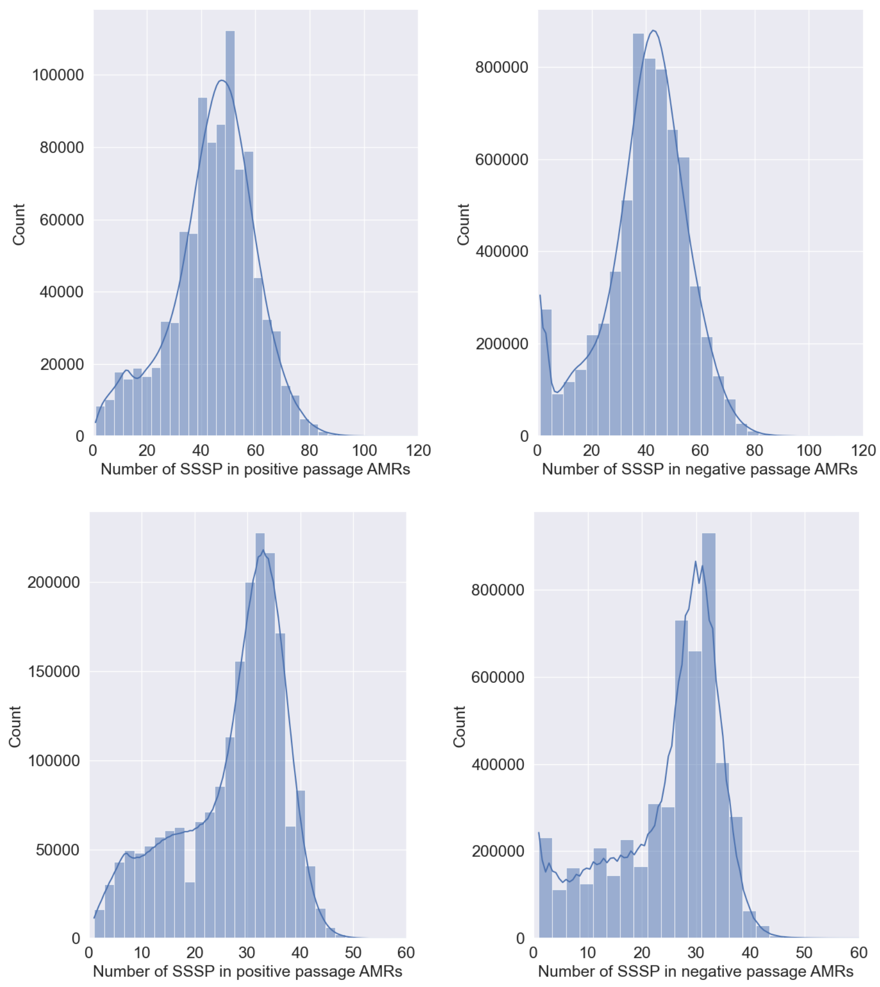


Figure B.2: Number of SSSPs AMR graphs in train set of dataset NQ and TQA.

overfitting if introducing more weight parameters.

B.3 Qualitative Examples

We take the ranking scores given by palm 2 L as a baseline to investigate how the graph-based model benefits reranking in Open-Domain Question Answering. Since TQA is a much more complex dataset with more positive documents, we take an example from TQA.

	NQ				TQA			
Embedding/Metric	**MRR_dev**	**MRR_test**	**MH_dev**	**MH_test**	**MRR_dev**	**MRR_test**	**MH_dev**	**MH_test**
BART-GST	28.4	25.0	**53.2**	48.7	17.5	17.6	39.1	39.5
GCN	28.9	27.7	51.1	**50.0**	**20.0**	**19.4**	41.6	**41.4**
GAT	28.1	27.1	52.3	47.2	19.1	18.9	**43.0**	41.0
GraphSage	**29.8**	26.5	52.3	47.2	19.6	18.4	42.9	39.7
GIN	28.4	**27.8**	50.2	48.5	19.7	18.9	42.2	39.3

Table B.2: Results of G-RAG with difference GNN models. We use Mean Hits @ 10.

Question: Ol' Blue Eyes is the nickname of?

Gold Answer: ['Sinatra (film)', 'Biography of Frank Sinatra', 'Columbus Day Riot', 'Life of Frank Sinatra', 'A Voice in Time: 1939–1952', 'Sinatra', 'Biography of frank sinatra', 'Ol' Blue Eyes', 'A Voice in Time: 1939-1952', 'Political beliefs of frank sinatra', 'Franck Sinatra', 'Old Blue Eyes', 'Frank Sinatra', 'Frank Sinatra I', 'Francis Albert Frank Sinatra', 'Francis A. Sinatra', 'Frank Sinatra, Sr.', 'Francis Albert Sinatra', 'Political beliefs of Frank Sinatra', 'Old blue eyes', 'Frank sanatra', 'Frank sinatra', 'Frank senatra', 'FBI Files on Frank Sinatra', 'Francis Sinatra']

Number of Positive documents: 24 positive documents out of 100 documents

The following are the Top-10 documents given by the proposed GNN-based reranker. Each document is accompanied by relevant information about its AMR graph, including the number of nodes and edges, as well as the count of single-source shortest paths (SSSPs) originating from the node labeled "question". If the node "question" is not present in the AMR graph, the SSSPs count is noted as 0. Additionally, we present the corresponding score assigned by palm 2-L and its rank based on the palm 2 reranker. The ranking assigned by the retriever DPR is also provided for reference. [30].

1st: Sinatra in 1998, for example, the building was bathed in blue light to represent the singer's nickname ”Ol' Blue Eyes”. After actress Fay Wray, who starred in ”King Kong”, died in September 2004, the building lights were extinguished for 15 minutes. The floodlights bathed the building in red, white, and blue for several months after the destruction of the World Trade Center in September 2001, then reverted to the standard schedule. On June 4, 2002, the Empire State Building donned purple and gold (the royal colors of Elizabeth II), in thanks for the United Kingdom playing the Star Spangled Banner

- -

AMR graph information: # nodes: 51, # edges 82, # SSSP: 32

Score by palm 2: 50/100, **Rank by palm 2:** 9/100

Rank by DPR: 5/100

2nd: and actively campaigned for presidents such as Harry S. Truman, John F. Kennedy, and Ronald Reagan. In crime, the FBI investigated Sinatra and his alleged relationship with the Mafia. While Sinatra never learned how to read music, he had an impressive understanding of it, and he worked very hard from a young age to improve his abilities in all aspects of music. A perfectionist, renowned for his dress sense and performing presence, he always insisted on recording live with his band. His bright blue eyes earned him the popular nickname ”Ol' Blue Eyes”. Sinatra led a colorful personal life, and

- -

AMR graph information: # nodes: 53, # edges 75, # SSSP: 34

Score by palm 2: 100/100, **Rank by palm 2:** 1/100

Rank by DPR: 1/100

3rd: claimed that Sinatra had grown ”tired of entertaining people, especially when all they wanted were the same old tunes he had long ago become bored by”. While he was in retirement, President Richard Nixon asked him to perform at a Young Voters Rally in anticipation of the upcoming campaign. Sinatra obliged and chose to sing ”My Kind of Town” for the rally held in Chicago on October 20, 1972. In 1973, Sinatra came out of his short-lived retirement with a television special and album. The album, entitled ”Ol’ Blue Eyes Is Back”, arranged by Gordon Jenkins and Don Costa,

- -

AMR graph information: # nodes: 52, # edges 85, # SSSP: 19

Score by palm 2: 20/100, **Rank by palm 2:** 27/100

Rank by DPR: 8/100

4th: State Police would attend, searching for organized crime members in the audience. During a 1979 appearance in Providence, Mayor Buddy Cianci named Sinatra an honorary fire chief, complete with a helmet bearing the name ”F. SINATRA” with nickname ”Ol’ Blue Eyes” beneath. David Bowie’s concert on May 5, 1978 was one of three recorded for his live album ”Stage”. The Bee Gees performed two sold-out concerts here on August 28–29, 1979 as part of their Spirits Having Flown Tour. The Kinks recorded much of their live album and video, ”One for the Road” at the Civic Center September 23, 1979.

- -

AMR graph information: # nodes: 54, # edges 67, # SSSP: 0

Score by palm 2: 50/100, **Rank by palm 2:** 9/100

Rank by DPR: 6/100

5th: illness). Pasetta was the producer of the Elvis Presley concert special, ”Aloha from Hawaii Via Satellite” in January 1973. The show still holds the record for the most watched television special in history; viewing figures are between 1 and 1.5 billion live viewers worldwide. 1973 also saw Pasetta direct ”Magnavox Presents Frank Sinatra” (also known as ”Ol’ Blue Eyes Is Back”), the television special that marked Frank Sinatra’s comeback from retirement. Pasetta died in a 2015 single-car accident. The vehicle driven by Keith Stewart collided with Pasetta shortly after Stewart had allowed his passengers to disembark. Marty Pasetta Martin Allen

- -

AMR graph information: # nodes: 39, # edges 59, # SSSP: 39

Score by palm 2: 50/100, **Rank by palm 2:** 9/100

Rank by DPR: 3/100

6th: him feel wealthy and important, and that he was giving his very best to the audience. He was also obsessed with cleanliness—while with the Tommy Dorsey band he developed the nickname ”Lady Macbeth”, because of frequent showering and switching his outfits. His deep blue eyes earned him the popular nickname ”Ol’ Blue Eyes”. For Santopietro, Sinatra was the personification of America in the 1950s: ”cocky, eye on the main chance, optimistic, and full of the sense of possibility”. Barbara Sinatra wrote, ”A big part of Frank’s thrill was the sense of danger that he exuded, an underlying, ever-present tension only

- -

AMR graph information: # nodes: 44, # edges 81, # SSSP: 30

Score by palm 2: 100/100, **Rank by palm 2:** 1/100

Rank by DPR: 2/100

7th: where his suite and those of his entourage were on the 23rd floor. His tour, his first in Australia in 15 years and billed as ”Ol’ Blue Eyes Is Back,” was scheduled to include two shows in Melbourne, followed by three in Sydney. In his first show, according to news reports from 1974, Sinatra referred on stage to the media as ”parasites” and ”bums” and to women specifically as ”the broads of the press, the hookers of the press,” then adding, ”I might offer them a buck and a half, I’m not sure.” The character of Rod Blue in the

- -

AMR graph information: # nodes: 32, # edges 61, # SSSP: 32

Score by PaLM 2: 50/100, **Rank by palm 2:** 9/100

Rank by DPR: 11/100

8th: RLPO, BBC Concert Orchestra (for ”Friday Night Is Music Night”), Lahti Symphony Orchestra, Northern Sinfonia, the Melbourne Symphony Orchestra, the Adelaide Symphony Orchestra for the Adelaide Cabaret Festival and the RTÉ Concert Orchestra. His most popular show is the interactive ”Sinatra Jukebox” where, ”instead of an hour of songs and anecdote, halfway through members of the audience were invited to fill in request forms”. Reviewing the show, ”Cabaret Scenes” said, ”I can think of no other singer to better pay homage to Ol’ Blue Eyes on his 100th birthday.” In 2014 he performed on BBC Radio 2 with the BBC

- -

AMR graph information: # nodes: 51, # edges 57, # SSSP: 20

Score by palm 2: 20/100, **Rank by palm 2:** 27/100

Rank by DPR: 14/100

9th: as his tribute to ”The Great American Songbook”. The album has Oleg’s vocals and arrangements by a big band leader Patrick Williams (a late period Frank Sinatra recording associate) and sound engineering by Al Schmitt, whose 60-year career yielded 150 gold and platinum albums, 20 Grammy awards and who also recorded Ol’ Blue Eyes. ”Bring Me Sunshine” was produced at the legendary Capitol Records studios in Hollywood, CA. Songwriter, Charles Strouse quoted: ””The Great American Songbook, to which I am proud to be a contributor, is one of our greatest cultural exports, Oleg is a living example of what an

- -

AMR graph information: # nodes: 52, # edges 68, # SSSP: 12

Score by palm 2: 50/100, **Rank by palm 2:** 9/100

Rank by DPR: 27/100

10th: first place wins The Founding Director is Ben Ferris (2004+). Sydney Film School runs two courses: The Diploma of Screen & Media and The Advanced Diploma of Screen & Media. Some of the accolades afforded to Sydney Film School graduates for their work include: Best Student Documentary Film at Antenna Film Festival: ”Ol’ Blue Eyes”, Matt Cooney Finalist at Bondi Short Film Festival: ”Letters Home”, Neilesh Verma Industry Advisory Board (IAB) Pitch Competition winner: ”Lotus Sonny”, Gary Sofarelli Opening Night screening; Best Australian Animation & Best Australian Composer at World of Women WOW Film Festival, 2012: ”Camera Obscura”, Marta Maia

- -

AMR graph information: # nodes: 58, # edges 69, # SSSP: 35

Score by palm 2: 0/100, **Rank by palm 2:** 30/100

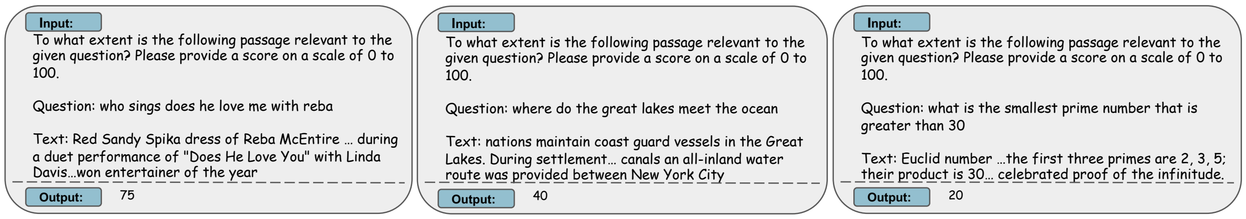
Rank by DPR: 39/100

By analyzing the above result, we note that documents (such as 1st, 2nd, and 4th) containing exact words from the question (i.e., these words are “Ol’ Blue Eyes” and “nickname” in our example) are prioritized at the top by most rankers. However, if a document includes word variations or lacks sufficient keywords, it poses a challenge for the baseline reranker to identify its relevance, see the 9th and 10th documents. To address this issue, the AMR graph of documents is used in our method to comprehend more intricate semantics. The SSSPs from the ‘question’ node in the AMR graph also play the crucial role in uncovering the underlying connections between the question and the words in the documents.

Another challenging scenario for the baseline reranker arises when several keywords or even gold answers are present in the documents but are weakly connected, making recognition difficult. For example, in the 7th, 8th, and 9th documents there are both “Ol’ Blue Eyes” and “Sinatra” which are gold answers, yet these words are not directly linked as the sentence: “Ol’ Blue Eyes is the nickname of “Sinatra”. Instead, the connection between these two words is very loose. Luckily, the 7th, 8th, and 9th documents are connected to the 1st document in the document graph due to common nodes like ’Sinatra’ and ’Ol Blue Eyes.’ The 1st document stands out as more easily identifiable as a positive document, given its incorporation of all keywords from the questions. These words not only have a strong connection but also collectively contribute to a cohesive answer to the question. Leveraging this information and employing a message-passing mechanism, we can enable the 7th, 8th, and 9th document to adeptly discern potential keywords. Consequently, this approach enhances their ranking, based on the insights derived from the well-connected and information-rich 1st document.

B.4 Examples of LLM-generate Relevant Score

Some examples of LLM-generate relevant score are illustrated in Fig B.3.


Input:
To what extent is the following passage relevant to the given question? Please provide a score on a scale of 0 to 100.

Question: who sings does he love me with reba

Text: Red Sandy Spika dress of Reba McEntire ... during a duet performance of "Does He Love You" with Linda Davis...won entertainer of the year

Output: 75

Input:
To what extent is the following passage relevant to the given question? Please provide a score on a scale of 0 to 100.

Question: where do the great lakes meet the ocean

Text: nations maintain coast guard vessels in the Great Lakes. During settlement... canals an all-inland water route was provided between New York City

Output: 40

Input:
To what extent is the following passage relevant to the given question? Please provide a score on a scale of 0 to 100.

Question: what is the smallest prime number that is greater than 30

Text: Euclid number ...the first three primes are 2, 3, 5; their product is 30... celebrated proof of the infinitude.

Output: 20

Figure B.3: Examples of LLM-generate relevant score.

APPENDIX C

Proofs for Chapter 4

C.1 Proof of Lemma 1

Let $\mathcal{P} = \{a_1, a_2, \cdots, a_k\}$ be a set of k independent random vectors in $\mathbb{R}^d$. For $\forall i \in [k]$, $a_i = [a_{i1}, a_{i2}, \cdots, a_{id}]^\top \in \mathbb{R}^d$, we have

$$a_{i\ell} := \begin{cases} a_{i\ell} \sim \mathcal{N}(0, \frac{1}{s}) & \text{with probability } \frac{s}{d}, \\ 0 & \text{otherwise.} \end{cases} \tag{C.1.1}$$

Thus, we have the following properties

$$\begin{aligned} \mathbb{E}[\langle a_i, a_i \rangle] &= 1, \ \forall i \in [k], \\ \mathbb{E}[\langle a_i, a_j \rangle] &= 0, \ \forall i, j \in [k], i \neq j, \\ \mathbb{E}[\|a_i\|_0] &= s, \ \forall i \in [k]. \end{aligned}$$

Based on the above definitions, three steps achieve the proof of Lemma 1:

1. Prove that under certain condition, for any $i, j \in [k]$ with $i \neq j$, with probability at least $1 - \frac{2\delta}{k^2}$, we have $|\langle a_i, a_j \rangle| \leq \varepsilon$. With probability at least $1 - \frac{\delta}{k}$, we have $| \, \|a_i\|_2^2 - 1| \leq \tau$ and $\|a_i\|_0 \leq s + \tau$ for any $i \in [k]$. This is provided in Lemma 10.

2. By a union bound over all the $\binom{k}{2} = k(k-1)/2$ possible pairs of (i, j) mentioned in Step 1, it concludes that for all $i, j \in [k]$ with $i \neq j$, we have $|\langle a_i, a_j \rangle| \leq \varepsilon$ with probability at least

$1-\delta$. We also have $|\,\|a_i\|_2^2 - 1| \le \tau$ and $\|a_i\|_0 \le s+\tau$ for all $i \in [k]$ with probability at least $1-\delta$ by a union bound over all $i \in [k]$.

3. We normalize $\forall\, a_i \in \mathcal{P}$ and get $\tilde{\mathcal{P}} = \{\tilde{a}_1, \tilde{a}_2, \cdots, \tilde{a}_k\}$ where $\|\tilde{a}_i\|_2 = 1$ with $i \in [k]$. From $\|a_i\|_0 \le s+\tau$ and $0 \le \tau < 1$ mentioned in Step 2, we can bound $\|\tilde{a}_i\|_0 \le s$ with $s \in [k]$. Based on Lemma 10 and normalized set $\tilde{\mathcal{P}}$, Theorem 2 presents the condition where the feature matrix $\Phi \in \mathbb{R}^{k\times d}$ in Lemma 1 can be constructed by setting $\mathrm{rows}(\Phi) = (\tilde{a}_i)_{i=1}^k$.

Lemma 10 *Let $0 < \delta < 1$. Consider the set $\mathcal{P} = \{a_1, a_2, \cdots, a_k\}$ described in (C.1.1).*

If $0 < \varepsilon \le \frac{C^2 s}{d}$, by choosing $k \ge \sqrt{\delta}\exp\left(\frac{d\varepsilon^2}{4C^2}\right)$, we have

$$\text{for any } i,j \in [k],\ i \ne j,\ \ |\langle a_i, a_j\rangle| \le \varepsilon \ \text{ with probability at least } 1 - 2\delta/k^2. \tag{C.1.2}$$

If $\varepsilon > \frac{C^2 s}{d}$, by choosing $k \ge \sqrt{\delta}\exp\left(\frac{s\varepsilon}{4}\right)$, we have

$$\text{for any } i,j \in [k],\ i \ne j,\ \ |\langle a_i, a_j\rangle| \le \varepsilon \ \text{ with probability at least } 1 - 2\delta/k^2. \tag{C.1.3}$$

For sufficiently small τ, $0 \le \tau < 1$, by choosing $k \ge \frac{\delta}{2}e^{\tau^2/8}$, we have

$$\text{for any } i \in [k],\ \left|\|a_i\|_2^2 - 1\right| \le \tau \ \text{ with probability at least } 1 - \delta/k. \tag{C.1.4}$$

Moreover, by choosing $k \ge \delta e^{2\tau^2/d}$, we have

$$\text{for any } i \in [k],\ \|a_i\|_0 \le s+\tau \ \text{ with probability at least } 1 - \delta/k. \tag{C.1.5}$$

Proof: *Please refer to Section C.2 for detailed proof.* □

Proposition 2 *Let $0 < \delta < 1$, $0 \le \tau < 1$, $c > 1$ and $C' = \frac{2c^3}{(1+\tau)\sqrt{c^2-1}}$. Consider the normalized set $\tilde{\mathcal{P}} = \{\tilde{a}_1, \tilde{a}_2, \cdots, \tilde{a}_k\}$ derived from $\mathcal{P}$ (C.1.1). For sufficiently small τ, we have*

$$\text{for all } i,j \in [k]\ i \ne j,\ \ |\langle \tilde{a}_i, \tilde{a}_j\rangle| \le \varepsilon,\ \ \|\tilde{a}_i\|_0 \le s \ \text{ with probability at least } 1-\delta, \tag{C.1.6}$$

by choosing $k \geq \sqrt{\delta}\exp\left(\frac{d(1+\tau)\varepsilon^2}{4C'}\right)$ *if* $0 < \varepsilon \leq \frac{C's}{d}$*. If* $\varepsilon > \frac{C's}{d}$*, we choose* $k \geq \sqrt{\delta}\exp\left(\frac{s(1+\tau)\varepsilon}{4}\right)$ *to achieve (C.1.6).*

Therefore, with probability at least $1-\delta$*, the normalized set* $\tilde{\mathcal{P}}$ *satisfies that for all* $i, j \in [k],\ i \neq j,\ \langle \tilde{a}_i, \tilde{a}_j\rangle \leq \varepsilon,\ \|\tilde{a}_i\|_0 \leq s$*. Hence, the feature matrix* $\Phi \in \mathbb{R}^{k\times d}$ *in Lemma 1 can be established by choosing* $\mathrm{rows}(\Phi) = (\tilde{a}_i)_{i=1}^k$ *where* $\tilde{a}_i \in \tilde{\mathcal{P}}$ *when* k *is sufficiently large according to Proposition 2.*

C.2 Proof of Lemma 10

We first introduce some existential definitions and propositions which are helpful to our proof.

Definition 4 *A random variable* X *with mean* $\mu = \mathbb{E}[X]$ *is sub-exponential if there are non-negative parameters* (v, α) *such that*

$$\mathbb{E}\left[e^{\lambda(X-\mu)}\right] \leq e^{\frac{v^2\lambda^2}{2}}, \quad \forall\, |\lambda| < \frac{1}{\alpha}.$$

Proposition 3 (Sub-exponential tail bound) *Assume that* X *is sub-exponential with parameters* (v, α)*. Then*

$$\mathbb{P}[|X-\mu| \geq t] \leq \begin{cases} 2e^{-\frac{t^2}{2v^2}}, & 0 \leq t \leq \frac{v^2}{\alpha}, \\ 2e^{-\frac{t}{2\alpha}}, & t > \frac{v^2}{\alpha}. \end{cases}$$

For $\forall\, a \in \mathcal{P}$, each element of a can be taken as the product of two independent random variables, i.e., one is from the Bernoulli distribution and the other is from the Gaussian distribution. Hence, the individual term, i.e., $a_{i\ell}a_{j\ell}$, of $\langle a_i, a_j\rangle = \sum_{\ell=1}^d a_{i\ell}a_{j\ell}$ with $\forall a_i, a_j \in \mathcal{P},\ i \neq j$ can be represented by a random variable Z_ℓ. Specifically, $Z_\ell = P_\ell X_\ell Q_\ell Y_\ell$ where $\ell \in [d]$ is the product of independent random variables. Herein, P_ℓ and Q_ℓ are independent Bernoulli random variables which take the value 1 with probability s/d and the value 0 with probability $1 - s/d$. X_ℓ and Y_ℓ are independent

Gaussian random variables drawn from $\mathcal{N}(0, 1/s)$. For $|\lambda| < m$, we have

$$\begin{aligned}
\mathbb{E}[e^{\lambda Z_\ell}] &= \sum_{pq\in\{0,1\}} \mathbb{P}[P_\ell Q_\ell = pq] \cdot \frac{s}{2\pi} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{\lambda(pq)xy} \cdot e^{-s(x^2+y^2)/2} dxdy \\
&= \frac{s}{2\pi} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{\lambda xy} \cdot e^{-s(x^2+y^2)/2} dxdy \cdot \left(\frac{s}{d}\right)^2 \\
&\quad + \frac{s}{2\pi} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-s(x^2+y^2)/2} dxdy \cdot \left(1 - \left(\frac{s}{d}\right)^2\right) \\
&\overset{(i)}{\leq} \frac{s}{2\pi} \cdot \frac{2\pi}{\sqrt{s^2-\lambda^2}} \cdot \left(\frac{s}{d}\right)^2 + \frac{s}{2\pi} \cdot \frac{2\pi}{s} \left(1 - \left(\frac{s}{d}\right)^2\right) \\
&\leq \frac{s^3}{d^2\sqrt{s^2-\lambda^2}} + 1 \\
&\overset{(ii)}{=} \frac{c^3\lambda^2}{d^2\sqrt{c^2-1}} + 1 \\
&\overset{(iii)}{\leq} e^{\frac{c^3\lambda^2}{d^2\sqrt{c^2-1}}}
\end{aligned} \tag{C.2.1}$$

where step (i) comes from

$$\begin{aligned}
&\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{\lambda(xy)} e^{-s(x^2+y^2)/2} dxdy \\
=& \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-s(x-\frac{\lambda}{s}y)^2/2} e^{\lambda^2 y^2/(2s)} e^{-sy^2/2} dxdy \\
=& \sqrt{\frac{2\pi}{s}} \int_{-\infty}^{\infty} e^{\lambda^2 y^2/(2s)} e^{-s^2y^2/(2s)} dy \\
=& \sqrt{\frac{2\pi}{s}} \int_{-\infty}^{\infty} e^{-y^2(s^2-\lambda^2)/(2s)} dy \\
=& \frac{2\pi}{\sqrt{s^2-\lambda^2}},
\end{aligned} \tag{C.2.2}$$

step (ii) is derived by choosing $s = c|\lambda|$, $c > 1$, and step (iii) is due to the fact $x + 1 \leq e^x$.

Following (C.2.1) and Definition 4, we find that

$$\mathbb{E}[e^{\lambda Z_\ell}] \leq e^{\frac{c^3\lambda^2}{d^2\sqrt{c^2-1}}} = e^{\frac{v^2\lambda^2}{2}}, \quad \text{for all } |\lambda| < m \text{ and } v^2 = \frac{2c^3}{d^2\sqrt{c^2-1}}, c > 1, \tag{C.2.3}$$

which shows that Z_ℓ is sub-exponential with parameters $(v_\ell, \alpha_\ell) = (C/d, 1/s)$ where $C = \sqrt{\frac{2c^3}{\sqrt{c^2-1}}}$ and $c > 1$. Furthermore, the variable $\sum_{\ell=1}^{d} (Z_\ell - \mathbb{E}[Z_\ell])$ is sub-exponential with the parameters (v_*, α_*), where

$$\alpha_* := \max_{\ell=1,\ldots,n} \alpha_\ell = \frac{1}{s} \quad \text{and} \quad v_* := \sqrt{\sum_{\ell=1}^{d} v_\ell^2}.$$

Based on the fact $\mathbb{E}[Z_\ell] = 0$, the tail bound can be derived from Proposition 3,

$$\mathbb{P}\left[\left|\sum_{\ell=1}^{d} Z_\ell\right| \geq t\right] \leq \begin{cases} 2e^{-\frac{t^2}{2v_*^2}}, & 0 \leq t \leq \frac{v_*^2}{\alpha_*}, \\ 2e^{-\frac{t}{2\alpha_*}}, & t > \frac{v_*^2}{\alpha_*}. \end{cases} \tag{C.2.4}$$

Thus, we have for two vectors $a_i, a_j \in \mathcal{P}$ and $i \neq j$,

$$\mathbb{P}\left[|\langle a_i, a_j\rangle| \geq t\right] \leq \begin{cases} 2e^{-\frac{dt^2}{2C^2}}, & 0 \leq t \leq \frac{C^2 s}{d}, \\ 2e^{-\frac{mt}{2}}, & t > \frac{C^2 s}{d}, \end{cases} \tag{C.2.5}$$

where $C = \sqrt{\frac{2c^3}{\sqrt{c^2-1}}}$ and $c > 1$. By setting $2e^{-\frac{dt^2}{2C^2}} = 2\delta/k^2$, we have $t = \sqrt{\frac{2C^2}{d}\log(\frac{k^2}{\delta})}$. We choose $k \geq \sqrt{\delta}\exp\left(\frac{d\varepsilon^2}{4C^2}\right)$ such that $t \geq \varepsilon$. Hence, we conclude $\mathbb{P}\left[|\langle a_i, a_j\rangle| \geq \varepsilon\right] \leq 2\delta/k^2$, which implies the statement (C.1.2) when $0 < \varepsilon \leq \frac{C^2 s}{d}$ in Lemma 10. Similar arguments can be applied to the proof of the statement (C.1.3) when $\varepsilon > \frac{C^2 s}{d}$ in Lemma 10. The proof of the statement (C.1.4) can also be completed by following similar but simpler arguments of proving the statement (C.1.2) and (C.1.3).

We are left to the proof of statement (C.1.5). For $\forall\ a \in \mathcal{P}$, the random variable $\|a\|_0$ obeys the binomial distribution with parameters d and s/d, i.e., $\mathcal{B}(d, s/d)$. It is the discrete probability distribution of the number of d independent Bernoulli trials which return Boolean-valued outcome: the ℓ-th ($\ell \in [d]$) element of a is non-zero (with probability s/d) or zero (with probability $1 - s/d$).

According to the book by Ross [187], we first introduce several properties of the binomial distribution. The cumulative distribution function of binomial distribution $\mathcal{B}(n, p)$ can be represented by

$$\mathbb{F}(k; n, p) = \mathbb{P}[X \leq k] = \sum_{i=0}^{\lfloor k \rfloor} \binom{n}{i} p^i (1-p)^{n-i},$$

where we also have $\mathbb{F}(n-k; n, 1-p) = 1 - \mathbb{F}(k; n, p)$. Based on Hoeffding's inequality, $F(k; n, p)$

can be bounded by

$$\mathbb{F}(k;n,p) \leq \exp\left(-2n\left(p-\frac{k}{n}\right)^2\right).$$

Hence, the upper tail bound for the random variable $\|a\|_0$ is given by

$$\mathbb{P}[\|a\|_0 \geq m+\tau] = \mathbb{F}(d-s-\tau; d, 1-\frac{s}{d}) \leq \exp\left(\frac{-2\tau^2}{d}\right), \tag{C.2.6}$$

where $0 \leq \tau < 1$. By choosing $k \geq \delta \exp(2\tau^2/d)$, it yields $\mathbb{P}[\|a\|_0 \geq s+\tau] \leq \frac{\delta}{k} \leq \exp\left(\frac{-2\tau^2}{d}\right)$. Thus, we completed the proof of statement (C.1.5).

C.3 $\mathrm{poly}(s)$-Query Algorithm for s-sparsity Case with Noise

All results above focus on the noiseless case. We briefly discuss the noisy cases. Consider the stochastic misspecified sparse linear bandits where a feature matrix $\Phi \in \mathbb{R}^{k\times d}$, $x_t \in \mathrm{rows}(\Phi)$, and the reward

$$r_{x_t} = \langle x_t, \theta^* \rangle + \nu_{x_t} + \eta_t \tag{C.3.1}$$

where $\nu_{x_t} \in [-\varepsilon, \varepsilon]$ and $\{\eta_t\}$ is a sequence of independent 1-subgaussian random variables.

Based on the reward function (C.3.1) and the notation in Algorithm 4, we start with the approximation error of $f(\theta^*)$:

$$\begin{aligned}
&|\langle f(a), \hat{\theta}_f\rangle - \langle a, \theta^*\rangle| \\
\leq& |\langle f(a), \hat{\theta}_f\rangle - \langle f(a), f(\theta^*)\rangle| + |\langle f(a), f(\theta^*)\rangle - \langle a, \theta^*\rangle|, \\
\leq& \left| f(a)^\top G(\rho)^{-1} \sum_{b_t\in\mathcal{S}} \rho(b_t)\nu_{b_t} b_t + f(a)^\top G(\rho)^{-1} \sum_{b_t\in\mathcal{S}} \rho(b_t) b_t \eta_t \right| + 2\upsilon \\
\leq& \left| f(a)^\top G(\rho)^{-1} \sum_{b_t\in\mathcal{S}} \rho(b_t)\nu_{b_t} b_t \right| + \left| f(a)^\top G(\rho)^{-1} \sum_{b_t\in\mathcal{S}} \rho(b_t) b_t \eta_t \right| + 2\upsilon
\end{aligned} \tag{C.3.2}$$

for $\forall\ a \in \mathrm{rows}(\Phi)$.

The first term in (C.3.2) can be bounded as

$$\begin{aligned}\left|f(a)^\top G(\rho)^{-1}\sum_{b_t\in\mathcal{S}}\rho(b_t)\nu_{b_t}b_t\right| &\leq \varepsilon\sum_{b_t\in\mathcal{S}}\rho(b_t)\left|f(a)^\top G(\rho)^{-1}b_t\right| \\ \leq&\varepsilon\sqrt{\left(\sum_{b_t\in\mathcal{S}}\rho(b_t)\right)f(a)^\top\sum_{b_t\in\mathcal{S}}\rho(b_t)G(\rho)^{-1}b_tb_t^\top G(\rho)^{-1}f(a)} \\ =&\varepsilon\sqrt{\sum_{b_t\in\mathcal{S}}\rho(b_t)\left\|f(a)\right\|_{G(\rho)^{-1}}^2} \\ \leq&2\varepsilon\sqrt{p}\,, \end{aligned} \tag{C.3.3}$$

where is derived from Jensen's inequality and the fact that $\|f(a)\|_{G^{-1}}^2 \leq 2p/t$ for t-th time step in Algorithm 4. The second term in (C.3.2) can be bounded by standard concentration bounds: with probability at least $1-2/(kn)$,

$$\begin{aligned}\left|f(a)^\top G(\rho)^{-1}\sum_{b_t\in\mathcal{S}}\rho(b_t)b_t\eta_t\right| &\leq \|f(a)\|_{G^{-1}}\sqrt{2\log{(kn)}} \\ &\leq \sqrt{\frac{4p}{t}\log{(kn)}}. \end{aligned} \tag{C.3.4}$$

Combining (C.3.2), (C.3.3), (C.3.4), we have

$$|\langle f(a),\hat{\theta}_f\rangle - \langle a,\theta^*\rangle| \leq 2\varepsilon\sqrt{p}+\sqrt{\frac{4p}{t}\log{(kn)}}+2\upsilon. \tag{C.3.5}$$

Similarly to the analysis in Section 4.6, we can derive the final approximate error as

$$\begin{aligned}&|\langle f(a),\hat{\theta}_f\rangle - \langle a,\theta^*\rangle| \\ \leq&C\left((\log(k))^{\frac{1}{4}}\sqrt{\varepsilon}+\sqrt{\frac{p}{t}\log{(kn)}}\right). \end{aligned} \tag{C.3.6}$$

Based on (C.3.6), the active action set in Algorithm 4 in the noise case should be

$$\mathcal{S}\leftarrow\left\{a\in\mathcal{S}:\max_{b\in\mathcal{S}}\langle\hat{\theta}_f,b-a\rangle\leq C\left((\log(k))^{\frac{1}{4}}\sqrt{\varepsilon}+\sqrt{\frac{p}{t}\log{(kn)}}\right)\right\}.$$

APPENDIX D

Proofs for Chapter 5

D.1 Proof of Lemma 2

We begin with the following definition:

Definition 5 *A random variable X is σ-subgaussian if for all $\tau \in \mathbb{R}$, it holds that* $\mathbb{E}[\exp(\tau X)] \leq \exp(\tau^2\sigma^2/2)$.

Based on the definitions of $\boldsymbol{\theta}^\natural$ and $\hat{\boldsymbol{\theta}}_k$ in Section 5.7.1, for any $0 \leq j \leq k-1$ and $t_j \leq i \leq t_{j+1}-1$ with the action pending sequence $(a_{i-m}, \cdots, a_{i-1})$, since $0 \leq V_m^k(s, \vec{\boldsymbol{a}}) \leq 1/(1-\gamma)$ for any s, then $\{V_m^k(s_{i+1}, \vec{\boldsymbol{a}}_{i+1}) - \langle \boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}), \boldsymbol{\theta}^\natural \rangle\}$ is a sequence of $1/(1-\gamma)$-subgaussian random variables. Define $Z_{i,m}$ as

$$Z_{i,m} = V_m^k(s_{i+1}, \vec{\boldsymbol{a}}_{i+1}) - \langle \boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}), \boldsymbol{\theta}^\natural \rangle, \tag{D.1.1}$$

where $0 \leq j \leq k-1$ and $t_j \leq i \leq t_{j+1}-1$. Thus $\{Z_{i,m}\}$ is a sequence of $1/(1-\gamma)$-subgaussian random variables.

Based on the definition of $Z_{i,m}$ (D.1.1), we thus have

$$\begin{aligned}
&\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1}\Bigg(\big(V_m^k(s_{i+1},\vec{\boldsymbol{a}}_{i+1})-\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\theta}\rangle\big)^2\\
&\quad-\big(V_m^k(s_{i+1},\vec{\boldsymbol{a}}_{i+1})-\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\theta}^\natural\rangle\big)^2\Bigg)\\
=&\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1}\Bigg(\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\theta}^\natural\rangle-\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\theta}\rangle\Bigg)^2\\
&\quad+2Z_{i,m}\Bigg(\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\theta}^\natural\rangle\\
&\quad\quad-\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\theta}\rangle\Bigg)
\end{aligned}\tag{D.1.2}$$

By reformulating the first term in (D.1.2), we arrive

$$\begin{aligned}
&\frac{1}{2}\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1}\big(\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\theta}^\natural\rangle-\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\theta}\rangle\big)^2\\
=&\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1}\big(V_m^k(s_{i+1},\vec{\boldsymbol{a}}_{i+1})-\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\theta}\rangle\big)^2\\
&\quad-\big(V_m^k(s_{i+1},\vec{\boldsymbol{a}}_{i+1})-\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\theta}^\natural\rangle\big)^2+W_{i,m}(\boldsymbol{\theta})
\end{aligned}\tag{D.1.3}$$

where

$$\begin{aligned}
W_{i,m}(\boldsymbol{\theta})=&-\frac{1}{2}\big(\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\theta}^\natural\rangle-\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\theta}\rangle\big)^2\\
&\quad+2Z_{i,m}\big(\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\theta}\rangle-\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\theta}^\natural\rangle\big)\ .
\end{aligned}\tag{D.1.4}$$

Recall that $\hat{\boldsymbol{\theta}}_k$ (5.4). Substituting $\hat{\boldsymbol{\theta}}_k$ into (D.1.3), we have

$$\frac{1}{2}\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1}\Big(\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\theta}^\natural\rangle-\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\hat{\boldsymbol{\theta}}_k\rangle\Big)^2\le\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1}W_{i,m}(\hat{\boldsymbol{\theta}}_k),\tag{D.1.5}$$

where the inequality holds due to

$$\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1}\Big(V_m^k(s_{i+1},\vec{\boldsymbol{a}}_{i+1})-\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\hat{\boldsymbol{\theta}}_k\rangle\Big)^2$$
$$\le\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1}\big(V_m^k(s_{i+1},\vec{\boldsymbol{a}}_{i+1})-\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\theta}^\natural\rangle\big)^2. \tag{D.1.6}$$

We move to bound $\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1}W_{i,m}(\hat{\boldsymbol{\theta}}_k)$ in (D.1.5). Let $\mathcal{G}:=\{\boldsymbol{\theta}\in\mathbb{R}^d|\|\boldsymbol{\theta}\|_2=\sqrt{d}\}$. We have

$$\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1}W_{i,m}(\hat{\boldsymbol{\theta}}_k)=\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1}W_{i,m}(\hat{\boldsymbol{\theta}}_k)-W_{i,m}(\boldsymbol{\rho})+W_{i,m}(\boldsymbol{\rho})$$
$$\le\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1}W_{i,m}(\hat{\boldsymbol{\theta}}_k)-W_{i,m}(\boldsymbol{\rho})+\max_{\tilde{\boldsymbol{\rho}}\in\mathcal{G}}W_{i,m}(\tilde{\boldsymbol{\rho}}) \tag{D.1.7}$$

We can bound two terms in (D.1.7) separately.

- For the first term in (D.1.7), we have

$$\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1}W_{i,m}(\hat{\boldsymbol{\theta}}_k)-W_{i,m}(\boldsymbol{\rho})$$
$$\overset{(a)}{=}\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1}\frac{1}{2}\big(\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\rho}\rangle-\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\theta}^\natural\rangle\big)^2$$
$$-\frac{1}{2}\Big(\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\hat{\boldsymbol{\theta}}_k\rangle-\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\theta}^\natural\rangle\Big)^2$$
$$+2Z_{i,m}\Big(\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\hat{\boldsymbol{\theta}}_k\rangle-\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\rho}\rangle\Big)$$
$$\overset{(b)}{\le}\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1}\frac{1}{2}\Big(\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\rho}\rangle-\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\hat{\boldsymbol{\theta}}_k\rangle\Big)\cdot$$
$$\Big(2\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\theta}^\natural\rangle+\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\rho}\rangle+\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\hat{\boldsymbol{\theta}}_k\rangle\Big)$$
$$+2|Z_{i,m}|\cdot|\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\hat{\boldsymbol{\theta}}_k\rangle-\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\rho}\rangle|, \tag{D.1.8}$$

where the equality (a) holds due to the definition of $Z_{i,m}$ (D.1.1), and the inequality (b) is derived based on Cauchy-Schwartz inequality. Furthermore, for any $\boldsymbol{\theta}$ and bounded function

V in Definition 2 such that $\|\boldsymbol{\theta}\|_2 \le \sqrt{d}$ and $\|\boldsymbol{\phi}_V(s,a)\|_2 \le \sqrt{d}$, we have

$$\sqrt{\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1} \left|\langle \boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}), \boldsymbol{\theta}\rangle\right|^2} \le d\sqrt{t_k}, \tag{D.1.9}$$

where the inequality holds via Cauchy-Schwartz inequality. Hence, we can bound the first term in (D.1.8) with $2dt_k$ via Cauchy-Schwartz inequality. By reformulating (D.1.8), we arrive

$$\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1} W_{i,m}(\hat{\boldsymbol{\theta}}_k) - W_{i,m}(\boldsymbol{\rho}) \le 2dt_k + 2\sqrt{\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1} |Z_{i,m}|^2} \cdot \sqrt{t_k}, \tag{D.1.10}$$

where the inequality holds due to the selection of $\boldsymbol{\rho}$

$$\begin{aligned}\boldsymbol{\rho} = \arg\min_{\tilde{\boldsymbol{\rho}}\in\Theta} \Bigg(\max_{j=0,\cdots,k-1} \max_{i=t_j,\cdots,t_{j+1}-1} \Big| &\langle \boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}), \hat{\boldsymbol{\theta}}_k\rangle \\ &- \langle \boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}), \tilde{\boldsymbol{\rho}}\rangle \Big| \Bigg)\end{aligned} \tag{D.1.11}$$

such that

$$\sqrt{\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1} \left|\langle \boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}), \hat{\boldsymbol{\theta}}_k\rangle - \langle \boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}), \boldsymbol{\rho}\rangle\right|^2} \le \sqrt{t_k}. \tag{D.1.12}$$

It remains to bound $\sqrt{\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1} |Z_{i,m}|^2}$. Recall the definition of the sequence of $1/(1-\gamma)$-subgaussian random variables, i.e., $Z_{i,m}$ (D.1.1), we have

$$\mathbb{P}\left(Z_{i,m} \ge \frac{1}{1-\gamma}\sqrt{2\log(\delta^{-1})}\right) \le \delta, \tag{D.1.13}$$

Note that for $V_m^\pi(s)$ (5.1) where $\sum_{i=0}^{m-1}\gamma^i r(s_{t+i}, a_{t-m+i}) < m$, $\{r(s_{t+i}, a_{t-m+i})\}_{i=0}^{m-1}$ depends on both the deterministic action pending sequence $\vec{\boldsymbol{a}}_t = (a_{t-m}, \cdots, a_{t-1})$ and next observation states instead of depending on statistical state-action pairs $\{(s_{t+i}, \pi(s_{t-m+i}))\}_{i=0}^{m-1}$, thereby may yielding extra difference between $V_m^k(s_{i+1}, \vec{\boldsymbol{a}}_{i+1})$ and $\langle \boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}), \boldsymbol{\theta}^\natural\rangle$. Therefore, with probability exceeding $1-\delta$, we have

$$\sqrt{\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1} |Z_{i,m}|^2} \le \sqrt{\frac{2}{(1-\gamma)^2} t_k \log(2t_k(t_k+1)\delta^{-1}) + m^2}, \tag{D.1.14}$$

which derives from the union bound and the inequality (D.1.13). Recall the selection of $\boldsymbol{\rho}$ (D.1.11), that with probability $1-\delta$, we have

$$\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1} W_{i,m}(\hat{\boldsymbol{\theta}}_k) - W_{i,m}(\boldsymbol{\rho}) \le 2dt_k + 2\sqrt{\frac{2}{(1-\gamma)^2} t_k^2 \log(2t_k(t_k+1)\delta^{-1}) + m^2 t_k}. \tag{D.1.15}$$

- We continue to bound the second term in (D.1.7). Given $\boldsymbol{\theta}$ satisfying the conditions in Definition 2, by Definition 5, we can conclude that

$$\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1} 2Z_{i,m}\left(\langle \boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}), \boldsymbol{\theta}\rangle - \langle \boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}), \boldsymbol{\theta}^\natural\rangle\right) \tag{D.1.16}$$

is ζ-subgaussian where $\zeta = 2|\langle \boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}), \boldsymbol{\theta}\rangle - \langle \boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}), \boldsymbol{\theta}^\natural\rangle|/(1-\gamma)$. Therefore, for any $0 \le j \le k-1$ and $t_j \le i \le t_{j+1}-1$, given the action sequence $\vec{\boldsymbol{a}}_i = (a_{i-m}, \cdots, a_{i-1})$, with probability at least $1-\delta$, we have

$$\begin{aligned}
&\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1} W_{i,m}(\boldsymbol{\theta}) \\
\le &\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1} -\frac{1}{2}\left(\langle \boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}), \boldsymbol{\theta}^\natural\rangle - \langle \boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}), \boldsymbol{\theta}\rangle\right)^2 \\
&\quad + \tau\,\frac{2|\langle \boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}), \boldsymbol{\theta}\rangle - \langle \boldsymbol{\phi}_{V_m^k}(s_i, a_{i-m}, \vec{\boldsymbol{a}}_{i+1}), \boldsymbol{\theta}^\natural\rangle|^2}{(1-\gamma)^2} + \frac{1}{\tau}\log\left(\frac{1}{\delta}\right) \\
\overset{(a)}{=} &\frac{4}{(1-\gamma)^2}\log\left(\delta^{-1}\right),
\end{aligned} \tag{D.1.17}$$

where the equality (a) holds by selecting $\tau = (1-\gamma)^2/4$.

Thus, based on the definition of $\mathcal{G} := \{\boldsymbol{\theta} \in \mathbb{R}^d | \|\boldsymbol{\theta}\|_2 = \sqrt{d}\}$ and the union bound, the second term in (D.1.7) can be bounded by

$$\max_{\tilde{\boldsymbol{\rho}} \in \mathcal{G}} W_{i,m}(\tilde{\boldsymbol{\rho}}) \le \frac{4}{(1-\gamma)^2}\log\left(d\delta^{-1}\right), \tag{D.1.18}$$

with probability exceeding $1-\delta$.

Given (D.1.5), (D.1.7) and (D.1.15), by using the union bound, we have

$$\sum_{j=0}^{k-1}\sum_{i=t_j}^{t_{j+1}-1}\Big(\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\boldsymbol{\theta}^\natural\rangle-\langle\boldsymbol{\phi}_{V_m^k}(s_i,a_{i-m},\vec{\boldsymbol{a}}_{i+1}),\hat{\boldsymbol{\theta}}_k\rangle\Big)^2+\lambda\|\boldsymbol{\theta}^\natural-\hat{\boldsymbol{\theta}}_k\|_2^2$$

$$\le\frac{8}{(1-\gamma)^2}\log\left(d\delta^{-1}\right)+4\sqrt{\frac{2}{(1-\gamma)^2}t_k^2\log(2t_k(t_k+1)\delta^{-1})+m^2t_k}+4dt_k+\lambda d, \tag{D.1.19}$$

with probability exceeding $1-\delta$, for any $0\le j\le k-1$ and $t_j\le i\le t_{j+1}-1$.

D.2 Proof of Lemma 3

We use induction to prove this lemma. We only need to prove that for all $0\le t\le T$, $Q_m^k\ge Q_m^*$. We have

$$\frac{1}{1-\gamma}=Q_m^0(s,a,\vec{\boldsymbol{a}}_1)\ge Q_m^*(s,a,\vec{\boldsymbol{a}}_1),$$

where the inequality holds due to the fact that $Q^*(s,a)\le 1/(1-\gamma)$ caused by $0\le r(s,a)\le 1$. Assume that the statement holds for $t-1$ with the action sequence $\vec{\boldsymbol{a}}_{t-1}=(a_{t-1-m},\cdots,a_{t-2})$, then $Q_m^k(s,a,\vec{\boldsymbol{a}}_{t-1})\ge Q_m^*(s,a,\vec{\boldsymbol{a}}_{t-1})$, which leads to

$$V_m(s)=\max_{a\in\mathcal{A}}Q_m^k(s,a,\vec{\boldsymbol{a}}_{t-1})\ge\max_{a\in\mathcal{A}}Q_m^*(s,a,\vec{\boldsymbol{a}}_{t-1})=V_m^*(s). \tag{D.2.1}$$

The action sequence at t becomes $\vec{\boldsymbol{a}}_t=(a_{t-m},\cdots,a_{t-1})$ where $a_{t-1}=\operatorname{argmax}_{a\in\mathcal{A}}Q_m^k(\cdot,a,\vec{\boldsymbol{a}}_{t-1})$. Furthermore, we have

$$\begin{aligned}
&Q_m^k(s,a,\vec{\boldsymbol{a}}_t)-r(s,a_{t-m})\\
&=\gamma\max_{\boldsymbol{\theta}\in\mathcal{D}_k}\sum_{(s_{t+1},\cdots,s_{t+m},s')}\langle\boldsymbol{\phi}(s_{t+1}|s_t,a_{t-m}),\boldsymbol{\theta}\rangle\cdots\langle\boldsymbol{\phi}(s'|s_{t+m},a),\boldsymbol{\theta}\rangle V_m(s')\\
&\ge\gamma\sum_{(s_{t+1},\cdots,s_{t+m},s')}\langle\boldsymbol{\phi}(s_{t+1}|s_t,a_{t-m}),\boldsymbol{\theta}^\natural\rangle\cdots\langle\boldsymbol{\phi}(s'|s_{t+m},a),\boldsymbol{\theta}^\natural\rangle V_m(s')\\
&=\gamma[\mathbb{P}V_m^\pi](s,a_{t-m},\vec{\boldsymbol{a}}_{t+1}),
\end{aligned} \tag{D.2.2}$$

where the action sequence is $\vec{\boldsymbol{a}}_{t+1} = (a_{t-m+1}, \cdots, a_{t-1}, a)$. Additionally, we can bound $Q_m^k(s, a, \vec{\boldsymbol{a}}_t)$ by

$$\begin{aligned}Q_m^k(s, a, \vec{\boldsymbol{a}}_t) &= r(s, a_{t-m}) + \gamma \sum_{(s_{t+1}, \cdots, s_{t+m}, s')} \langle \boldsymbol{\phi}(s_{t+1}|s_t, a_{t-m}), \boldsymbol{\theta}^\natural \rangle \cdots \langle \boldsymbol{\phi}(s'|s_{t+m}, a), \boldsymbol{\theta}^\natural \rangle V_m(s') \\ &\leq 1 + \frac{\gamma}{1-\gamma} = \frac{1}{1-\gamma},\end{aligned}$$

where the inequality satisfied since $V_m(s) \leq 1/(1-\gamma)$. We thus arrive at that

$$\begin{aligned}&Q_m^k(s, a, \vec{\boldsymbol{a}}_t) \\ &\overset{(a)}{\geq} r(s, a_{t-m}) + \gamma \sum_{(s_{t+1}, \cdots, s_{t+m}, s')} \langle \boldsymbol{\phi}(s_{t+1}|s_t, a_{t-m}), \boldsymbol{\theta}^\natural \rangle \cdots \langle \boldsymbol{\phi}(s'|s_{t+m}, a), \boldsymbol{\theta}^\natural \rangle V_m(s') \\ &\overset{(b)}{\geq} r(s, a_{t-m}) + \gamma \sum_{(s_{t+1}, \cdots, s_{t+m}, s')} \langle \boldsymbol{\phi}(s_{t+1}|s_t, a_{t-m}), \boldsymbol{\theta}^\natural \rangle \cdots \langle \boldsymbol{\phi}(s'|s_{t+m}, a), \boldsymbol{\theta}^\natural \rangle V_m^*(s') \\ &= Q_m^*(s, a, \vec{\boldsymbol{a}}_t),\end{aligned}$$

where the inequality (a) derives based on (D.2.2), and the inequality (b) derives based on (D.2.1). Therefore, for the action sequence $\vec{\boldsymbol{a}}_t = (a_{t-m}, \cdots, a_{t-1})$, the statement in Lemma 3, i.e.,

$$Q_m^k(s, a, \vec{\boldsymbol{a}}_t) \geq Q_m^*(s, a, \vec{\boldsymbol{a}}_t)$$

can be satisfied for all time step t. The proof is thus completed.

D.3 Proof of Theorem 8

To prove Theorem 8, we begin with the following lemma.

Lemma 11 *Suppose the conditions in Lemma 2 are satisfied. Consider the action sequence $\vec{\boldsymbol{a}}_t = (a_{t-m}, \cdots, a_{t-1})$, for any $0 \leq k \leq N_T - 1$ and $t_k \leq t \leq t_{k+1} - 1$, there exists a $\boldsymbol{\theta}_t \in \mathcal{D}_k$ such that*

$$Q_m^k(s_t, a_t, \vec{\boldsymbol{a}}_t) \leq r(s_t, a_{t-m}) + \gamma \big\langle \boldsymbol{\theta}_t, \boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) \big\rangle + m\gamma^{t-t_k-m+1}. \tag{D.3.1}$$

Proof: *The proof is provided in Appendix D.5.* □

Next, we prove Theorem 8.

Denote $N_T - 1$ as the number of epochs when Algorithm 7 occupied at T-th time step, thus there is $t_{N_T} = T + 1$. Given the action sequence $\vec{\boldsymbol{a}}_t = (a_{t-m}, \cdots, a_{t-1})$, we have

$$\begin{aligned}\text{Regret}(T) = \sum_{k=0}^{N_T-1} \sum_{t=t_k}^{t_{k+1}-1} \left[V_m^*(s_t, \vec{\boldsymbol{a}}_t) - V_m^\pi(s_t, \vec{\boldsymbol{a}}_t)\right] &\overset{(a)}{\leq} \sum_{k=0}^{N_T-1} \sum_{t=t_k}^{t_{k+1}-1} \left[V_m^k(s_t, \vec{\boldsymbol{a}}_t) - V_m^\pi(s_t, \vec{\boldsymbol{a}}_t)\right] \\ &\overset{(b)}{\leq} \sum_{k=0}^{N_T-1} \sum_{t=t_k}^{t_{k+1}-1} Q_m^k(s_t, a_t, \vec{\boldsymbol{a}}_t) - V_m^\pi(s_t, \vec{\boldsymbol{a}}_t)\end{aligned} \tag{D.3.2}$$

where the inequality (a) can be derived under the conditions of Lemma 3, and the equality (b) holds due to the update rule $V_m^k(\cdot, \vec{\boldsymbol{a}}_{t+1}) \leftarrow \max_{a\in\mathcal{A}} Q_m^k(\cdot, a, \vec{\boldsymbol{a}}_t)$ and the reward value is no less that zero.

Implied By Lemma 11, for $t_k \leq t \leq t_{k+1} - 1$, $Q_m^k(s_t, a_t, \vec{\boldsymbol{a}}_t)$ calculated on Line 7 of Algorithm 7 holds that

$$\begin{aligned}Q_m^k(s_t, a_t, \vec{\boldsymbol{a}}_t) &\leq r(s_t, a_{t-m}) + \gamma\left\langle \boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}), \boldsymbol{\theta}_t\right\rangle + m\gamma^{t-t_k-m+1} \\ &\leq r(s_t, a_{t-m}) + \gamma\left\langle \boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}), \boldsymbol{\theta}_t\right\rangle + m\gamma^{t_{k+1}-t_k-m} \\ &\leq r(s_t, a_{t-m}) + \gamma\left\langle \boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}), \boldsymbol{\theta}_t\right\rangle + m\gamma^{\alpha-m},\end{aligned} \tag{D.3.3}$$

where α denotes $\max_k t_{k+1} - t_k$. By selecting $\alpha = \left\lceil \frac{\log((mT)/(1-\gamma))}{1-\gamma} \right\rceil + m$, (D.3.3) can be reformulated as

$$Q_m^k(s_t, a_t, \vec{\boldsymbol{a}}_t) \leq r(s_t, a_{t-m}) + \gamma\left\langle \boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}), \boldsymbol{\theta}_t\right\rangle + \frac{1-\gamma}{T}. \tag{D.3.4}$$

Given the action sequence $\vec{\boldsymbol{a}}_t = (a_{t-m}, \cdots, a_{t-1})$, it yields from the Bellman equation, given by

$$\begin{aligned}V_m^\pi(s_t, \vec{\boldsymbol{a}}_t) &= r(s_t, a_{t-m}) + \gamma[\mathbb{P}V_m^\pi](s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) \\ &\overset{(a)}{=} r(s_t, a_{t-m}) + \gamma \sum_{s'\in\mathcal{S}} \left\langle \boldsymbol{\phi}(s'|s_t, a_{t-m}), \boldsymbol{\theta}^\natural\right\rangle V_m^\pi(s', \vec{\boldsymbol{a}}_{t+1}) \\ &\overset{(b)}{=} r(s_t, a_{t-m}) + \gamma\left\langle \boldsymbol{\phi}_{V_m^\pi}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}), \boldsymbol{\theta}^\natural\right\rangle,\end{aligned} \tag{D.3.5}$$

where the equality (a) and equality (b) are satisfied due to the assumption on the parameterized-CDMDP, i.e., Definition 2. Substituting (D.3.3) and (D.3.5) into (D.3.2), it yields

$$\begin{aligned}
&\sum_{k=0}^{N_T-1}\sum_{t=t_k}^{t_{k+1}-1}\big[V_m^k(s_t,\vec{\boldsymbol{a}}_t)-V_m^\pi(s_t,\vec{\boldsymbol{a}}_t)\big]\\
\le&\sum_{k=0}^{N_T-1}\sum_{t=t_k}^{t_{k+1}-1}\Big(\gamma\big\langle\boldsymbol{\phi}_{V_m^k}(s_t,a_{t-m},\vec{\boldsymbol{a}}_{t+1}),\boldsymbol{\theta}_t\big\rangle-\gamma\big\langle\boldsymbol{\phi}_{V_m^\pi}(s_t,a_{t-m},\vec{\boldsymbol{a}}_{t+1}),\boldsymbol{\theta}^\natural\big\rangle+\frac{1-\gamma}{T}\Big)\\
=&\gamma\sum_{k=0}^{N_T-1}\sum_{t=t_k}^{t_{k+1}-1}\Big(\big\langle\boldsymbol{\phi}_{V_m^k}(s_t,a_{t-m},\vec{\boldsymbol{a}}_{t+1}),\boldsymbol{\theta}_t\big\rangle-\big\langle\boldsymbol{\phi}_{V_m^k}(s_t,a_{t-m},\vec{\boldsymbol{a}}_{t+1}),\boldsymbol{\theta}^\natural\big\rangle\Big)\\
&\quad+\gamma\sum_{k=0}^{N_T-1}\sum_{t=t_k}^{t_{k+1}-1}\big\langle\boldsymbol{\phi}_{V_m^k}(s_t,a_{t-m},\vec{\boldsymbol{a}}_{t+1})-\boldsymbol{\phi}_{V_m^\pi}(s_t,a_{t-m},\vec{\boldsymbol{a}}_{t+1}),\boldsymbol{\theta}^\natural\big\rangle+(1-\gamma)\\
=&J_1+J_2+J_3+(1-\gamma),
\end{aligned}\tag{D.3.6}$$

where

$$\begin{aligned}
J_1&=\gamma\sum_{k=0}^{N_T-1}\sum_{t=t_k}^{t_{k+1}-1}\Big(\big\langle\boldsymbol{\phi}_{V_m^k}(s_t,a_{t-m},\vec{\boldsymbol{a}}_{t+1}),\boldsymbol{\theta}_t\big\rangle-\big\langle\boldsymbol{\phi}_{V_m^k}(s_t,a_{t-m},\vec{\boldsymbol{a}}_{t+1}),\boldsymbol{\theta}^\natural\big\rangle\Big),\\
J_2&=\gamma\sum_{k=0}^{N_T-1}\sum_{t=t_k}^{t_{k+1}-1}\Big\{\big[\mathbb{P}(V_m^k-V_m^\pi)\big](s_t,a_{t-m},\vec{\boldsymbol{a}}_{t+1})-\big(V_m^k(s_{t+1},\vec{\boldsymbol{a}}_{t+1})-V_m^\pi(s_{t+1},\vec{\boldsymbol{a}}_{t+1})\big)\Big\},\\
J_3&=\gamma\sum_{k=0}^{N_T-1}\sum_{t=t_k}^{t_{k+1}-1}\big(V_m^k(s_{t+1},\vec{\boldsymbol{a}}_{t+1})-V_m^\pi(s_{t+1},\vec{\boldsymbol{a}}_{t+1})\big).
\end{aligned}$$

The three terms J_1, J_2, and J_3 can be bounded separately, which is presented in the following.

- In terms of J_1, we have

$$
\begin{aligned}
J_1 &\overset{(a)}{\leq} \gamma \sum_{k=0}^{N_T-1} \sum_{t=t_k}^{t_{k+1}-1} \left| \left\langle \boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}), \boldsymbol{\theta}_t - \boldsymbol{\theta}^\natural \right\rangle \right| \\
&\overset{(b)}{\leq} \sum_{k=0}^{N_T-1} \sum_{t=t_k}^{t_{k+1}-1} \Bigg(\left\| \boldsymbol{\Sigma}_t^{1/2} (\boldsymbol{\theta}_t - \hat{\boldsymbol{\theta}}_k) \right\|_2 \\
&\qquad\qquad + \left\| \boldsymbol{\Sigma}_t^{1/2} (\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}^\natural) \right\|_2 \Bigg) \cdot \left\| \boldsymbol{\Sigma}_t^{-1/2} \cdot \boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) \right\|_2 \\
&\overset{(c)}{\leq} 2 \sum_{k=0}^{N_T-1} \sum_{t=t_k}^{t_{k+1}-1} \Bigg(\left\| \boldsymbol{\Sigma}_{t_k}^{1/2} (\boldsymbol{\theta}_t - \hat{\boldsymbol{\theta}}_k) \right\|_2 \\
&\qquad\qquad + \left\| \boldsymbol{\Sigma}_{t_k}^{1/2} (\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}^\natural) \right\|_2 \Bigg) \cdot \left\| \boldsymbol{\Sigma}_t^{-1/2} \cdot \boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) \right\|_2 \\
&\overset{(d)}{\leq} 4 \sum_{k=0}^{N_T-1} \sum_{t=t_k}^{t_{k+1}-1} \beta_k \left\| \boldsymbol{\Sigma}_t^{-1/2} \cdot \boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) \right\|_2, \qquad \text{(D.3.7)}
\end{aligned}
$$

where the inequality (a) holds due to the fact $0 \leq \left\langle \boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}), \boldsymbol{\theta}^\natural \right\rangle \leq 1/(1-\gamma)$, the inequality (b) derived based on the Cauchy-Schwarz inequality and triangle inequality, the inequality (c) holds because $\det(\boldsymbol{\Sigma}_t) \leq 2\det(\boldsymbol{\Sigma}_{t_k})$ and Lemma 15, and the inequality (d) holds since $\boldsymbol{\theta}_t \in \mathcal{D}_k$ derided from Lemma 2.

Additionally, based on the fact that $0 \leq V_m^* \leq 1/(1-\gamma)$, we have

$$
\left\langle \boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}), \boldsymbol{\theta}_t \right\rangle - \left\langle \boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}), \boldsymbol{\theta}^\natural \right\rangle \leq \frac{1}{1-\gamma}. \qquad \text{(D.3.8)}
$$

Combining (D.3.7) and (D.3.8), we can bound J_1 as

$$
\begin{aligned}
J_1 &\leq \sum_{k=0}^{N_T-1} \sum_{t=t_k}^{t_{k+1}-1} \min\left\{ \frac{1}{1-\gamma}, 4\beta_k \left\| \boldsymbol{\Sigma}_t^{-1/2} \cdot \boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) \right\|_2 \right\} \\
&\overset{(a)}{\leq} 4 \sum_{k=0}^{N_T-1} \sum_{t=t_k}^{t_{k+1}-1} \beta_k \min\left\{ 1, \left\| \boldsymbol{\Sigma}_t^{-1/2} \cdot \boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) \right\|_2 \right\} \\
&\overset{(b)}{\leq} 4 \left(T \sum_{k=0}^{N_T-1} \sum_{t=t_k}^{t_{k+1}-1} \beta_k \min\left\{ 1, \left\| \boldsymbol{\Sigma}_t^{-1/2} \cdot \boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) \right\|_2^2 \right\} \right)^{\frac{1}{2}}, \qquad \text{(D.3.9)}
\end{aligned}
$$

where the inequality (a) holds due to the fact $1/(1-\gamma) \leq \beta$, the inequality (a) holds derived from Cauchy-Schwarz inequality. Combining the fact $\|\boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1})\|_2 \leq \sqrt{d}/(1-\gamma)$ deduced by Definition 2 and the fact $|V_m^k| \leq 1/(1-\gamma)$ implied by Lemma 3, we can utilize Lemma 14 to generate that

$$\sum_{k=0}^{N_T-1} \sum_{t=t_k}^{t_{k+1}-1} \min\left\{1, \left\|\boldsymbol{\Sigma}_t^{-1/2} \cdot \boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1})\right\|_2^2\right\} \leq 2d \log \frac{\lambda + T/(1-\gamma)^2}{\lambda}. \tag{D.3.10}$$

Substituting (D.3.10) into (D.3.9), we have

$$J_1 \leq 6\sqrt{dT\beta_T \log \frac{\lambda + T/(1-\gamma)^2}{\lambda}}. \tag{D.3.11}$$

- In terms of J_2, we begin with defining several vital notations and corresponding value functions. For any $0 \leq k \leq N_T - 1$, $t_k \leq t \leq t_{k+1} - 1$ and random action sequence $\vec{\boldsymbol{a}}_t = (a_{t-m}, \cdots, a_{t-1})$, Let $\mathcal{H}_m$ be the trajectory generated by the random sample path $\{(s_{t'}, a_{t'-m})\}_{t' \leq t}$ and the state sequence as $\vec{\boldsymbol{s}}_t = (s_{t-m}, \cdots, s_{t-1})$. We further define

$$\breve{V}_m^{\pi}(s, \vec{\boldsymbol{s}}_t) = \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r(s_{t+i}, \pi(s_{t-m+i})) \middle| s_t = s\right], \tag{D.3.12}$$

$$\breve{V}_m(\cdot) = \max_{a \in \mathcal{A}} \breve{Q}_m^k(\cdot, a, \vec{\boldsymbol{a}}_t), \tag{D.3.13}$$

$$\begin{aligned}\breve{Q}_m^k(s_t, a, \vec{\boldsymbol{a}}_t) &= r(s_t, \pi_k(s_{t-m})) \\ &\quad + \max_{\boldsymbol{\theta} \in \mathcal{D}_k} \sum_{\substack{(s_{t+1}, \cdots, s_{t+m}, s') \\ a_{t-m} \cdots a_{t-1} \in \mathcal{A}}} \langle \boldsymbol{\phi}(s_{t+1}|s_t, a_{t-m}), \boldsymbol{\theta}\rangle \cdots \langle \boldsymbol{\phi}(s'|s_{t+m}, a), \boldsymbol{\theta}\rangle \breve{V}_m(s'),\end{aligned} \tag{D.3.14}$$

$$\breve{V}_m^k(\cdot, \vec{\boldsymbol{a}}_{t+1}) = \max_{a \in \mathcal{A}} \breve{Q}_m^k(\cdot, a, \vec{\boldsymbol{a}}_t). \tag{D.3.15}$$

Recall $V_m^{\pi}(s)$ (5.1) and the update rule of V_m^k on Line 8 in Algorithm 7.

$$\begin{aligned}\eta_m^{\pi} &= [\mathbb{P}V_m^{\pi}](s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) - \breve{V}_m^{\pi}(s_{t+1}, \vec{\boldsymbol{s}}_{t+1}), \\ \eta_m^k &= [\mathbb{P}V_m^k](s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) - \breve{V}_m^k(s_{t+1}, \vec{\boldsymbol{a}}_{t+1}).\end{aligned} \tag{D.3.16}$$

Therefore, we have

$$|\eta_m^k|, |\eta_m^\pi| \leq \frac{1}{1-\gamma}, \tag{D.3.17}$$

where due to the assumption of the reward $r \in [0,1]$. We can verify that $\mathbb{E}[\eta_m^k|\mathcal{H}_m] = 0$ since s_{t+1} are sampled according to the distribution $\mathbb{P}(\cdot|s_t, a_{t-m})$. Likewise, we have $\mathbb{E}[\eta_m^\pi|\mathcal{H}_m] = 0$, which implies that

$$\mathbb{E}[\eta_m^k - \eta_m^\pi|\mathcal{H}_m] = 0 \tag{D.3.18}$$

Given $V_m^\pi(s, \vec{\boldsymbol{a}}_{t+1})$ (5.1) where $\vec{\boldsymbol{a}}_{t+1} = (a_{t-m+1}, \cdots, a_{t-1}, a_t)$ and $\breve{V}_m^\pi(s, \vec{\boldsymbol{s}}_{t+1})$ (D.3.12), the difference between these two terms yields due to the deterministic action sequence

$$\vec{\boldsymbol{a}}_{t+1} = (a_{t-m+1}, \cdots, a_{t-1}, a_t)$$

and the stochastic action sequence $(\pi(s_{t-m+1}), \cdots, \pi(s_{t-1}), \pi(s_t))$. Meanwhile, the difference between $V_m^k(s)$ with a certain action sequence $\vec{\boldsymbol{a}}_{t+1} = (a_{t-m+1}, \cdots, a_{t-1}, a_t)$ and $\breve{V}_m^k(s)$ (D.3.13) yields when computing $Q_m^k(s, a, \vec{\boldsymbol{s}}_t)$ and $\breve{Q}_m^k(s, a, \vec{\boldsymbol{s}}_t)$ based on the deterministic action sequence and the stochastic action sequence. Both of differences depend on the sum of the first m rewards, e.g.,

$$\sum_{i=0}^{m-1} \gamma^i r(s_{t+1+i}, \pi(s_{t+1-m+i})) - \gamma^i r(s_{t+1+i}, a_{t+1-m+i}),$$

thus we have

$$\mathbb{E}\left[\left(\breve{V}_m^k(s_{t+1}, \vec{\boldsymbol{a}}_{t+1}) - V_m^k(s_{t+1}, \vec{\boldsymbol{a}}_{t+1})\right) - \left(\breve{V}_m^\pi(s_{t+1}, \vec{\boldsymbol{s}}_{t+1}) - V_m^\pi(s_{t+1}, \vec{\boldsymbol{a}}_{t+1})\right)\big|\mathcal{H}_m\right] = 0, \tag{D.3.19}$$

and $|\breve{V}_m^k(s_{t+1}, \vec{\boldsymbol{a}}_{t+1}) - V_m^k(s_{t+1}, \vec{\boldsymbol{a}}_{t+1})| \leq 1/(1-\gamma)$, $|\breve{V}_m^\pi(s_{t+1}, \vec{\boldsymbol{s}}_{t+1}) - V_m^\pi(s_{t+1}, \vec{\boldsymbol{a}}_{t+1})| \leq 1/(1-\gamma)$. Combining (D.3.18) and (D.3.19), it yields that

$$\begin{aligned}
&\mathbb{E}\left[\left[\mathbb{P}(V_m^k - V_m^\pi)\right](s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) - \left(V_m^k(s_{t+1}, \vec{\boldsymbol{a}}_{t+1}) - V_m^\pi(s_{t+1}, \vec{\boldsymbol{a}}_{t+1})\right)\big|\mathcal{H}_m\right] \\
=&\mathbb{E}\Big[\eta_m^k - \eta_m^\pi + \left(\breve{V}_m^k(s_{t+1}, \vec{\boldsymbol{a}}_{t+1}) - V_m^k(s_{t+1}, \vec{\boldsymbol{a}}_{t+1})\right) - \\
&\quad \left(\breve{V}_m^\pi(s_{t+1}, \vec{\boldsymbol{s}}_{t+1}) - V_m^\pi(s_{t+1}, \vec{\boldsymbol{a}}_{t+1})\right)\big|\mathcal{H}_m\Big] \\
=&0.
\end{aligned} \tag{D.3.20}$$

Hence, we conclude that

$$\begin{aligned}
&\left|\left[\mathbb{P}(V_m^k - V_m^\pi)\right](s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) - \left(V_m^k(s_{t+1}, \vec{\boldsymbol{a}}_{t+1}) - V_m^\pi(s_{t+1}, \vec{\boldsymbol{a}}_{t+1})\right)\right| \\
\le &|V_m^k| + |V_m^\pi| + |\breve{V}_m^k(s_{t+1}, \vec{\boldsymbol{a}}_{t+1}) - V_m^k(s_{t+1}, \vec{\boldsymbol{a}}_{t+1})| + |\breve{V}_m^\pi(s_{t+1}, \vec{\boldsymbol{s}}_{t+1}) - V_m^\pi(s_{t+1}, \vec{\boldsymbol{a}}_{t+1})| \\
= &\frac{4}{1-\gamma}. \qquad \text{(D.3.21)}
\end{aligned}$$

Thus, $\left[\mathbb{P}(V_m^k - V_m^\pi)\right](s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) - \left(V_m^k(s_{t+1}, \vec{\boldsymbol{a}}_{t+1}) - V_m^\pi(s_{t+1}, \vec{\boldsymbol{a}}_{t+1})\right)$ establishes a martingale difference sequence. Moreover, Lemma 3 implies that $0 \le |V_m^k(s) - V_m^\pi(s)| \le 1/(1-\gamma) + (1-\gamma^m)/(1-\gamma)$, which can be combined with the update rule in line 7 of Algorithm 7, yielding

$$\left|\left[\mathbb{P}(V_m^k - V_m^\pi)\right](s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) - \left(V_m^k(s_{t+1}, \vec{\boldsymbol{a}}_{t+1}) - V_m^\pi(s_{t+1}, \vec{\boldsymbol{a}}_{t+1})\right)\right| \le \frac{2-\gamma^m}{1-\gamma}.$$

Hence, we can derive the bound of J_2 based on Azuma-Hoeffding inequality in Lemma 13,

$$\begin{aligned}
J_2 &= \gamma \sum_{k=0}^{N_T-1} \sum_{t=t_k}^{t_{k+1}-1} \left[\mathbb{P}(V_m^k - V_m^\pi)\right](s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) - \left(V_m^k(s_{t+1}, \vec{\boldsymbol{a}}_{t+1}) - V_m^\pi(s_{t+1}, \vec{\boldsymbol{a}}_{t+1})\right) \\
&\le \frac{2-\gamma^m}{1-\gamma}\sqrt{T \ln \frac{1}{\delta}}. \qquad \text{(D.3.22)}
\end{aligned}$$

- In terms of J_3, we arrive

$$\begin{aligned}
J_3 &= \gamma \sum_{k=0}^{N_T-1} \sum_{t=t_k}^{t_{k+1}-1} \left(V_m^k(s_{t+1}, \vec{\boldsymbol{a}}_{t+1}) - V_m^\pi(s_{t+1}, \vec{\boldsymbol{a}}_{t+1})\right) \\
&= \gamma \sum_{k=0}^{N_T-1} \Bigg[\sum_{t=t_k}^{t_{k+1}-1} \left(V_m^k(s_t, \vec{\boldsymbol{a}}_t) - V_m^\pi(s_t, \vec{\boldsymbol{a}}_t)\right) - \left(V_m^k(s_{t_k}, \vec{\boldsymbol{a}}_{t_k}) - V_m^\pi(s_{t_k}, \vec{\boldsymbol{a}}_{t_k})\right) \\
&\qquad + \left(V_m^k(s_{t_{k+1}}, \vec{\boldsymbol{a}}_{t_{k+1}}) - V_m^\pi(s_{t_{k+1}}, \vec{\boldsymbol{a}}_{t_{k+1}})\right)\Bigg] \\
&\overset{(a)}{\le} \gamma \sum_{k=0}^{N_T-1} \left[\sum_{t=t_k}^{t_{k+1}-1} \left(V_m^k(s_t, \vec{\boldsymbol{a}}_t) - V_m^\pi(s_t, \vec{\boldsymbol{a}}_t)\right) + \frac{1+\gamma^m}{1-\gamma}\right] \\
&= \gamma \sum_{k=0}^{N_T-1} \sum_{t=t_k}^{t_{k+1}-1} \left(V_m^k(s_t, \vec{\boldsymbol{a}}_t) - V_m^\pi(s_t, \vec{\boldsymbol{a}}_t)\right) + \frac{N_T\gamma(1+\gamma^m)}{1-\gamma}, \qquad \text{(D.3.23)}
\end{aligned}$$

where the inequality (a) holds due to the fact that $0 \leq |V_m^k(s, \vec{\boldsymbol{a}}_t) - V_m^\pi(s, \vec{\boldsymbol{a}}_t)| \leq 1/(1-\gamma)$, $0 \leq |V_m^k(s, \vec{\boldsymbol{a}}_t) - V_m^\pi(s, \vec{\boldsymbol{a}}_t)| \leq \gamma^m/(1-\gamma)$ implied by Lemma 3 and the update rule in line 7 of Algorithm 7.

Substituting (D.3.11), (D.3.22) and (D.3.23) into (D.3.6), we conclude

$$\begin{aligned}
&\sum_{k=0}^{N_T-1} \sum_{t=t_k}^{t_{k+1}-1} \left[V_m^k(s_t, \vec{\boldsymbol{a}}_t) - V_m^\pi(s_t, \vec{\boldsymbol{a}}_t)\right] \\
\leq & 6\sqrt{dT\beta_T \log \frac{\lambda + T/(1-\gamma)^2}{\lambda}} + \frac{2-\gamma^m}{1-\gamma}\sqrt{T \ln \frac{1}{\delta}} + \gamma \sum_{k=0}^{N_T-1} \sum_{t=t_k}^{t_{k+1}-1} \left[V_m^k(s_t, \vec{\boldsymbol{a}}_t) - V_m^\pi(s_t, \vec{\boldsymbol{a}}_t)\right] \\
& + \frac{N_T\gamma(1+\gamma^m)}{1-\gamma} + (1-\gamma). \qquad \text{(D.3.24)}
\end{aligned}$$

By summarizing the above inequalities, we have

$$\begin{aligned}
&\sum_{k=0}^{N_T-1} \sum_{t=t_k}^{t_{k+1}-1} \left[V_m^k(s_t, \vec{\boldsymbol{a}}_t) - V_m^\pi(s_t, \vec{\boldsymbol{a}}_t)\right] \\
\leq & 1 + \frac{6}{1-\gamma}\sqrt{dT\beta_T \log \frac{\lambda + T/(1-\gamma)^2}{\lambda}} + \frac{2-\gamma^m}{(1-\gamma)^2}\sqrt{T \ln \frac{1}{\delta}} + \frac{N_T\gamma(1+\gamma^m)}{(1-\gamma)^2}. \qquad \text{(D.3.25)}
\end{aligned}$$

Substituting β_T and (D.3.25) into (D.3.2) and reorganizing it, we have

$$\begin{aligned}
\text{Regret}(T) \leq & \frac{6}{1-\gamma}\sqrt{dT \log \frac{\lambda + T/(1-\gamma)^2}{\lambda}} \cdot \Bigg(\frac{8}{(1-\gamma)^2} \log\left(d\delta^{-1}\right) \\
& + 4\sqrt{\frac{2}{(1-\gamma)^2} t_k^2 \log(2t_k(t_k+1)\delta^{-1}) + m^2 t_k + \lambda d}\Bigg)^{1/2} \\
& + 1 + \frac{2-\gamma^m}{(1-\gamma)^2}\sqrt{T \ln \frac{1}{\delta}} + \frac{N_T\gamma(1+\gamma^m)}{(1-\gamma)^2} \\
\leq & \frac{6}{1-\gamma}\sqrt{dT \log \frac{\lambda + T/(1-\gamma)^2}{\lambda}} \left(\frac{8}{1-\gamma}\sqrt{\log \frac{\lambda(1-\gamma)^2 + Td}{\delta\lambda(1-\gamma)^2}} + \sqrt{\lambda d}\right) \\
& + 1 + \frac{(2-\gamma^m)\sqrt{T \log \delta^{-1}}}{(1-\gamma)^2} + \frac{\gamma(1+\gamma^m)}{(1-\gamma)^2} \min\Bigg\{2d \log \frac{\lambda + Td}{\lambda(1-\gamma)^2}, \\
& \frac{\log\left[(mT)/(1-\gamma)\right]}{1-\gamma} + m\Bigg\}, \qquad \text{(D.3.26)}
\end{aligned}$$

where the inequality (a) derives from Lemma 16. The proof is completed by taking the union bound of Lemma 2 and Lemma 13.

D.4 Proof of Theorem 9

Given a parameter vector $\boldsymbol{\theta}$, we consider the two-state parameterized-CDMDP $(\mathcal{S}, \mathcal{A}, \gamma, r, \mathbb{P}_{\boldsymbol{\theta}}, 1)$. Specifically, there are two states, i.e., the initial state x_0 and another state x_1. For each action $\mathbf{a} \in \mathcal{A}$, define a deterministic rewards, i.e., $r(x_0, \mathbf{a}) = 0$, $r(x_1, \mathbf{a}) = 1$. Moreover, the probability transition function is given by $\mathbb{P}_{\boldsymbol{\theta}}(x_0|x_0, \mathbf{a}) = 1 - \delta - \langle \mathbf{a}, \boldsymbol{\theta} \rangle, \mathbb{P}_{\boldsymbol{\theta}}(x_1|x_0, \mathbf{a}) = \delta + \langle \mathbf{a}, \boldsymbol{\theta} \rangle, \mathbb{P}_{\boldsymbol{\theta}}(x_0|x_1, \mathbf{a}) = \delta, \mathbb{P}_{\boldsymbol{\theta}}(x_1|x_1, \mathbf{a}) = 1 - \delta$.

In the CDMDP $(\mathcal{S}, \mathcal{A}, \gamma, r, \mathbb{P}_{\boldsymbol{\theta}}, 1)$ with the constant delay $m = 1$, from an observation state, we can utilize probability transition function $\mathbb{P}_{\boldsymbol{\theta}}$ and the action sequence (ω) where $\omega \in \mathcal{A}$ to *simulate* the outcome of the next time step which is the current but unobserved state. Then the optimal policy is to choose an action $\mathbf{a}_{\boldsymbol{\theta}} = [\mathrm{sgn}(\theta_i)]_{i=1}^{d-1}$ based on this approximated current state. Hence, based on the optimality Bellman equation, $V_m^*(x_0)$ and $V_m^*(x_1)$ can be approximately represented by

$$V_m^*(x_0) = r(x_0, \omega) + \gamma \mathbb{E}_{\substack{s \sim \mathbb{P}_{\boldsymbol{\theta}}(\cdot|x_0, \omega) \\ s' \sim \mathbb{P}_{\boldsymbol{\theta}}(\cdot|s, \mathbf{a}_{\boldsymbol{\theta}})}} V_m^*(s'),\ V_m^*(x_1) = r(x_1, \omega) + \gamma \mathbb{E}_{\substack{s \sim \mathbb{P}_{\boldsymbol{\theta}}(\cdot|x_1, \omega) \\ s' \sim \mathbb{P}_{\boldsymbol{\theta}}(\cdot|s, \mathbf{a}_{\boldsymbol{\theta}})}} V_m^*(s'). \tag{D.4.1}$$

We define

$$\varsigma_0 = (1 - \delta - \langle \mathbf{a}_{\boldsymbol{\theta}}, \boldsymbol{\theta} \rangle)^2 + \delta(\delta + \langle \omega, \boldsymbol{\theta} \rangle), \tag{D.4.2}$$

$$\varsigma_1 = \delta(1 - \delta - \langle \mathbf{a}_{\boldsymbol{\theta}}, \boldsymbol{\theta} \rangle) + \delta(1 - \delta), \tag{D.4.3}$$

and then substitute aforementioned r and $\mathbb{P}_{\boldsymbol{\theta}}$ into (D.4.1), yielding the following equations,

$$V_m^*(x_0) = \frac{\gamma(1 - \varsigma_0)}{(1 - \gamma)(\gamma(\varsigma_1 - \varsigma_0) + 1)}, V_m^*(x_1) = \frac{1 - \gamma \varsigma_0}{(1 - \gamma)(\gamma(\varsigma_1 - \varsigma_0) + 1)}. \tag{D.4.4}$$

Based on Lemma 17, it yields

$$\mathbb{E}\mathrm{Regret}(T) \geq \mathbb{E}\left[\sum_{t=1}^{T} \left[V_m^*(s_t, \vec{\boldsymbol{a}}_t) - \frac{1}{1 - \gamma} r(s_t, a_{t-1}) \right] - \frac{\gamma}{(1 - \gamma)^2} \right].$$

Note that the lower bound can be represented by $\mathbb{E}[N_1]$ due to $r(x_0, \mathbf{a}) = 0$ and $r(x_1, \mathbf{a}) = 1$. It yields that

$$\frac{1}{|\boldsymbol{\Gamma}|} \sum_{\boldsymbol{\theta}} \mathbb{E}[N_1] \leq \frac{T}{2} + \frac{1}{2\delta} \frac{\psi}{(d-1)|\boldsymbol{\Gamma}|} \sum_{j=1}^{d-1} \sum_{\boldsymbol{\theta}} \left[\mathbb{E}_{\boldsymbol{\theta}'}[N_0] + \frac{\sqrt{\log 2} T}{2} \sqrt{d_{\mathrm{KL}}(\mathcal{P}' \| \mathcal{P})} \right], \tag{D.4.5}$$

where the j-th coordinates of $\boldsymbol{\theta}'$ and $\boldsymbol{\theta}$ are different and the rest are same. An upper bound of (D.4.5) can be derived from Lemma 18 and 19, which depends on ψ and δ. By choosing $\psi = d\sqrt{1-\gamma}/(66\sqrt{2T})$ and $\delta = 1-\gamma$, the final result can be derived.

Finally, we consider the expectation of regret for different $\boldsymbol{\theta}$ and sum these expectations over all possible $\boldsymbol{\theta}$, given by:

$$\begin{aligned}
&\frac{1}{|\boldsymbol{\Gamma}|}\sum_{\boldsymbol{\theta}}\left[\mathbb{E}\mathrm{Regret}(T) + \frac{\gamma}{(1-\gamma)^2}\right] \\
\geq &\frac{1}{|\boldsymbol{\Gamma}|}\sum_{\boldsymbol{\theta}}\mathbb{E}\left[N_0 V_m^*(x_0) + N_1 V_m^*(x_1) - \frac{1}{1-\gamma}\sum_{t=1}^{T} r(s_t, a_{t-1})\right] \\
\overset{(a)}{=} &\frac{1}{1-\gamma}\frac{1}{|\boldsymbol{\Gamma}|}\sum_{\boldsymbol{\theta}}\mathbb{E}\left[N_0\frac{\gamma(1-\varsigma_0)}{(1-\gamma)(\gamma(\varsigma_1-\varsigma_0)+1)} + N_1\left(\frac{1-\gamma\varsigma_0}{(1-\gamma)(\gamma(\varsigma_1-\varsigma_0)+1)} - 1\right)\right] \\
= &\frac{1}{1-\gamma}\frac{1}{|\boldsymbol{\Gamma}|}\sum_{\boldsymbol{\theta}}\mathbb{E}\left[T\frac{\gamma(1-\varsigma_0)}{(1-\gamma)(\gamma(\varsigma_1-\varsigma_0)+1)} + N_1\frac{\gamma(\gamma\varsigma_1-\gamma\varsigma_0-\varsigma_1+1)}{(1-\gamma)(\gamma(\varsigma_1-\varsigma_0)+1)}\right] \\
= &T\frac{\gamma(1-\varsigma_0)}{(1-\gamma)^2(\gamma(\varsigma_1-\varsigma_0)+1)} + \frac{\gamma(\gamma\varsigma_1-\gamma\varsigma_0-\varsigma_1+1)}{(1-\gamma)^2(\gamma(\varsigma_1-\varsigma_0)+1)}\frac{1}{|\boldsymbol{\Gamma}|}\sum_{\boldsymbol{\theta}}\mathbb{E}[N_1], \qquad \text{(D.4.6)}
\end{aligned}$$

where the equality (a) derives from the definitions of $V_m^*(x_0)$, $V_m^*(x_1)$ (D.4.4) and the fact that $r(x_1, a) = 1$ and $r(x_0, a) = 0$. By Lemma 18, we can bound $|\boldsymbol{\Gamma}|^{-1}\sum_{\boldsymbol{\theta}}\mathbb{E}[N_1]$ as

$$\frac{1}{|\boldsymbol{\Gamma}|}\sum_{\boldsymbol{\theta}}\mathbb{E}[N_1] \leq \frac{T}{2} + \frac{5\psi T}{22\delta} + \frac{3}{2}\sqrt{\frac{\log 2}{2}}\frac{\psi^2 T^{\frac{3}{2}}}{d\delta^{\frac{3}{2}}}. \qquad \text{(D.4.7)}$$

Substituting (D.4.7) into (D.4.6), it yields

$$
\begin{aligned}
&\frac{1}{|\boldsymbol{\Gamma}|}\sum_{\boldsymbol{\theta}}\left[\mathbb{E}\mathrm{Regret}(T)+\frac{\gamma}{(1-\gamma)^2}\right]\\
\geq& T\frac{\gamma(1-\varsigma_0)}{(1-\gamma)^2(\gamma(\varsigma_1-\varsigma_0)+1)}+\frac{\gamma(\gamma\varsigma_1-\gamma\varsigma_0-\varsigma_1+1)}{(1-\gamma)^2(\gamma(\varsigma_1-\varsigma_0)+1)}\left(\frac{T}{2}+\frac{5\psi T}{22\delta}+\frac{3}{2}\sqrt{\frac{\log 2}{2}\frac{\psi^2T^{\frac{3}{2}}}{d\delta^{\frac{3}{2}}}}\right)\\
\overset{(a)}{\geq}&\frac{1}{(1-\gamma)^2(\gamma(\varsigma_1-\varsigma_0)+1)}\left[\frac{(1-\varsigma_0)\gamma T}{2}+\gamma(\gamma\varsigma_1-\gamma\varsigma_0-\varsigma_1+1)\left(\frac{5\psi T}{22\delta}+\frac{3}{2}\sqrt{\frac{\log 2}{2}\frac{\psi^2T^{\frac{3}{2}}}{d\delta^{\frac{3}{2}}}}\right)\right]\\
\overset{(b)}{\geq}&\frac{1}{2(1-\gamma)^2}\left[\frac{(1-\varsigma_0)\gamma T}{2}+\gamma(\gamma\varsigma_1-\gamma\varsigma_0-\varsigma_1+1)\left(\frac{5\psi T}{22\delta}+\frac{3}{2}\sqrt{\frac{\log 2}{2}\frac{\psi^2T^{\frac{3}{2}}}{d\delta^{\frac{3}{2}}}}\right)\right]\\
\overset{(c)}{\geq}&\frac{1}{2(1-\gamma)^2}\left[\frac{\psi\gamma T}{2}-\gamma\frac{10\delta\psi T}{22\delta}-\gamma 3\sqrt{\frac{\log 2}{2}\frac{\psi^2\delta T^{\frac{3}{2}}}{d\delta^{\frac{3}{2}}}}\right]\\
\overset{(d)}{\geq}&\frac{\gamma d\sqrt{T}}{2640\sqrt{\log 2}(1-\gamma)^{1.5}},
\end{aligned}
$$

where the inequality (a) holds due to the fact $\gamma(\varsigma_1-\varsigma_0)+1\leq 2$, the inequality (b) holds since $0\leq\varsigma_0,\varsigma_1\leq 1$ and $\varsigma_0\leq 1-\psi,\varsigma_1\leq 2\delta$, the inequality (c) holds due to the fact that $3\psi/2<\delta\leq 1/5$, the inequality (d) holds under the condition of $\psi=d\sqrt{1-\gamma}/(66\sqrt{2T})$ and $\delta=1-\gamma$. Thus, there exists $\boldsymbol{\theta}\in\boldsymbol{\Gamma}$ such that

$$
\mathbb{E}\mathrm{Regret}(T)\geq\frac{\gamma d\sqrt{T}}{2640\sqrt{\log 2}(1-\gamma)^{\frac{3}{2}}}-\frac{\gamma}{(1-\gamma)^2}. \tag{D.4.8}
$$

Hence, we can conclude the proof by selecting $\tilde{\boldsymbol{\theta}}=(\boldsymbol{\theta}^\top,1)^\top\in\mathbb{R}^d$.

D.5 Proof of Lemma 11

Suppose for any $0\leq k\leq N_T-1$, $t_k\leq t\leq t_{k+1}-1$, action sequence for both Q_m^k and $\tilde{Q}_m^k$ is $\vec{\boldsymbol{a}}_t=(a_{-m},\cdots,a_{-1})$, where $\tilde{Q}_m^k(s,a,\vec{\boldsymbol{a}}_t)$ for any $(s,a)\in\mathcal{S}\times\mathcal{A}$ is defined as

$$
\mathbb{E}\left[\sum_{i=0}^{m-1}\gamma^i r(s_{t-(m+1)+i},a_{-m+i})+\sum_{i=m}^{\infty}\gamma^i r(s_{t-(m+1)+i},\pi^k(s_{t-(2m+1)+i}))\middle|s_{t-(m+1)}=s\right].
$$

We begin with bounding $Q_m^k(s,a,\vec{\boldsymbol{a}}_t)-\tilde{Q}_m^k(s,a,\vec{\boldsymbol{a}}_t)$.

By the update rule in Algorithm 7, we denote $V_m^{(t-1)}(\cdot) = \max_{a \in \mathcal{A}} Q_m^k(\cdot, a, \vec{\boldsymbol{a}}_t)$, thus have

$$Q_m^k(s, a, \vec{\boldsymbol{a}}_t) = r(s, a_{-m}) + \gamma \max_{\boldsymbol{\theta} \in \mathcal{D}k} \sum (s_{t+1}, \cdots, s_{t+m}, s')$$

$$\langle \boldsymbol{\phi}(s_{t+1}|s, a_{-m}), \boldsymbol{\theta} \rangle \cdots \langle \boldsymbol{\phi}(s'|s_{t+m}, a), \boldsymbol{\theta} \rangle V_m^{(t-1)}(s'),$$

$$\tilde{Q}_m^k(s, a, \vec{\boldsymbol{a}}t) = r(s, a_{-m}) + \gamma \max_{\boldsymbol{\theta} \in \mathcal{D}k} \sum s_{t-m} \cdots s_{t-1}, s' \in \mathcal{S}$$

$$\langle \boldsymbol{\phi}(s_{t-m}|s, a_{-m}), \boldsymbol{\theta} \rangle \cdots \langle \boldsymbol{\phi}(s'|s_{t-1}, a), \boldsymbol{\theta} \rangle V_m^{(t-m)}(s').$$

Hence for any $(s, a) \in \mathcal{S} \times \mathcal{A}$ and an action sequence $\vec{\boldsymbol{a}}_t = (a_{-m}, \cdots, a_{-1})$, we have

$$\begin{aligned}
&\left| Q_m^k(s, a, \vec{\boldsymbol{a}}_t) - \tilde{Q}_m^k(s, a, \vec{\boldsymbol{a}}_t) \right| \\
=&\gamma \left| \max_{\boldsymbol{\theta} \in \mathcal{D}_k} \sum_{(s_{t+1}, \cdots, s_{t+m}, s')} \langle \boldsymbol{\phi}(s_{t+1}|s, a_{-m}), \boldsymbol{\theta} \rangle \cdots \langle \boldsymbol{\phi}(s'|s_{t+m}, a), \boldsymbol{\theta} \rangle V_m^{(t-1)}(s') \right. \\
&\left. - \max_{\boldsymbol{\theta} \in \mathcal{D}_k} \sum_{s_{t-m} \cdots s_{t-1}, s' \in \mathcal{S}} \langle \boldsymbol{\phi}(s_{t-m}|s, a_{-m}), \boldsymbol{\theta} \rangle \cdots \langle \boldsymbol{\phi}(s'|s_{t-1}, a), \boldsymbol{\theta} \rangle V_m^{(t-m)}(s') \right| \\
\overset{(a)}{\leq}&\gamma \max_{\boldsymbol{\theta} \in \mathcal{D}_k} \left| \sum_{s^1 \cdots s^m, s' \in \mathcal{S}} \langle \boldsymbol{\phi}(s^1|s, a_{-m}), \boldsymbol{\theta} \rangle \cdots \langle \boldsymbol{\phi}(s'|s^m, a), \boldsymbol{\theta} \rangle \left[V_m^{(t-1)}(s') - V_m^{(t-m)}(s') \right] \right| \\
\overset{(b)}{=}&\gamma \left| \sum_{s^1 \cdots s^m, s' \in \mathcal{S}} \langle \boldsymbol{\phi}(s^1|s, a_{-m}), \tilde{\boldsymbol{\theta}} \rangle \cdots \langle \boldsymbol{\phi}(s'|s^m, a), \tilde{\boldsymbol{\theta}} \rangle \left[V_m^{(t-1)}(s') - V_m^{(t-m)}(s') \right] \right| \qquad \text{(D.5.1)}
\end{aligned}$$

where the inequality (a) derives based on the upper bound of $\max$ function, and the equation (b) holds since $\tilde{\boldsymbol{\theta}}$ is the solution of the maximum optimization problem. We can further bound (D.5.1) as follows:

$$\begin{aligned}
&\gamma \left| \sum_{s^1 \cdots s^m, s' \in \mathcal{S}} \langle \boldsymbol{\phi}(s^1|s, a_{-m}), \tilde{\boldsymbol{\theta}} \rangle \cdots \langle \boldsymbol{\phi}(s'|s^m, a), \tilde{\boldsymbol{\theta}} \rangle \left[V_m^{(t-1)}(s') - V_m^{(t-m)}(s') \right] \right| \\
\overset{(a)}{\leq}&\gamma \max_{s' \in \mathcal{S}} \left| V_m^{(t-1)}(s') - V_m^{(t-m)}(s') \right| \\
\overset{(b)}{\leq}&\gamma \max_{s' \in \mathcal{S}} \left| \max_{a' \in \mathcal{A}} Q_m^k(s', a', \vec{\boldsymbol{a}}_{t-1}) - \max_{a' \in \mathcal{A}} \tilde{Q}_m^k(s', a', \vec{\boldsymbol{a}}_{t-1}) \right| \\
\leq&\gamma \max_{(s', a') \in \mathcal{S} \times \mathcal{A}} \left| Q_m^k(s', a', \vec{\boldsymbol{a}}_{t-1}) - \tilde{Q}_m^k(s', a', \vec{\boldsymbol{a}}_{t-1}) \right|, \qquad \text{(D.5.2)}
\end{aligned}$$

where action sequence is $\vec{\boldsymbol{a}}_{t-1} = (a_{-m-1}, a_{-m}, \cdots, a_{-2})$. Here, the inequality (a) holds since

$$\Big|\langle \boldsymbol{\phi}(s'|s, a_{-m}), \tilde{\boldsymbol{\theta}}\rangle \cdots \langle \boldsymbol{\phi}(s'|s^m, a), \tilde{\boldsymbol{\theta}}\rangle f(s,a)\Big| \le \max_{s' \in \mathcal{S}} |f(s')|$$

for any (s, s', a) with the action sequence $(a_{-m}, \cdots, a_{-1})$, the inequality (b) is derived based on the contraction property of $\max$ function. Substituting (D.5.2) into (D.5.1) and maximizing $|Q_m^k(s, a, \vec{\boldsymbol{a}}_t) - \tilde{Q}_m^k(s, a, \vec{\boldsymbol{a}}_t)|$ over (s, a), it yields

$$\begin{aligned}
&\max_{(s,a)\in\mathcal{S}\times\mathcal{A}} \Big|Q_m^k(s, a, \vec{\boldsymbol{a}}_t) - \tilde{Q}_m^k(s, a, \vec{\boldsymbol{a}}_t)\Big| \\
\le&\gamma \max_{(s,a)\in\mathcal{S}\times\mathcal{A}} \Big|Q_m^k(s, a, \vec{\boldsymbol{a}}_{t-1}) - \tilde{Q}_m^k(s, a, \vec{\boldsymbol{a}}_{t-1})\Big|. \\
\le&\gamma^{t-t_k-m+1} \max_{(s,a)\in\mathcal{S}\times\mathcal{A}} \Big|Q_m^k(s, a, \vec{\boldsymbol{a}}_{m-1}) - \tilde{Q}_m^k(s, a, \vec{\boldsymbol{a}}_{m-1})\Big| \\
\le&\gamma^{t-t_k-m+1} \max_{(s,a)\in\mathcal{S}\times\mathcal{A}} \left|r(s, a_{-m}) + \sum_{i=1}^{m-1} r(s_{m-1+i}, a_{-m+i}) - \frac{1}{1-\gamma}\right| \\
\overset{(a)}{\le}& m\gamma^{t-t_k-m+1},
\end{aligned} \tag{D.5.3}$$

where the inequality (a) holds since $0 \le r(s,a) \le 1$ for all (s,a). Hence $Q_m^k(s, a, \vec{\boldsymbol{a}}_t) - \tilde{Q}_m^k(s, a, \vec{\boldsymbol{a}}_t)$ can be bounded via $Q_m^k(s, a, \vec{\boldsymbol{a}}_t) - \tilde{Q}_m^k(s, a, \vec{\boldsymbol{a}}_t) \le m\gamma^{t-t_k-m+1}$.

Recall that $V_m(\cdot) = \max_{a\in\mathcal{A}} Q_m^k(\cdot, a, \vec{\boldsymbol{a}}_t)$, we thus have

$$\begin{aligned}
&Q_m^k(s_t, a, \vec{\boldsymbol{a}}_t) \\
&= r(s_t, a_{t-m}) + \gamma \max_{\boldsymbol{\theta}\in\mathcal{D}k} \sum (s_{t+1}, \cdots, s_{t+m}, s') \langle \boldsymbol{\phi}(s_{t+1}|s_t, a_{t-m}), \boldsymbol{\theta}\rangle \cdots \\
&\quad \langle \boldsymbol{\phi}(s'|s_{t+m}, a), \boldsymbol{\theta}\rangle V_m(s') \\
&\overset{(a)}{=} r(s_t, a_{t-m}) + \gamma \sum_{(s_{t+1},\cdots,s_{t+m},s')} \langle \boldsymbol{\phi}(s_{t+1}|s_t, a_{t-m}), \tilde{\boldsymbol{\theta}}\rangle \cdots \langle \boldsymbol{\phi}(s'|s_{t+m}, a), \tilde{\boldsymbol{\theta}}\rangle V_m(s') \\
&= r(s_t, a_{t-m}) + \gamma \sum_{(s_{t+1},\cdots,s_{t+m},s')} \langle \boldsymbol{\phi}(s_{t+1}|s_t, a_{t-m}), \tilde{\boldsymbol{\theta}}\rangle \cdots \\
&\qquad \langle \boldsymbol{\phi}(s'|s_{t+m}, a), \tilde{\boldsymbol{\theta}}\rangle V_m^k(s', \vec{\boldsymbol{a}}_{t+m+1}) \\
&\quad + \gamma \sum (s_{t+1}, \cdots, s_{t+m}, s') \langle \boldsymbol{\phi}(s_{t+1}|s_t, a_{t-m}), \tilde{\boldsymbol{\theta}}\rangle \cdots \\
&\qquad \langle \boldsymbol{\phi}(s'|s_{t+m}, a), \tilde{\boldsymbol{\theta}}\rangle [V_m(s') - V_m^k(s', \vec{\boldsymbol{a}}_{t+m+1}))] \\
&\overset{(b)}{\leq} r(s_t, a_{t-m}) + \gamma \tilde{\mathbb{P}} V_m^k(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) \\
&\qquad + \gamma \tilde{\mathbb{P}}[\tilde{V}_m^k - V_m^k](s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) \\
&\overset{(c)}{\leq} r(s_t, a_{t-m}) + \gamma \tilde{\mathbb{P}} V_m^k(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) \\
&\qquad + \gamma \max (s, a) \in \mathcal{S}\times\mathcal{A} \left| Q_m^k(s, a, \vec{\boldsymbol{a}}_t) - \tilde{Q}_m^k(s, a, \vec{\boldsymbol{a}}_t) \right| \\
&\overset{(d)}{\leq} r(s_t, a_{t-m}) + \gamma \tilde{\mathbb{P}} V_m^k(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1}) + m\gamma^{t-tk-m+1}
\end{aligned} \tag{D.5.4}$$

where the equation (a) holds since $\tilde{\boldsymbol{\theta}}$ is the solution of the maximum optimization problem, $\tilde{\mathbb{P}}(s'|s_t, a_{t-m}) = \langle \tilde{\boldsymbol{\theta}}, \boldsymbol{\phi}(s'|s_t, a_{t-m})\rangle$. Additionally, the action sequence is denoted as

$$\vec{\boldsymbol{a}}_{t+1} = (a_{t-m+1}, \cdots, a_{t-1}, a).$$

Specifically, $\tilde{V}_m^k(s', \vec{\boldsymbol{a}}_{t+1})$ denotes as

$$\mathbb{E}\left[\sum_{i=0}^{m-1} \gamma^i r(s_{t-m+i}, a_{t-m+1+i}) + \sum_{i=m}^{\infty} \gamma^i r(s_{t-m+i}, \pi^k(s_{t-2m+i})) \middle| s_{t-m} = s' \right].$$

The inequality (b) holds based on the fact that the reward $r \in [0, 1]$. The inequality (c) can be derived since $|\tilde{\mathbb{P}} f(s_t, a_{t-m}, \vec{\boldsymbol{a}}_t)| \leq \max_{s'\in\mathcal{S}} |f(s', \vec{\boldsymbol{a}}_t)|$ and $\max_s |\tilde{\mathbb{P}} V_m^k(s, \vec{\boldsymbol{a}}_t) - \tilde{\mathbb{P}} V_m^k(s, \vec{\boldsymbol{a}}_t)| \leq$

$\max_{s,a} |Q_m^k(s, a, \vec{\boldsymbol{a}}_t) - \tilde{Q}_m^k(s, a, \vec{\boldsymbol{a}}_t)|$, the inequality (d) derives from (D.5.3). The proof can be completed by taking $\boldsymbol{\theta}_t = \tilde{\boldsymbol{\theta}}$.

D.6 Proof of Lemma 18

Based on the initial state $x_0 \in \mathcal{S}$ and action sequence (ω), it yields

$$\begin{aligned}
\mathbb{E}N_1 &= \sum_{t=2}^{T} \mathbb{P}(s_t = x_1) \\
&= \mathbb{P}_{\boldsymbol{\theta}}(s_2 = x_1 | s_1 = x_0, a_1 = \omega) \\
&\quad + \underbrace{\sum_{t=3}^{T} \mathbb{P}(s_t = x_1 | s_{t-1} = x_1, s_{t-2} = x_1)\mathbb{P}(s_{t-1} = x_1 | s_{t-2} = x_1)\mathbb{P}(s_{t-2} = x_1)}_{I_1} \\
&\quad + \underbrace{\sum_{t=3}^{T} \mathbb{P}(s_t = x_1 | s_{t-1} = x_0, s_{t-2} = x_1)\mathbb{P}(s_{t-1} = x_0 | s_{t-2} = x_1)\mathbb{P}(s_{t-2} = x_1)}_{I_2} \\
&\quad + \underbrace{\sum_{t=3}^{T} \mathbb{P}(s_t = x_1, s_{t-1} = x_1, s_{t-2} = x_0)}_{I_3} \\
&\quad + \underbrace{\sum_{t=3}^{T} \mathbb{P}(s_t = x_1, s_{t-1} = x_0, s_{t-2} = x_0)}_{I_4}
\end{aligned} \tag{D.6.1}$$

In terms of I_1, since $\mathbb{P}(s_t = x_1 | s_{t-1} = x_1) = 1 - \delta$ regardless of action, we have

$$I_1 = (1-\delta)^2 \sum_{t=3}^{T} \mathbb{P}(s_{t-2} = x_1) = (1-\delta)^2 \mathbb{E}N_1 - (1-\delta)^2 \mathbb{P}(s_{T-1} = x_1) - (1-\delta)^2 \mathbb{P}(s_T = x_1). \tag{D.6.2}$$

Likewise, I_2 can be decomposed as

$$I_2 = \delta(1-\delta)\mathbb{E}N_1 - \delta(1-\delta)\mathbb{P}(s_{T-1} = x_1) - \delta(1-\delta)\mathbb{P}(s_T = x_1) \tag{D.6.3}$$

To bound I_3, we can decompose I_3 as follows:

$$\begin{aligned}
I_3 &= \mathbb{P}(s_3 = x_1|s_2 = x_1)\mathbb{P}_{\boldsymbol{\theta}}(s_2 = x_1|s_1 = x_0, a_1 = \omega) \\
&\quad + \sum_{t=4}^{T}\sum_{\mathbf{a}} \mathbb{P}(s_t = x_1|s_{t-1} = x_1)\mathbb{P}(s_{t-1} = x_1|s_{t-2} = x_0, a_{t-2} = \mathbf{a})\mathbb{P}(s_{t-2} = x_0, a_{t-2} = \mathbf{a}) \\
&= (1-\delta)\mathbb{P}_{\boldsymbol{\theta}}(s_2 = x_1|s_1 = x_0, a_1 = \omega) + \sum_{t=4}^{T}\sum_{\mathbf{a}}(1-\delta)(\delta + \langle \mathbf{a}, \boldsymbol{\theta}\rangle)\mathbb{P}(s_{t-2} = x_0, a_{t-2} = \mathbf{a}) \\
&= (1-\delta)\mathbb{P}_{\boldsymbol{\theta}}(s_2 = x_1|s_1 = x_0, a_1 = \omega) \\
&\quad + \sum_{\mathbf{a}}(1-\delta)(\delta + \langle \mathbf{a}, \boldsymbol{\theta}\rangle)\Big[\mathbb{E}N_0^{\mathbf{a}} - \mathbb{P}(s_{T-1} = x_0, a_{T-1} = \mathbf{a}) - \mathbb{P}(s_T = x_0, a_T = \mathbf{a})\Big].
\end{aligned} \tag{D.6.4}$$

Similarly, I_4 can be decomposed as

$$\begin{aligned}
I_4 &= \mathbb{P}_{\boldsymbol{\theta}}(s_2 = x_0|s_1 = x_0, a_1 = \omega)\sum_{\mathbf{a}}(\delta + \langle \mathbf{a}, \boldsymbol{\theta}\rangle) \\
&\quad + \sum_{\mathbf{a}}(\delta + \langle \mathbf{a}, \boldsymbol{\theta}\rangle)(1 - \delta - \langle \mathbf{a}, \boldsymbol{\theta}\rangle)\Big[\mathbb{E}N_0^{\mathbf{a}} - \mathbb{P}(s_{T-1} = x_0, a_{T-1} = \mathbf{a}) - \mathbb{P}(s_T = x_0, a_T = \mathbf{a})\Big]
\end{aligned} \tag{D.6.5}$$

Substituting (D.6.2)-(D.6.5) into (D.6.1) and reorganizing it, we have

$$\begin{aligned}
\mathbb{E}N_1 &= \sum_{\mathbf{a}}(1 + \langle \mathbf{a}, \boldsymbol{\theta}\rangle/\delta)(1 - \delta - \langle \mathbf{a}, \boldsymbol{\theta}\rangle)\mathbb{E}N_0^{\mathbf{a}} + \sum_{\mathbf{a}}(1 + \langle \mathbf{a}, \boldsymbol{\theta}\rangle/\delta)\mathbb{P}_{\boldsymbol{\theta}}(s_2 = x_0|s_1 = x_0, a_1 = \omega) \\
&\quad - \Bigg[\frac{1-\delta}{\delta}\left(\mathbb{P}(s_{T-1} = x_1) + \mathbb{P}(s_T = x_1)\right) + \sum_{\mathbf{a}}(1 + \langle \mathbf{a}, \boldsymbol{\theta}\rangle/\delta)(2 - 2\delta - \langle \mathbf{a}, \boldsymbol{\theta}\rangle) \\
&\quad \cdot \left(\mathbb{P}(s_{T-1} = x_0, a_{T-1} = \mathbf{a}) + \mathbb{P}(s_T = x_0, a_T = \mathbf{a})\right)\Bigg] \qquad \text{(D.6.6)} \\
&\le \sum_{\mathbf{a}}(1 + \langle \mathbf{a}, \boldsymbol{\theta}\rangle/\delta)\mathbb{E}N_0^{\mathbf{a}} + \delta^{-1}\sum_{\mathbf{a}}\langle \mathbf{a}, \boldsymbol{\theta}\rangle\mathbb{E}N_0^{\mathbf{a}} \\
&= \mathbb{E}N_0 + \delta^{-1}\sum_{\mathbf{a}}\langle \mathbf{a}, \boldsymbol{\theta}\rangle\mathbb{E}N_0^{\mathbf{a}}, \qquad \text{(D.6.7)}
\end{aligned}$$

where the last term in (D.6.6) is non-negative due to the fact $\langle \mathbf{a}, \boldsymbol{\theta}\rangle \ge -\psi \ge -\delta$. From (D.6.7), we have

$$\mathbb{E}N_1 \le T/2 + \delta^{-1}\sum_{\mathbf{a}}\langle \mathbf{a}, \boldsymbol{\theta}\rangle\mathbb{E}N_0^{\mathbf{a}}. \tag{D.6.8}$$

Denote the last term in (D.6.6) as υ, we arrive at bounding $\mathbb{E}N_0$. By (D.6.7), we have

$$\begin{aligned}
\mathbb{E}N_1 &\overset{(a)}{\geq} (1-\delta)\mathbb{E}N_0 + (1-\delta)/\delta \sum_{\mathbf{a}} \langle \mathbf{a}, \boldsymbol{\theta} \rangle \mathbb{E}N_0^{\mathbf{a}} - \upsilon \\
&\overset{(b)}{\geq} (1-\delta)(1-\psi/\delta)\mathbb{E}N_0 - \frac{1-\delta}{\delta}\left[\mathbb{P}(s_{T-1}=x_1) + \mathbb{P}(s_T = x_1)\right] \\
&\quad - 2\left(1+\frac{\psi}{\delta}\right)\left[\mathbb{P}(s_{T-1}=x_0) + \mathbb{P}(s_T = x_0)\right] \\
&= (1-\delta)(1-\psi/\delta)\mathbb{E}N_0 - 1/\delta + \frac{1-3\delta-2\psi}{\delta}\left[\mathbb{P}(s_{T-1}=x_0) + \mathbb{P}(s_T = x_0)\right] \\
&\overset{(c)}{\geq} (1-\delta)(1-\psi/\delta)\mathbb{E}N_0 - (1-\delta)/\delta, \qquad \text{(D.6.9)}
\end{aligned}$$

where the inequality (a) holds since $\langle \mathbf{a}, \boldsymbol{\theta} \rangle \leq \psi \leq \delta$ and the inequality (b) holds due to the fact that $\langle \mathbf{a}, \boldsymbol{\theta} \rangle \leq \psi$, the inequality (c) holds since $\mathbb{P}(s_{T-1}=x_0) > 0, \mathbb{P}(s_T = x_0) > 0$ and $\delta \leq 1/5$. (D.6.9) implies that

$$\mathbb{E}N_0 \leq \frac{T + (1-\delta)/\delta}{(1-\delta)(1-\psi/\delta)+1} \overset{(a)}{\leq} \frac{10}{11}T,$$

where the inequality (a) holds since $3\psi/2 \leq \delta \leq 1/5$ and $(1-\delta)/\delta < T/11$.

D.7 Technical Lemmas

Lemma 12 *Based on the definitions of $\boldsymbol{\theta}^{\natural}$ and $\hat{\boldsymbol{\theta}}_k$ in Section 5.7.1, for any $0 \leq j \leq k-1$ and $t_j \leq i \leq t_{j+1}-1$ with the action pending sequence $(a_{i-m}, \cdots, a_{i-1})$, we have for any $\tau > 0$, with probability at least $1-\delta$,*

$$\sum_{j=0}^{k-1} \sum_{i=t_j}^{t_{j+1}-1} Z_{i,m} < \frac{1}{\tau}\log\left(\frac{1}{\delta}\right) + \tau\,\frac{t_k(1-\gamma)^2}{2}\,. \qquad \text{(D.7.1)}$$

Lemma 13 (Azuma-Hoeffding inequality) *Consider a real-valued martingale sequence $\{X_n\}_{n=0}^{\infty}$ satisfying that $|X_n - X_{n-1}| \leq \mu$ for $\forall n \in \mathbb{N}$ where the constant $\mu \geq 0$. Then for $n \geq 1$, we have*

$$|X_n - X_0| \leq 2\mu\sqrt{n \log \delta^{-1}}, \qquad \text{(D.7.2)}$$

with probability exceeding $1-\delta$.

Lemma 14 (Lemma 11 in [147]) *Consider a sequence* $\{\mathbf{x}_n\}_{n=1}^T \subset \mathbb{R}^d$ *satisfying that* $\|\mathbf{x}_n\|_2 \le \mu$. *Then with* $\mathbf{A}_0 = \lambda \mathbf{I}$ *and* $\mathbf{A}_t = \mathbf{A}_0 + \sum_{n=1}^{t-1} \mathbf{x}_n \mathbf{x}_n^\top$, *we have*

$$\sum_{n=1}^{T} \min\left\{1, \|\mathbf{A}_{n-1}^{-1}\mathbf{x}_n\|_2^2\right\} \le 2d \log \frac{d\lambda + T\mu^2}{d\lambda}.$$

Lemma 15 (Lemma 12 in [147]) *Consider positive definite matrices* $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{d\times d}$ *satisfying that* $\mathbf{A} \succeq \mathbf{B}$. *Then for any* $\mathbf{x} \in \mathbb{R}^d$, *we have*

$$\|\mathbf{A}\mathbf{x}\|_2 \le \|\mathbf{B}\mathbf{x}\|_2 \cdot \sqrt{\det(\mathbf{A})/\det(\mathbf{B})}.$$

Lemma 16 *Recall* N_T *defined in Section 5.7 and the selection of* α *(5.7), we have*

$$N_T \le \min\left\{2d\log[(\lambda + dT)/(\lambda(1-\gamma)^2)], \frac{\log[(mT)/(1-\gamma)]}{1-\gamma} + m\right\}.$$

Lemma 16 shows that $N_T = \tilde{O}(d)$ times are required for Algorithm 7 to update its policy until converging to the optimal policy.

Proof: *Theorem 10 can be proved based on the transformation between sample complexity of exploration and regret bound. The details are deferred to Appendix 5.7.4.* □

D.8 Proof of Theorem 9

The essential argument used in the proof of Theorem 9 is to construct a class of hard-to-learn CDMDPs. The construction of these CDMDPs is briefly introduced here and the details on the proof are provided in Appendix D.4.

Given a vector $\boldsymbol{\theta}$, we consider the CDMDP $(\mathcal{S}, \mathcal{A}, \gamma, r, \mathbb{P}_{\boldsymbol{\theta}}, 1)$. Specifically, the state space $\mathcal{S}$ contains two states, i.e., the initial state x_0 and another state x_1. The action space $\mathcal{A}$ contains 2^{d-1} vectors $\mathbf{a} \in \{-1, 1\}^{d-1}$. For each action $\mathbf{a} \in \mathcal{A}$, we define a deterministic rewards, i.e., $r(x_0, \mathbf{a}) = 0, r(x_1, \mathbf{a}) = 1$. Moreover, the probability transition function is given by $\mathbb{P}_{\boldsymbol{\theta}}(x_0|x_0, \mathbf{a}) = 1 - \delta - \langle \mathbf{a}, \boldsymbol{\theta}\rangle, \mathbb{P}_{\boldsymbol{\theta}}(x_1|x_0, \mathbf{a}) = \delta + \langle \mathbf{a}, \boldsymbol{\theta}\rangle, \mathbb{P}_{\boldsymbol{\theta}}(x_0|x_1, \mathbf{a}) = \delta, \mathbb{P}_{\boldsymbol{\theta}}(x_1|x_1, \mathbf{a}) = 1 - \delta$. Here, $\boldsymbol{\theta}$

has dimension of $(d-1)$ such that $\boldsymbol{\theta} \in \boldsymbol{\Gamma}$, $\boldsymbol{\Gamma} = \{-\psi/(d-1), \psi/(d-1)\}^{d-1}$, where $0 < \delta$, $0 < \psi \leq d-1$ are pre-defined parameters. Note that CDMDP $(\mathcal{S}, \mathcal{A}, \gamma, r, \mathbb{P}_{\boldsymbol{\theta}}, 1)$ can be represented as a parameterized-CDMDP with the parameter vector $\boldsymbol{\Theta} = (\boldsymbol{\theta}^\top, 1)^\top \in \mathbb{R}^d$ and corresponding feature mapping $\boldsymbol{\phi}(s'|s, \mathbf{a})$ such that $\mathbb{P}_{\boldsymbol{\theta}}(s'|s, \mathbf{a}) = \langle \boldsymbol{\phi}(s'|s, \mathbf{a}), \boldsymbol{\Theta} \rangle$.

Selecting $\psi = d\sqrt{1-\gamma}/(66\sqrt{2T})$ and $\delta = 1-\gamma$. Without loss of generality, we consider a deterministic policy π, since the regret for a stochastic policy π is no less than the one for the deterministic policy. For the trajectory of states in T time steps, suppose the initial observation state $s_1 = x_0$ with the action sequence $(\mathbf{a}_0)$ where $\mathbf{a}_0 \in \mathcal{A}$, we have $s_{t+1} \sim \mathbb{P}_{\boldsymbol{\theta}}(\cdot|s_t, a_{t-1})$, a_t is further decided by π_t.

Let $\mathcal{P}$ denote the distribution over $\mathcal{S}^T$, which depends on the random variable $\boldsymbol{\theta}$, where $s_1 = x_0$. Let $\mathbb{E}$ denote the expectation w.r.t. distribution $\mathcal{P}$. Suppose we have an CDMDP $M(\mathcal{S}, \mathcal{A}, \gamma, r, \mathbb{P}_{\boldsymbol{\theta}}, 1)$. During this proof, the starting state s_1 is set to be x_0. For simplicity, let Regret(T) denote the regret of $M(\mathcal{S}, \mathcal{A}, \gamma, r, \mathbb{P}_{\boldsymbol{\theta}}, 1)$ with the policy π within the duration T.

To achieve the goal, we begin with several fundamental lemmas.

Lemma 17 *Consider a parameterized-transition CDMDP $(\mathcal{S}, \mathcal{A}, \gamma, r, \mathbb{P}_{\boldsymbol{\theta}}, 1)$ with action sequence $\vec{\boldsymbol{a}}_t = (a_{t-1})$, the regret Regret$(T)$ holds that*

$$\mathbb{E}\mathit{Regret}(T) \geq \mathbb{E}\Bigg[\sum_{t=1}^{T}\Big[V_m^*(s_t, \vec{\boldsymbol{a}}_t) - \frac{1}{1-\gamma} r(s_t, a_{t-1})\Big] - \frac{\gamma}{(1-\gamma)^2}\Bigg].$$

Theorem 17 implies that we can compute the lower bound of regret via the lower bound of $V_m^*(s_t, \vec{\boldsymbol{a}}_t) - r(s_t, a_{t-1})/(1-\gamma)$. Therein, the calculation of $V_m^*(x_0)$ and $V_m^*(x_1)$ are provided in Appendix D.4. Moreover, the reward function $r(s_t, a_{t-1})$ for $t = 1, \cdots, T$ can be characterized by the total number of visiting the state x_1 due to the assumption on the reward function such that $r(x_0, \mathbf{a}) = 0$, $r(x_1, \mathbf{a}) = 1$.

Denote N_0 as the total number of visits to the state x_0, while denote N_1 as the total number of visits to the state x_1. Define $N_0^{\mathbf{a}}$ as the total number of visiting the state x_0 followed by executing action $\mathbf{a}$. The relation between three notations $\mathbb{E}[N_1]$, $\mathbb{E}[N_0^{\mathbf{a}}]$ and $\mathbb{E}[N_0]$ is presented in the following lemma.

Lemma 18 *Assume that $3\psi/2 \leq \delta \leq 1/5$ and $(1-\delta)/\delta < T/11$, then for $\mathbb{E}[N_1]$ and $\mathbb{E}[N_0]$, we have*

$$\mathbb{E}[N_1] \leq \frac{T}{2} + \frac{1}{\delta}\sum_{\mathbf{a}} \langle \mathbf{a}, \boldsymbol{\theta} \rangle \mathbb{E}[N_0^{\mathbf{a}}], \textit{ and } \quad \mathbb{E}[N_0] \leq 10T/11.$$

***Proof**: The proof is provided in Appendix D.6.* □

Additionally, Next lemma gives the bound for KL divergence, which can be derived based on Lemma 13 in [188].

Lemma 19 *Consider two parameterized-transition CDMDPs $(\mathcal{S}, \mathcal{A}, \gamma, r, \mathbb{P}_{\boldsymbol{\theta}}, 1)$ and $(\mathcal{S}, \mathcal{A}, \gamma, r, \mathbb{P}_{\boldsymbol{\theta}'}, 1)$. Assume that $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ only differs from j-th element, $3\psi/2 \leq \delta \leq 1/5$. Then, for $\mathcal{P}$ w.r.t $\boldsymbol{\theta}$ and $\mathcal{P}'$ w.r.t $\boldsymbol{\theta}'$, the KL divergence between $\mathcal{P}'$ and $\mathcal{P}$ is given by:*

$$d_{KL}(\mathcal{P}' \| \mathcal{P}) \leq \frac{36\psi^2}{(d-1)^2\delta}\mathbb{E}N_0.$$

Based on Lemma 19 and Lemma 18, we can represent the upper bound of $\mathbb{E}[N_1]$ for ψ and δ. Thus, the lower bound of the regret can be characterized by the representations of $V_m^*(x_0)$, $V_m^*(x_1)$ and the upper bound of $\mathbb{E}[N_1]$. The details on the proof are deferred to Appendix D.4.

D.9 Details on Implementation of Algorithm 7

In implementing Algorithm 7, a key focus is the efficient computation of $\boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1})$ and $\langle \boldsymbol{\phi}(s_{t+1}|s_t, a_{t-m}), \boldsymbol{\theta} \rangle \cdots \langle \boldsymbol{\phi}(s'|s_{t+m}, a), \boldsymbol{\theta} \rangle V_m(s')$. Monte Carlo integration is one approach, and its details are outlined below. As Algorithm 7 is an online reinforcement learning algorithm, it updates the action sequence and avoids caching all past observations. Consequently, it incurs only $O(d^2)$ space complexity to store a vector $\mathbf{b}_t$ and a matrix $\boldsymbol{\Sigma}_t$, along with $O(kn)$ for the action sequence where $a \in \mathcal{A}$ and $a \in \mathbb{R}^n$.

Algorithm 7 offers a versatile framework for addressing model-based reinforcement learning in delayed feedback environments. This algorithm seamlessly integrates with regression methods and planning algorithms from [128], sidestepping stringent assumptions on the Markovian dynamics, such as deterministic or mildly stochastic probability transition functions. The proposed parameterized-transition assumption is more general than the state-of-the-art assumption in [128], encompassing a broader class of nontrivial and vital Markov Decision Processes (MDPs).

Focusing on the reinforcement learning discounted parameterized-transition Constrained Markov Decision Process (CDMDP) with delayed feedback, our algorithm leverages two primary strategies: associating the value function with the pending action sequence and employing a parameterized probability transition function $\mathbb{P}$ to simulate outcomes for k time steps based on the action sequence.

Optimistic planning, generally computationally intractable, becomes particularly challenging in delayed feedback environments. Fortunately, randomized approaches like [189, 190] and approximate dynamic programming methods offer tractable and good approximations. Theoretical guarantees demonstrate that approximation errors have a mild impact on regret [191].

Next, we delve into the efficient computation of the integration $\boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1})$ via Monte Carlo integration. We explore a special case of the parameterized transition setting outlined in [126], adapting it to the delayed feedback setting. For simplicity, we consider the scenario where $|\mathcal{A}|$ is finite and define the feature mapping $\boldsymbol{\phi}(s'|s, a)$ as the element-wise product of two feature mappings $\Psi(s)$ and $\Phi(s, a)$, where $|\sum_{s'}[\Psi(s')]_j| \leq C'$, $|[\Phi(s, a)]_j| \leq 1$, and $C' > 0$ is a constant.

In terms of $\boldsymbol{\phi}_{V_m^k}$, Monte Carlo integration can be used to approximate $\boldsymbol{\phi}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1})$ up to ϵ-accuracy from $\hat{\boldsymbol{\phi}}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1})$ with sample complexity of $\tilde{O}(1/\epsilon^2)$. Here, the j-th coordinate of the integration $[\hat{\boldsymbol{\phi}}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1})]_j$ is represented as

$$[\hat{\boldsymbol{\phi}}_{V_m^k}(s_t, a_{t-m}, \vec{\boldsymbol{a}}_{t+1})]_j = \frac{1}{D} \sum_{s' \in \mathcal{S}} [\Psi(s')]_j \cdot [\Phi(s_t, a_{t-m})]_j \cdot \sum_{i=1}^{D} V_m^k(s_{ij}, \vec{\boldsymbol{a}}_{t+1}) \tag{D.9.1}$$

with s_{ij} generated via $s_{ij} \sim [\Psi(\cdot)]_j / \sum_{s' \in \mathcal{S}} [\Psi(s')]_j$, $i = 1, \ldots, D$.

For $\langle \boldsymbol{\phi}(s_{t+1}|s_t, a_{t-m}), \boldsymbol{\theta} \rangle \cdots \langle \boldsymbol{\phi}(s'|s_{t+m}, a), \boldsymbol{\theta} \rangle V_m(s')$, we begin with several definitions and notations. Given the observation state s_t and the action sequence $\vec{\boldsymbol{a}}_t = (a_{t-m}, \cdots, a_{t-1})$ and have

$$\langle \boldsymbol{\phi}^m_{V_m}(s_t, a, \vec{\boldsymbol{a}}_t), \boldsymbol{\theta}^m \rangle = \sum_{(s_{t+1}, \cdots, s_{t+m}, s')} \langle \boldsymbol{\phi}(s_{t+1}|s_t, a_{t-m}), \boldsymbol{\theta} \rangle \cdots \langle \boldsymbol{\phi}(s'|s_{t+m}, a), \boldsymbol{\theta} \rangle V_m(s'), \quad \text{(D.9.2)}$$

where the state trajectory $(s_{t+1}, \cdots, s_{t+m}, s')$ is simulated by probability transition function and derived from s_t by executing the action sequence $(a_{t-m}, \cdots, a_{t-1}, a)$. We can further decompose as

$$\left[\boldsymbol{\phi}^m_{V_m}(s_t, a, \vec{\boldsymbol{a}}_t)\right]_j = \sum_{s' \in \mathcal{S}} V_m(s')[\Psi_m(s')]_j [\Phi_m(s_t, a, \vec{\boldsymbol{a}}_t)]_j, \quad \text{(D.9.3)}$$

where

$$[\Psi_m(s')]_j = \sum_{(s_{t+1}, \cdots, s_{t+m}, s')} [\Psi(s_{t+1})]_j, \cdots, [\Psi(s_{t+m})]_j [\Psi(s')]_j, \quad \text{(D.9.4)}$$

and

$$[\Phi_m(s_t, a, \vec{\boldsymbol{a}}_t)]_j = \sum_{(s_{t+1}, \cdots, s_{t+m}, s')} [\Phi(s_t, a_{t-m})]_j \cdots [\Phi(s_{t+m-1}, a_{t-1})]_j [\Phi(s_{t+m}, a)]_j. \quad \text{(D.9.5)}$$

Hence, Monte Carlo integration can be exploited to evaluate $\sum_{s' \in \mathcal{S}} V_m(s')[\Psi_m(s')]_j$ and achieve a uniform accurate estimation for all (s_t, a) with the action sequence $\vec{\boldsymbol{a}}_t = (a_{t-m}, \cdots, a_{t-1})$. Based on the above definitions, it yields the following proposition, which shows that we can approximate $\boldsymbol{\phi}^m_{V_m}(s_t, a, \vec{\boldsymbol{a}}_t)$ up to ϵ-accuracy by $\hat{\boldsymbol{\phi}}^m_{V_m}(s_t, a, \vec{\boldsymbol{a}}_t)$ (D.9.6) using $D \sim \tilde{O}(1/\epsilon^2)$ samples.

Proposition 4 *Consider V_m defined in Line 7 in Algorithm 7, which can be bounded by $1/(1-\gamma)$. For $i = 1, \cdots, D, j = 1, \cdots, d$, based on equations (D.9.4) and (D.9.5), we denote:*

$$\left[\hat{\boldsymbol{\phi}}^m_{V_m}(s_t, a, \vec{\boldsymbol{a}}_t)\right]_j = \frac{1}{D} \sum_{s' \in \mathcal{S}} [\Psi_m(s')]_j \cdot [\Phi_m(s_t, a, \vec{\boldsymbol{a}}_t)]_j \sum_{i=1}^{D} V_m(s_{ij}), \quad \text{(D.9.6)}$$

where s_{ij} is drawn according to the distribution of $[\Psi_m(\cdot)]_j / \sum_{s' \in \mathcal{S}} [\Psi_m(s')]_j$, then with probability exceeding $1 - \delta$, for the action sequence $\vec{\boldsymbol{a}}_t = (a_{t-m}, \cdots, a_{t-1})$ and $(s_t, a) \in \mathcal{S} \times \mathcal{A}$, we have

$$\left| \left[\boldsymbol{\phi}^m_{V_m}(s_t, a, \vec{\boldsymbol{a}}_t)\right]_j - \left[\hat{\boldsymbol{\phi}}^m_{V_m}(s_t, a, \vec{\boldsymbol{a}}_t)\right]_j \right| \leq \frac{C' \log(\delta^{-1})}{\sqrt{D}(1-\gamma)}.$$

Intuitively, we may require to experience values of Q_m^k over all $(I_m, a) \in (\mathcal{S} \times \mathcal{A}^m) \times \mathcal{A}$ with $I_m = (s_t, a_{t-m}, \cdots, a_{t-1})$, which leads to a $|\mathcal{S}||\mathcal{A}|^m$ space complexity. The complexity can be remarkably diminished via Monte Carlo integration, which is presented as follows. Recall α (5.7), we begin with randomly collecting αdD samples $s_{ij}^t, t \in [\alpha], i = 1, \cdots, D, j = 1, \cdots, d$ via $s_{\cdot,j}^{\cdot} \sim [\Psi_m(\cdot)]_j / \sum_{s' \in \mathcal{S}} [\Psi_m(s')]_j$. For $k \leq N_T$, $t \leq \alpha - 1$, we can compute the value function $V_m^{(t)}(s_{ij}^t, \vec{\boldsymbol{a}}_t)$ based on $V_m^{(t-1)}(s_{ij}^{t-1}, \vec{\boldsymbol{a}}_{t-1})$ through the following induction rule:

$$V_m^{(t)}(s_{ij}^t, \vec{\boldsymbol{a}}_t) = \max_{a \in \mathcal{A}} r(s_{ij}^t, a_{t-m}) + \gamma^m \max_{\boldsymbol{\theta} \in \mathcal{D}_k} \left\langle \hat{\boldsymbol{\phi}}_{V_m^{(t-1)}}^m(s_{ij}^t, a, \vec{\boldsymbol{a}}_{t-1}), \boldsymbol{\theta}^m \right\rangle, \tag{D.9.7}$$

where $\hat{\boldsymbol{\phi}}_{V_m^{(t-1)}}^m(s_{ij}^t, a, \vec{\boldsymbol{a}}_{t-1})$ can be calculated via (D.9.6). The optimization problem (D.9.7) can be termed as a maximization problem constrained on the convex set $\mathcal{D}_k$, which can be efficiently solved by projected gradient methods [192]. Based on the above discussions, denote

$$[\hat{\boldsymbol{\phi}}_{V_m^{(t-1)}}^m(s_t, a, \vec{\boldsymbol{a}}_{t-1})]_j = \frac{1}{D} \sum_{s' \in \mathcal{S}} [\Psi_m(s')]_j \cdot [\Phi_m(s_t, a, \vec{\boldsymbol{a}}_t)]_j \sum_{i=1}^{D} V_m^{(t-1)}(s_{ij}^{t-1}, \vec{\boldsymbol{a}}_{t-1}), \tag{D.9.8}$$

then $Q_m^k(s_t, a, \vec{\boldsymbol{a}}_t)$ can be calculated as

$$Q_m^k(s_t, a, \vec{\boldsymbol{a}}_t) = r(s_t, a_{t-m}) + \gamma \max_{\boldsymbol{\theta} \in \mathcal{D}_k} \left\langle \hat{\boldsymbol{\phi}}_{V_m^{(t-1)}}^m(s_t, a, \vec{\boldsymbol{a}}_{t-1}), \boldsymbol{\theta}^m \right\rangle. \tag{D.9.9}$$

Note that when calculating $Q_m^k(s_t, a, \vec{\boldsymbol{a}}_t)$, only an action sequence $\vec{\boldsymbol{a}}_t = (a_{t-m}, \cdots, a_{t-1})$ and dD value functions $V_m^{(t-1)}(s_{ij}^{t-1}, \vec{\boldsymbol{a}}_{t-1}), i = 1, \cdots, D, j = 1, \cdots, d$ need to be stored, which takes $O(d/\epsilon^2 + m|\mathcal{A}|)$ space. According to Proposition 4, $Q_m^k(s_t, a, \vec{\boldsymbol{a}}_t)$ can be approximated up to ϵ-accuracy with the sample complexity of $\tilde{O}(1/\epsilon^2)$. Furthermore, we study the computational complexity of each k iteration of Algorithm 7. Given s_t and the action sequence the action sequence $\vec{\boldsymbol{a}}_t = (a_{t-m}, \cdots, a_{t-1})$, it takes $\alpha m d|\mathcal{S}|$ time complexity to obtain $\Psi_m(s')$ (D.9.4). Based on $\Psi_m(s')$, it arrives at solving the maximization problem. Recall the definition of α (5.7), it requires $\alpha dD|\mathcal{A}|$ of time complexity to solve the maximization problem $\max_{\boldsymbol{\theta} \in \mathcal{D}_k} \langle \hat{\boldsymbol{\phi}}_{V_m^{(t-1)}}^m(s_t, a, \vec{\boldsymbol{a}}_{t-1}), \boldsymbol{\theta}^m \rangle$ for any $a \in \mathcal{A}$. Next, it needs $\alpha dD|\mathcal{A}|$ time complexity to calculate $V_m^{(t-1)}(s_{ij}^{t-1}, \vec{\boldsymbol{a}}_{t-1}), i = 1, \cdots, D, j = 1, \cdots, d$. After computing $V_m^{(t-1)}(s_{ij}^{t-1}, \vec{\boldsymbol{a}}_{t-1})$, we need $dD|\mathcal{A}|$ time complexity to obtain $Q_m^k(s_t, a, \vec{\boldsymbol{a}}_t)$ for any $(s_t, a) \in \mathcal{S} \times \mathcal{A}$.

APPENDIX E

Proofs for Chapter 6

E.1 Summary of Appendix

We provide the proof of Lemma 4 and Lemma 5 in Appendix E.2 and Appendix E.3, respectively. In Appendix E.2, the proof is based on three lemmas, i.e., Lemma 20-22, and the proofs of these lemmas are provided in Appendix E.4-Appendix E.6, respectively. In Appendix E.5, it involves Lemma 23 which is proven in Appendix E.7. Moreover, the proof of Proposition 1 is provided in Appendix E.8. Some technical lemmas are introduced in Appendix E.9. Some further discussions on Experimental results regret results for finite actions, and reward & loss function examples are provided in the last sections.

E.2 Proof of Lemma 4

In this section, we prove that the ground-truth $\theta_\star$ always remains inside the neighborhood of the averaged iterate $\bar{\theta}_t$ for all times t,

$$\left\|\bar{\theta}_t - \theta_\star\right\|_2 \le \beta_t, \tag{E.2.1}$$

with high probability. Here, the parameter $\beta_t = \tilde{O}(1/\sqrt{t})$ (6.5) implies that the averaged iterate $\bar{\theta}_t$ is closed to the ground-truth step by step. The derivation of the form of β_t (6.5) is provided in the following proof.

In summary, the proof can be achieved via three steps:

1. For $t = 0$, we have $\theta_\star \in \mathcal{B}_t$, because $\|\theta_\star - \bar{\theta}_0\|_2 \le S = \beta_0$ where $\bar{\theta}_0 = 0 \in \mathbb{R}^d$, under

Assumption 2.

2. For $t \geq 1$, we aim to prove $\|\bar{\theta}_t - \theta_\star\|_2 \leq \beta_t$ with high probability.

3. We finish the proof by applying a union bond.

E.2.1 Technical Overview for the Second Case ($t \geq 1$)

Recall that $\widehat{g}_t(\theta_{t-1})$ and $g(\theta_{t-1}) = \mathbb{E}[\widehat{g}_t(\theta_{t-1})]$. At the t-th round, stochastic gradient descent update, i.e., $\theta_t = \theta_{t-1} - \eta_t \widehat{g}_t\left(\theta_{t-1}\right)$ with $t \geq 1$ can be reorganized by

$$\begin{aligned} \theta_t &= \theta_{t-1} - \eta_t \widehat{g}_t\left(\theta_{t-1}\right) \\ &= \theta_{t-1} - \eta_t \widehat{g}_t\left(\theta_{t-1}\right) + \eta_t g(\theta_{t-1}) - \eta_t g(\theta_{t-1}), \end{aligned} \tag{E.2.2}$$

where the gradient is g and the stochastic gradient is $\widehat{g}_t$.

We derive from (E.2.2):

$$\theta_t - \theta_\star = \theta_{t-1} - \theta_\star - \eta_t \widehat{g}_t\left(\theta_{t-1}\right) + \eta_t g(\theta_{t-1}) - \eta_t h(\theta_\star)\Delta_{t-1} + \eta_t h(\theta_\star)\Delta_{t-1} - \eta_t g(\theta_\star), \tag{E.2.3}$$

where

$$h(\theta_\star) = \mathbb{E}[\nabla^2 \ell_i(\theta_\star)], \tag{E.2.4}$$

$$g(\theta_\star) = \mathbb{E}[\nabla \ell_i(\theta_\star)]. \tag{E.2.5}$$

Let $\Delta_t := \theta_t - \theta_\star$ and $A_t := h(\theta_\star)$, (E.2.3) can be represented as

$$\Delta_t = \left(I - \eta_t A_t\right)\Delta_{t-1} + \eta_t z_t - \eta_t b_t, \tag{E.2.6}$$

where

$$z_t = g(\theta_{t-1}) - \widehat{g}_t(\theta_{t-1}), \tag{E.2.7}$$

$$b_t = g(\theta_{t-1}) - g(\theta_\star) - h(\theta_\star) \cdot \Delta_{t-1}. \tag{E.2.8}$$

Here, z_t (E.2.7) can be considered as the gradient noise. Note that z_t is a martingale difference sequence since $\mathbb{E}\left(z_t\right)=0$, which plays a vital role in the proof. To obtain sharp high-probability bounds, we study the detailed structure of the martingale differences and use inequalities that are nearly sharp under our assumptions. The proof is based on the arguments used in [193, 175, 181].

In the following, we sketch the proof of our main results under the step size regime $\eta_t=\eta_0 t^{-\alpha}$ with $\alpha \in (0,1)$. From (E.2.6) we can represent $\left(\Delta_t\right)_{t\geq 1}$ as

$$\Delta_t=\prod_{\ell=1}^{t}\left(I_d-\eta_\ell A_\ell\right)\Delta_0+\sum_{m=1}^{t}\prod_{\ell=m+1}^{t}\left(I_d-\eta_\ell A_\ell\right)\eta_m z_m-\sum_{m=1}^{t}\prod_{\ell=m+1}^{t}\left(I_d-\eta_\ell A_\ell\right)\eta_m b_m \quad \text{(E.2.9)}$$

Let $S_T=T\left(\bar{\theta}_T-\theta\right)=\sum_{t=1}^{T}\Delta_t$ which is further decomposed as $S_T=S_T^\star+S_T^z+S_T^b$, where

$$S_T^\star=\sum_{t=1}^{T}\prod_{\ell=1}^{t}\left(I_d-\eta_\ell A_\ell\right)\Delta_0, \quad \text{(E.2.10)}$$

$$S_T^z=\sum_{t=1}^{T}\sum_{m=1}^{t}\prod_{\ell=m+1}^{t}\left(I_d-\eta_\ell A_\ell\right)\eta_m z_m, \quad \text{(E.2.11)}$$

$$S_T^b=\sum_{t=1}^{T}\sum_{m=1}^{t}\prod_{\ell=m+1}^{t}\left(I_d-\eta_\ell A_\ell\right)\eta_m b_m. \quad \text{(E.2.12)}$$

To bound the target $\omega^\top S_T/T$ for any $\omega\in\mathbb{S}^{d-1}$, we deal with $S_T^\star$, S_T^z and S_T^b separately, which is illustrated in the following three lemmas. Before lemmas, we begin with an assumption that will be used in lemmas and helpful to the proof.

Lemma 20 *Recall the definition of $\Psi_{\chi,\alpha}$ (6.4). For any vector $\omega\in\mathbb{S}^{d-1}$ and $x>0$, we have*

$$\mathbb{P}\left(\left|\omega^\top S_T^\star\right|>x\right)\leq\frac{\left\|\theta_0-\theta_\star\right\|_2\Psi_{\mu\eta_0/2,\alpha}}{x}$$

Proof: *Please refer to Appendix E.4 for details.* □

Lemma 21 *Under Assumptions 2, 3, and 4, for any vector $\omega\in\mathbb{S}^{d-1}$ and $x>0$, we have*

$$\mathbb{P}\left(\left|\omega^\top S_T^z\right|>x\right)\leq 2\exp\left(-\frac{C'_{\chi,\alpha}B\mu^2x^2}{4T\sigma^2}\right),$$

where $C'_{\chi,\alpha}\asymp C_{\chi,\alpha}^{-2}$ and $C_{\chi,\alpha}$ is defined in (6.4) with $\chi=\mu\eta_0/2$ and $\sigma>0$ denotes the standard deviation of noise in Assumption 3.

Proof: *Please refer to Appendix E.5 for details.* □

Lemma 22 *Under Assumptions 2, 3, and 4, for any vector $\omega \in \mathbb{S}^{d-1}$, $\alpha \in (0,1)$ and $x > 0$, we have*

$$\mathbb{P}\left(\omega^\top S_T^b > x + \frac{2C_{\mu\eta_0/2,\alpha} L_H}{\mu^3} \sum_{m=1}^{T} m^{-2\alpha}\right) \leq 2\exp\left\{-\frac{C'_{\chi,\alpha} z^2 \mu^3}{4TL_H^2}\right\},$$

where $C_{\chi,\alpha}$ is defined in (6.4) with $\chi = \mu\eta_0/2$, and $C'_{\chi,\alpha} \asymp C_{\chi,\alpha}^{-2}$.

Proof: *Please refer to Appendix E.6 for details.* □

Combining Lemma 20 - 22, we observe that for any $\omega \in \mathbb{S}^{d-1}$, $x > 0$ and $\alpha = 1/2$, we have

$$\begin{aligned}&\mathbb{P}\left(\left|\omega^\top\left(\bar{\theta}_T - \theta_\star\right)\right| > x + \frac{10C_{\mu\eta_0/2,\alpha} L_H}{\mu^3\sqrt{T}}\right)\\ \leq& \frac{\|\theta_0 - \theta_\star\|_2 \Psi_{\mu\eta_0/2,\alpha}}{Tx} + 2\exp\left(-\frac{C'_{\mu\eta_0/2,\alpha} TB\mu^2 x^2}{4\sigma^2}\right) + 2\exp\left(-\frac{C'_{\mu\eta_0/2,\alpha} Tx^2\mu^3}{4L_H^2}\right)\end{aligned} \tag{E.2.13}$$

For any $x \gtrsim T^{-1/2}$, we have

$$\frac{\|\theta_0 - \theta_\star\|_2 \Psi_{\mu\eta_0/2,\alpha}}{Tx} \leq \exp\left(-\frac{C'_{\mu\eta_0/2,\alpha} Tx^2\mu^3}{4L_H^2}\right) \tag{E.2.14}$$

as long as $\|\theta_0 - \theta_\star\|_2 \leq 2\sqrt{T}\exp\left(-\frac{C'_{\mu\eta_0/2,\alpha}\mu^3}{4L_H^2}\right)/\Psi_{\mu\eta_0/2,\alpha}$, which is a mild condition on θ_0.

Consequently, (E.2.13) implies that

$$\begin{aligned}\mathbb{P}&\left(\left|\omega^\top\left(\bar{\theta}T - \theta\star\right)\right| > x + \frac{10C_{\mu\eta_0/2,\alpha} L_H}{\mu^3\sqrt{T}}\right)\\ &\leq 2\exp\left(-\frac{C'_{\mu\eta_0/2,\alpha} TB\mu^2 x^2}{4\sigma^2}\right)\\ &\quad + 4\exp\left(-\frac{C'_{\mu\eta_0/2,\alpha} Tx^2\mu^3}{4L_H^2}\right).\end{aligned} \tag{E.2.15}$$

By choosing

$$x = O\left(\sqrt{\frac{d}{T}\log\left(\frac{d}{\delta}\right)}\right),$$

we have

$$\frac{x}{\sqrt{d}} \asymp \frac{10C_{\mu\eta_0/2,\alpha}L_H}{\mu^3\sqrt{T}}.$$

Together with $\mathbb{P}\left(\left\|\bar{\theta}_T - \theta_\star\right\|_2 > 2x\right) \leq \sum_{j=1}^d \mathbb{P}\left(\left|\bar{\theta}_{T,j} - \theta_{\star,j}\right| > \frac{x}{\sqrt{d}} + \frac{10C_{\mu\eta_0/2,\alpha}L_H}{\mu^3\sqrt{T}}\right)$, it implies that

$$\mathbb{P}\left(\left\|\bar{\theta}_T - \theta_\star\right\|_2 > 2x\right) \leq 2d\exp\left(-\frac{C'_{\mu\eta_0/2,\alpha}TB\mu^2x^2}{4d\sigma^2}\right) + 4d\exp\left(-\frac{C'_{\mu\eta_0/2,\alpha}Tx^2\mu^3}{4dL_H^2}\right). \quad \text{(E.2.16)}$$

By choosing $B \geq \mu\sigma^2/L_H^2$, we have

$$\exp\left(-\frac{C'_{\mu\eta_0/2,\alpha}TB\mu^2x^2}{4d\sigma^2}\right) \leq \exp\left(-\frac{C'_{\mu\eta_0/2,\alpha}Tx^2\mu^3}{4dL_H^2}\right).$$

Thus, inequality (E.2.16) can be rearranged to

$$\mathbb{P}\left(\left\|\bar{\theta}_T - \theta_\star\right\|_2 > 2x\right) \leq 6d\exp\left(-\frac{C'_{\mu\eta_0/2,\alpha}Tx^2\mu^3}{4dL_H^2}\right) \quad \text{(E.2.17)}$$

By choosing

$$x = C''_{\mu\eta_0/2,\alpha}\sqrt{\frac{4dL_H^2}{T\mu^3}\log\left(\frac{1}{\delta}\right)},$$

we get

$$\left\|\bar{\theta}_T - \theta_\star\right\|_2 \leq C''_{\mu\eta_0/2,\alpha}\sqrt{\frac{16dL_H^2}{T\mu^3}\log\left(\frac{T^2}{\delta}\right)},$$

with probability at least $1 - \delta/T^2$, where $C''_{\chi,\alpha} \asymp C_{\chi,\alpha}$ and $C_{\chi,\alpha}$ is defined in (6.4). Finally, we apply a union bound to arrive at

$$\left\|\bar{\theta}_t - \theta_\star\right\|_2 \leq C''_{\chi,\alpha}\sqrt{\frac{16dL_H^2}{t\mu^3}\log\left(\frac{t}{\delta}\right)}, \quad \text{(E.2.18)}$$

for $\forall\, t \in [N]$ and $C''_{\chi,\alpha} \asymp C_{\chi,\alpha}$ where $\chi = \mu\eta_0/2$, with probability at least $1 - \delta$.

E.3 Proof of Lemma 5

In this section, we prove that $\forall t \in \mathbb{N}$ the set $\mathcal{A}_t$ (4.3) contains the optimal action x^* with high probability.

1. For $t=0$, $\mathcal{A}_0=\mathcal{D}$, thus $x^*\in\mathcal{A}_t$ is satisfied obviously.

2. For $t\geq 1$, let $x^t_{\max}=\arg\max_{a\in\mathcal{A}_{t-1}} r(\bar{\theta}_{t-1},a)$ we have

$$\begin{aligned}
r(\bar{\theta}_{t-1},x^t_{\max})-r(\bar{\theta}_{t-1},x^*) &= r(\bar{\theta}_{t-1},x^t_{\max})-r(\theta_\star,x^*)+r(\theta_\star,x^*)-r(\bar{\theta}_{t-1},x^*)\\
&\leq r(\bar{\theta}_{t-1},x^t_{\max})-r(\theta_\star,x^t_{\max})+r(\theta_\star,x^*)-r(\bar{\theta}_{t-1},x^*)\\
&\leq |r(\bar{\theta}_{t-1},x^t_{\max})-r(\theta_\star,x^t_{\max})|+|r(\theta_\star,x^*)-r(\bar{\theta}_{t-1},x^*)|\\
&\overset{(a)}{\leq} \left\|\bar{\theta}_{t-1}-\theta_\star\right\|_2\cdot L+\left\|\bar{\theta}_{t-1}-\theta_\star\right\|_2\cdot L\\
&= 2L\cdot\left\|\bar{\theta}_{t-1}-\theta_\star\right\|_2\\
&\overset{(b)}{\leq} 2L\beta_{t-1} \qquad\qquad \text{(E.3.1)}
\end{aligned}$$

   Here, step (a) comes from the first condition in Assumption 4 and step (b) is based on (6.9) derived from Lemma 4. From (E.3.1), we know that $x^*\in\mathcal{A}_t$.

By combining the above two cases, we complete the proof.

E.4 Proof of Lemma 20

In this section, we prove the tail bound on the quantity $|\omega^\top S_T^\star|$ with $S_T^\star$ (E.2.10). First, we can bound $\|I_d-\eta_\ell A_\ell\|$ by

$$\begin{aligned}
\|I_d-\eta_\ell A_\ell\| &\overset{(a)}{\leq} 1-\mu\eta_\ell+\frac{\mu}{2}\eta_\ell\\
&= 1-\frac{\mu}{2}\eta_\ell, \qquad\qquad \text{(E.4.1)}
\end{aligned}$$

for each $\ell\geq 1$, where step (a) comes from Assumption 4. Consequently, by the triangle inequality, we have

$$\left|\omega^\top S_T^\star\right|\leq\|\theta_0-\theta_\star\|_2\sum_{t=1}^{T}\prod_{\ell=1}^{t}\left(1-\frac{\mu}{2}\eta_\ell\right)\leq\|\theta_0-\theta_\star\|_2\,\Psi_{\frac{\mu}{2}\eta_0,\alpha},$$

where $\Psi_{\chi,\alpha}$ is defined in (6.4). Then, based on Markov's inequality, it arrives at

$$\mathbb{P}\left(\left|\omega^\top S_T^\star\right|>x\right)\leq\frac{\|\theta_0-\theta_\star\|_2\,\Psi_{\mu\eta_0/2,\alpha}}{x}.$$

E.5 Proof of Lemma 21

In this section, we prove the tail bound on the quantity $|\omega^\top S_T^z|$ with S_T^z (E.2.11).

Recall that $S_T^z = \sum_{t=1}^T \sum_{m=1}^t \prod_{\ell=m+1}^t (I_d - \eta_\ell A_\ell)\, \eta_m z_m$, where $z_m = g(\theta_{m-1}) - \widehat{g}_m(\theta_{m-1})$. For any $\omega \in \mathbb{S}^{d-1}$,

$$\omega^\top S_T^z = \frac{1}{B}\sum_{m=1}^T \sum_{i=(m-1)B+1}^{mB} \eta_m \omega^\top H_m \left(\nabla f(\theta_{t-1}) - \nabla \ell_i(\theta_{t-1})\right),$$

$$\text{where } H_m = \sum_{t=m+1}^T \prod_{\ell=m+1}^t (I_d - \eta_\ell A_\ell).$$

Thus, $\omega^\top S_T^z$ is a sum of independent zero-mean random variables conditional on

$$\mathcal{F}_{x,n} = \mathcal{S}\{x_1, x_2, \dots, x_n\}$$

where $n = TB$. We denote $\mathcal{S}$ as σ-algebra to avoid confusion of noise-related parameter σ. For $x > 0$, by Chernoff-Hoeffding inequality in Lemma 24, we have

$$\mathbb{P}\left(\left|\omega^\top S_T^z\right| > x \mid \mathcal{F}_{x,n}\right) \le 2\exp\left(-\frac{B^2 x^2}{D_T}\right), \tag{E.5.1}$$

where

$$\begin{aligned} D_T &= \sigma^2 \sum_{m=1}^T \sum_{i=(m-1)B+1}^{mB} \eta_m^2 \omega^\top H_m H_m^\top \omega \\ &= B\sigma^2 \sum_{m=1}^T \eta_m^2 \omega^\top H_m H_m^\top \omega =: B\sigma^2 \sum_{m=1}^T \eta_m^2 \xi_m. \end{aligned} \tag{E.5.2}$$

To complete the proof, we need to bound the D_T in (E.5.1) conditional on

$$\mathcal{F}_{x,n} = \mathcal{S}\{x_1, x_2, \dots, x_n\},$$

which is presented in the following lemma.

Lemma 23 *For $z > 0$, we have*

$$\mathbb{P}\left(|D_T - \mathbb{E}(D_T)| > z\right) \le 2\exp\left\{-\frac{Cz^2\mu^4}{TB^2}\right\}$$

where $C > 0$ depends on the convexity parameter μ, α, and the initial step size η_0.

***Proof**: Please refer to Appendix E.7 for details.* □

Thus, by Lemma 23, we have

$$\mathbb{P}\left(D_T > \mathbb{E}\left(D_T\right) + \frac{x^2}{\log x}\right) \leq \exp\left\{-\frac{Cx^4\mu^4}{(\log x)^2 T B^2}\right\}.$$

By a similar argument as that of (E.7.17), we get $\mathbb{E}\left(D_T\right) \leq C^2_{\mu\eta_0/2,\alpha} T B\sigma^2/\mu^2$. Hence, we have

$$\mathbb{P}\left(\left|\omega^\top S^z_T\right| > x\right) \leq 2\exp\left(\frac{-Bx^2\mu^2}{C^2_{\mu\eta_0/2,\alpha} T\sigma^2 + x^2/\log x}\right) \leq 2\exp\left(-\frac{C'_{\chi,\alpha} B x^2\mu^2}{4T\sigma^2}\right),$$

where $C'_{\chi,\alpha} \asymp C^{-2}_{\chi,\alpha}$ is positive constant depending on with $\chi = \mu\eta_0/2$ and α.

E.6 Proof of Lemma 22

In this section, we prove the tail bound on the quantity $|\omega^\top S^b_T|$ with S^b_T (E.2.12).

Recall that $S^b_T = \sum_{t=1}^T \sum_{m=1}^t \prod_{\ell=m+1}^t \left(I_d - \eta_\ell A_\ell\right) \eta_m b_m$ where $b_m = g(\theta_{t-1}) - g(\theta_\star) - h(\theta_\star) \cdot \Delta_{m-1}$ with $h(\theta_\star) = \mathbb{E}[\nabla^2 \ell_i(\theta_\star)]$. For any $\omega \in \mathbb{S}^{d-1}$,

$$|\omega^\top S^b_T| = \left|\sum_{m=1}^T \eta_m \omega^\top H_m b_m\right| \overset{(a)}{\leq} \left|\eta_m \omega^\top H_m \omega \cdot \frac{L_H}{2}\left\|\Delta_{m-1}\right\|_2\right|, \tag{E.6.1}$$

where $H_m = \sum_{t=m+1}^T \prod_{\ell=m+1}^t \left(I_d - \eta_\ell A_\ell\right)$, and step (a) comes from (6.1) in Assumption 4. Thus, we turn to bound

$$D_b := \sum_{m=1}^T \eta_m \omega^\top H_m \omega_m \cdot \frac{L_H}{2}\left\|\Delta_{m-1}\right\|_2^2 := \sum_{m=1}^T \eta_m \zeta_m, \tag{E.6.2}$$

Thus, D_b is a sum of random variables. We need to address the dependence between $\{A_\ell\}$ in H_m. To achieve this, we utilize the same argument as that of Lemma 23 and introduce the K-approximation of D_b as

$$D_{b,K} = \sum_{m=1}^T \eta_m \mathbb{E}\left(\xi_m \mid \mathcal{G}_{A,m+K}\right), \tag{E.6.3}$$

where $\mathcal{G}_{A,k} = \mathcal{S}\{A_1, A_2, \ldots, A_k\}$ for $\forall k \geq 1$. Note that $D_{b,K}$ is the sum of independent random variables, and $\mathbb{E}(D_b) = \mathbb{E}(D_{b,K})$. Thus, $|D_b - \mathbb{E}(D_b)|$ can be bounded by

$$\mathbb{P}(|D_b - \mathbb{E}(D_b)| > z) \leq \underbrace{\mathbb{P}(|D_b - D_{b,K}| > z/2)}_{J_1} + \underbrace{\mathbb{P}(|D_{b,K} - \mathbb{E}(D_{b,K})| > z/2)}_{J_2}. \tag{E.6.4}$$

- **Proof of** J_1 Recall the projection operator defined in (E.7.6)

$$\mathcal{P}_{A,k}(\cdot) = \mathbb{E}(\cdot \mid \mathcal{G}_{A,k}) - \mathbb{E}(\cdot \mid \mathcal{G}_{A,k-1}),$$

and $H_{m/k}$ defined in (E.7.7), i.e.,

$$H_{m/k} = \mathcal{H}(A_{m+1}, A_{m+2}, \ldots, A_{k-1}, A_k^\star, A_{k+1}, \ldots, A_T),$$

for $\forall m+1 \leq k \leq T$.

From $\zeta_m := \omega^\top H_m \omega \frac{L_H}{2} \cdot \|\Delta_{m-1}\|_2^2$, we have

$$\begin{aligned}
&\mathcal{P}_{A,k}(\zeta_m) \\
=&\frac{L_H}{2}\big[\mathbb{E}\left(\omega^\top H_m \omega \|\Delta_{m-1}\|_2^2 \mid \mathcal{G}_{A,k}\right) \\
&- \mathbb{E}\left(\omega^\top H_m \omega \|\Delta_{m-1}\|_2^2 \mid \mathcal{G}_{A,k-1}\right)\big] \\
=&\frac{L_H}{2}\big[\mathbb{E}\left(\omega^\top H_m \omega \|\Delta_{m-1}\|_2^2 \mid \mathcal{S}\{A_1, A_2, \ldots, A_k\}\right) \\
&- \mathbb{E}\left(\omega^\top H_m \omega \|\Delta_{m-1}\|_2^2 \mid \mathcal{S}\{A_1, A_2, \ldots, A_{k-1}\}\right)\big] \\
=&\frac{L_H}{2}\big[\mathbb{E}_{A_{k+1},\ldots,A_T}\left(\omega^\top H_m \omega \|\Delta_{m-1}\|_2^2 \mid A_1, \ldots, A_k\right) \\
&- \mathbb{E}_{A_{k+1},\ldots,A_T}\left[\mathbb{E}_{A_k}\left(\omega^\top H_m \omega \|\Delta_{m-1}\|_2^2 \mid A_1, \ldots, A_{k-1}\right)\right]\big] \\
=&\frac{L_H}{2}\left(\mathbb{E}_{A_k^\star}\left[\mathbb{E}_{A_{k+1},\ldots,A_T}\left(\omega^\top\left(H_m - H_{m/k}\right)\omega \|\Delta_{m-1}\|_2^2 \mid A_1, \ldots, A_k\right)\right]\right) \\
:=&\frac{L_H}{2} \cdot \mathbb{E}\left[\omega^\top\left(H_m - H_{m/k}\right)\omega \cdot \|\Delta_{m-1}\|_2^2\right]
\end{aligned} \tag{E.6.5}$$

Consequently, we can bound $|\mathcal{P}_{A,k}(\zeta_m)|$ by

$$\begin{aligned}
|\mathcal{P}_{A,k}(\zeta_m)| &\leq \frac{L_H}{2}\mathbb{E}\left[\left|H_m - H_{m/k}\right|\right] \cdot \mathbb{E}[\|\Delta_{m-1}\|_2^2] \\
&\overset{(a)}{\leq} \frac{SL_H\eta k}{4} C\eta_{m-1}\int_k^\infty \exp\left(-\frac{\mu\eta_0}{2}\int_{m+1}^z x^{-\alpha}dx\right)dz \\
&\overset{(b)}{\leq} \frac{SL_H C\eta_m^2}{4}\Psi_{\mu\eta_0/2,\alpha}
\end{aligned} \tag{E.6.6}$$

which step (a) comes from (E.7.11), Assumption 2 and $\mathbb{E}[\|\theta_t - \theta_\star\|_2^2] \le \frac{C}{\mu}\eta_t$ with $C > 0$ in Lemma 25. Moreover, step (b) is based on (6.4).

Consequently, we have

$$\begin{aligned}
|D_b - D_{b,K}| &= \left| \sum_{m=1}^{T} \eta_m \mathbb{E}\left(\zeta_m \mid \mathcal{G}_{A,T}\right) - \sum_{m=1}^{T} \eta_m \mathbb{E}\left(\zeta_m \mid \mathcal{G}_{A,m+K}\right) \right| \\
&= \left| \sum_{m=1}^{T} \eta_m \sum_{h=K+1}^{T-h} \mathcal{P}_{A,m+h}\left(\zeta_m\right) \right| \\
&\le \sum_{h=K+1}^{T-1} \left\{ \sum_{m=1}^{T-h} \eta_m \left\| \mathcal{P}_{A,m+h}\left(\zeta_m\right) \right\|^2 \right\}^{1/2} \\
&\le \frac{S L_H C T^{1+\alpha}}{(\mu/2)^2 \cdot K^\alpha} \exp\left(-\frac{\mu \eta_0 K}{2^{\alpha+1} T^\alpha} \right) \\
&\overset{(a)}{\le} C_{\mu\eta_0/2,\alpha} \cdot T^{-\frac{1}{2}},
\end{aligned} \tag{E.6.7}$$

where step (a) can be derived by choosing

$$K \asymp \left(\frac{4 S L_H}{\mu^2 \eta_0}\right) \log\left(S T^{1+\alpha}/\mu\right).$$

Then, based on Markov's inequality, it arrives at

$$\mathbb{P}\left(|D_b - D_{b,K}| > z/2\right) \le \frac{2 C_{\mu\eta_0/2,\alpha}}{z T^{1/2}}. \tag{E.6.8}$$

- **Proof of** J_2

Based on Chernoff-Hoeffding inequality in Lemma 24, we have

$$\mathbb{P}\left(|D_{b,K} - \mathbb{E}\left(D_{b,K}\right)| > z/2\right) \le 2 \exp\left(\frac{-2(z/2)^2}{\sum_{m=1}^{T} \left(\eta_m \mathbb{E} \left| \omega^\top H_m \omega \frac{L_H}{2} \left\| \Delta_{m-1} \right\|_2^2 \right| \right)^2} \right) \tag{E.6.9}$$

Based on (E.7.12) and (E.6.3) and Lemma 25, we can bound

$$\sum_{m=1}^{T} \eta_m^2 \mathbb{E} \left| \omega^\top H_m \omega \frac{L_H}{2} \left\| \Delta_{m-1} \right\|_2^2 \right|^2$$

by

$$\sum_{m=1}^{T} \eta_m^2 \mathbb{E}\left|\omega^\top H_m \omega \frac{L_H}{2}\|\Delta_{m-1}\|_2^2\right|^2 \le \sum_{m=1}^{T} \eta_m^2 \frac{L_H^2}{4} \cdot \mathbb{E}(\|H_m\|^2)\mathbb{E}[\|\Delta_{m-1}\|_2^2]$$
$$\le \sum_{m=1}^{T} \eta_m^2 \frac{L_H^2}{4} \cdot \mathbb{E}(\|H_m\|^2)\frac{C}{\mu}\eta_{m-1}$$
$$\le \frac{C_{\mu\eta_0/2,\alpha}^2 T L_H^2}{4(\mu/2)^3} = 2C_{\mu\eta_0/2,\alpha}^2 T L_H^2/\mu^3. \qquad \text{(E.6.10)}$$

Combining (E.6.9) and (E.6.10), we conclude

$$\mathbb{P}\left(|D_{b,K} - \mathbb{E}\left(D_{b,K}\right)| > z/2\right) \le 2\exp\left\{-\frac{C'_{\chi,\alpha} z^2 \mu^3}{4TL_H^2}\right\}, \qquad \text{(E.6.11)}$$

where $C'_{\chi,\alpha} \asymp C_{\chi,\alpha}^{-2}$ are positive constants depending on $\chi = \mu\eta_0/2$ and α.

Together with (E.6.8) and (E.6.11), we complete the proof via

$$\mathbb{P}\left(|D_b - \mathbb{E}\left(D_b\right)| > z\right) \le \frac{2C_{\mu\eta_0/2,\alpha}}{zT^{1/2}} + 2\exp\left\{-\frac{C'_{\mu\eta_0/2,\alpha} z^2 \mu^3}{4TL_H^2}\right\} \le 4\exp\left\{-\frac{C'_{\mu\eta_0/2,\alpha} z^2 \mu^3}{4TL_H^2}\right\}, \qquad \text{(E.6.12)}$$

where the last inequality can be satisfied. Hence, we get

$$\mathbb{P}\left(D_b > \mathbb{E}\left(D_b\right) + z\right) \le 2\exp\left\{-\frac{C'_{\mu\eta_0/2,\alpha} z^2 \mu^3}{4TL_H^2}\right\}.$$

To complete the proof, we remain to bound $\mathbb{E}\left(D_b\right)$. Based on Lemma 25, we have

$$\mathbb{E}\left(D_b\right) = \sum_{m=1}^{T} \eta_m \mathbb{E}\left(\omega^\top H_m \omega \frac{L_H}{2}\|\Delta_{m-1}\|_2^2\right) \le \sum_{m=1}^{T} \frac{L_H}{2}\eta_m \cdot \mathbb{E}\left(\|H_m\|\right)\mathbb{E}(\|\Delta_{m-1}\|_2^2)$$
$$\le \frac{L_H}{2}\frac{C}{\mu} \cdot C_{\mu\eta_0/2,\alpha} \cdot \sum_{m=1}^{T} (m^{-\alpha} \cdot \frac{2}{\mu})^2$$
$$= \frac{2C_{\mu\eta_0/2,\alpha} L_H}{\mu^3} \sum_{m=1}^{T} m^{-2\alpha}. \qquad \text{(E.6.13)}$$

E.7 Proof of Lemma 23

Recall that

$$D_T = B\sigma^2 \sum_{m=1}^{T} \eta_m^2 \omega^\top H_m H_m^\top \omega =: B\sigma^2 \sum_{m=1}^{T} \eta_m^2 \xi_m, \tag{E.7.1}$$

$$\text{where } H_m = \sum_{t=m+1}^{T} \prod_{\ell=m+1}^{t} (I_d - \eta_\ell A_\ell), \quad \text{and} \tag{E.7.2}$$

$$A_\ell = \frac{1}{B} \sum_{i=(\ell-1)B+1}^{\ell B} \nabla^2 \ell_i(\theta_\star). \tag{E.7.3}$$

To bound (E.7.1), we need to address the dependence between $\{A_\ell\}$ in H_m (E.7.2). To achieve this, we introduce the G-approximation of D_T as

$$D_{T,G} = B\sigma^2 \sum_{m=1}^{T} \eta_m^2 \mathbb{E}\left(\xi_m \mid \mathcal{G}_{A,m+G}\right), \tag{E.7.4}$$

where $\mathcal{G}_{A,k} = \mathcal{S}\left\{A_1, A_2, \ldots, A_k\right\}$ for $\forall k \geq 1$. Note that $D_{T,G}$ is the sum of independent random variables, and $\mathbb{E}\left(D_T\right) = \mathbb{E}\left(D_{T,G}\right)$. Thus, $|D_T - \mathbb{E}\left(D_T\right)|$ can be alternatively bounded by

$$\mathbb{P}\left(|D_T - \mathbb{E}\left(D_T\right)| > z\right) \leq \underbrace{\mathbb{P}\left(|D_T - D_{T,G}| > z/2\right)}_{J_1} + \underbrace{\mathbb{P}\left(|D_{T,G} - \mathbb{E}\left(D_{T,G}\right)| > z/2\right)}_{J_2}. \tag{E.7.5}$$

- **Proof of** J_1

  First, we define the projection operator

$$\mathcal{P}_{A,k}(\cdot) = \mathbb{E}\left(\cdot \mid \mathcal{G}_{A,k}\right) - \mathbb{E}\left(\cdot \mid \mathcal{G}_{A,k-1}\right), \tag{E.7.6}$$

  and denote $H_m = \mathcal{H}\left(A_{m+1}, A_{m+2}, \ldots, A_T\right)$. For $\forall m+1 \leq k \leq T$, define

$$H_{m/k} = \mathcal{H}\left(A_{m+1}, A_{m+2}, \ldots, A_{k-1}, A_k^\star, A_{k+1}, \ldots, A_T\right), \tag{E.7.7}$$

  where $A_t^\star$ are i.i.d. random matrix with $A_t^\star$ having the same distribution as A_t for $t \geq 1$. Note that

$$H_m - H_{m/k} = \sum_{j=k}^{T} \prod_{\ell=m+1}^{k-1} (I_d - \eta_\ell A_\ell) \prod_{\ell=k+1}^{j} (I_d - \eta_\ell A_\ell)\, \eta_k \left(A_k - A_k^\star\right). \tag{E.7.8}$$

From $\xi_m := \omega^\top H_m H_m^\top \omega$, (E.7.6) and (E.7.8), we have

$$\begin{aligned}
\mathcal{P}_{A,k}\left(\xi_m\right) &= \mathbb{E}\left(\omega^\top H_m H_m^\top \omega \mid \mathcal{G}_{A,k}\right) - \mathbb{E}\left(\omega^\top H_m H_m^\top \omega \mid \mathcal{G}_{A,k-1}\right) \\
&= \mathbb{E}\left(\omega^\top H_m H_m^\top \omega \mid \mathcal{S}\left\{A_1, A_2, \ldots, A_k\right\}\right) \\
&\quad - \mathbb{E}\left(\omega^\top H_m H_m^\top \omega \mid \mathcal{S}\left\{A_1, A_2, \ldots, A_{k-1}\right\}\right) \\
&= \mathbb{E}_{A_{k+1},\ldots,A_T}\left(\omega^\top H_m H_m^\top \omega \mid A_1, \ldots, A_k\right) \\
&\quad - \mathbb{E}_{A_{k+1},\ldots,A_T}\left[\mathbb{E}_{A_k}\left(\omega^\top H_m H_m^\top \omega \mid A_1, \ldots, A_{k-1}\right)\right] \\
&= \mathbb{E}_{A_k^\star}\left[\mathbb{E}_{A_{k+1},\ldots,A_T}\left(\omega^\top H_m H_m^\top \omega - \omega^\top H_{m/k} H_{m/k}^\top \omega \mid A_1, \ldots, A_k\right)\right] \\
&= \mathbb{E}_{A_k^\star}\left[\mathbb{E}_{A_{k+1},\ldots,A_T}\left[\left(\omega^\top H_m^{1/2} - \omega^\top H_{m/k}^{1/2}\right)\right.\right. \\
&\quad \left.\left.\left(\omega^\top H_m^{1/2} + \omega^\top H_{m/k}^{1/2}\right)\middle| A_1, \ldots, A_k\right]\right] \\
&= \mathbb{E}_{A_k^\star}\left[\mathbb{E}_{A_{k+1},\ldots,A_T}\left[\omega^\top\left(H_m + H_{m/k}\right)\left(H_m - H_{m/k}\right)^\top \omega \middle| A_1, \ldots, A_k\right]\right] \\
&:= \mathbb{E}\left[\omega^\top\left(H_m + H_{m/k}\right)\left(H_m - H_{m/k}\right)^\top \omega\right] \qquad \text{(E.7.9)}
\end{aligned}$$

Consequently, we can bound $|\mathcal{P}_{A,k}\left(\xi_m\right)|$ by

$$\begin{aligned}
|\mathcal{P}_{A,k}\left(\xi_m\right)| &\leq 2\mathbb{E}\left[\left\|H_m - H_{m/k}\right\| \cdot \left\|H_m\right\|\right] \\
&\overset{(a)}{\leq} \mu\eta_k \cdot \mathbb{E}\left(\left\|H_m\right\|\right) \int_k^\infty \exp\left(-\frac{\mu\eta_0}{2}\int_{m+1}^z x^{-\alpha} dx\right) dz. \qquad \text{(E.7.10)}
\end{aligned}$$

Here, step (a) is derived from the fact that

$$\begin{aligned}
\mathbb{E}\left(\left\|H_m - H_{m/k}\right\|\right) &\leq \sum_{j=k}^T \eta_k \left\|A_k - \Sigma_k\right\| \prod_{\ell=m+1}^j \left(1 - \frac{\mu}{2}\eta_\ell\right) \\
&\leq \frac{\mu}{2}\eta_k \int_k^\infty \exp\left(-\frac{\mu}{2}\eta_0 \int_{m+1}^z x^{-\alpha} dx\right) dz, \qquad \text{(E.7.11)}
\end{aligned}$$

where the first inequality is based on (E.4.1). We further bound $\mathbb{E}\left(\left\|H_m\right\|\right)$ by

$$\begin{aligned}
\mathbb{E}\left(\left\|H_m\right\|\right) &= \mathbb{E}\left(\left\|\sum_{t=m+1}^T \prod_{\ell=m+1}^t \left(I_d - \eta_\ell A_\ell\right)\right\|\right) \\
&\leq \int_{m+1}^\infty \exp\left(-\frac{\mu}{2}\eta_0 \int_{m+1}^z x^{-\alpha} dx\right) dz \overset{(a)}{\leq} \Psi_{\mu\eta_0/2,\alpha} \overset{(b)}{\leq} C_{\mu\eta_0/2,\alpha} \qquad \text{(E.7.12)}
\end{aligned}$$

where steps (a) and (b) are based on (6.4). Combining (E.7.14), (E.7.10) and (E.7.12), we have

$$\begin{aligned}|D_T - D_{T,G}| &= \left| B\sigma^2 \sum_{m=1}^{T} \eta_m^2 \mathbb{E}\left(\xi_m \mid \mathcal{G}_{A,T}\right) - B\sigma^2 \sum_{m=1}^{T} \eta_m^2 \mathbb{E}\left(\xi_m \mid \mathcal{G}_{A,m+G}\right) \right| \\ &= \left| B\sigma^2 \sum_{m=1}^{T} \eta_m^2 \sum_{h=G+1}^{T-h} \mathcal{P}_{A,m+h}\left(\xi_m\right) \right| \\ &\leq B\sigma^2 \sum_{h=G+1}^{T-1} \left\{ \sum_{m=1}^{T-h} \eta_m^2 \left\|\mathcal{P}_{A,m+h}\left(\xi_m\right)\right\|^2 \right\}^{1/2} \\ &\overset{(a)}{\leq} \frac{4C_{\mu\eta_0/2,\alpha} B\sigma^2 T^{1+\alpha}}{\mu^2 G^{\alpha}} \exp\left(-\frac{\mu\eta_0 G}{2^{\alpha+1}T^{\alpha}}\right) \\ &\overset{(b)}{\leq} C_{\mu\eta_0/2,\alpha} T^{-1/2}, \end{aligned} \tag{E.7.13}$$

where step (a) comes from (E.7.10), (E.7.11), and (E.7.12), and step (b) is derived by choosing G as

$$G \asymp \frac{2T^{\alpha}}{\lambda\eta_0} \log\left(\frac{4B\sigma^2 T^{1+\alpha}}{\mu^2}\right) \tag{E.7.14}$$

Then, based on Markov's inequality, it arrives at

$$\mathbb{P}\left(|D_T - D_{T,G}| > z/2\right) \leq \frac{2C_{\mu\eta_0/2,\alpha}}{zT^{1/2}}. \tag{E.7.15}$$

- **Proof of** J_2

Based on Chernoff-Hoeffding inequality in Lemma 24, we have

$$\mathbb{P}\left(|D_{T,G} - \mathbb{E}\left(D_{T,G}\right)| > z/2\right) \leq 2\exp\left(\frac{-2(z/2)^2}{(B\sigma^2)^2 \sum_{m=1}^{T} \left(\eta_m^2 \mathbb{E}\left|\omega^\top H_m\right|^2\right)^2}\right). \tag{E.7.16}$$

Based on (E.7.4) and (E.7.12), we can bound $\sum_{m=1}^{T} \eta_m^4 \mathbb{E}\left|\omega^\top H_m\right|^4$ by

$$\sum_{m=1}^{T} \eta_m^4 \mathbb{E}\left|\omega^\top H_m\right|^4 \leq \sum_{m=1}^{T} \eta_m^4 \cdot \mathbb{E}(\|H_m\|^4) \leq \frac{16C_{\mu\eta_0/2,\alpha}^4 T}{\mu^4}. \tag{E.7.17}$$

Combining (E.7.16) and (E.7.17), we conclude

$$\mathbb{P}\left(\left|D_{T,G}-\mathbb{E}\left(D_{T,G}\right)\right|>z/2\right)\leq 2\exp\left\{-\frac{\tilde{C}_{\chi,\alpha}z^2\mu^4}{TB^2}\right\}, \tag{E.7.18}$$

where $\tilde{C}_{\chi,\alpha}>0$ depends on $\chi=\mu\eta_0/2$ and α.

Together with (E.7.15) and (E.7.18), we complete the proof via

$$\mathbb{P}\left(\left|D_T-\mathbb{E}\left(D_T\right)\right|>z\right)\leq\frac{2C_{\mu\eta_0/2,\alpha}}{zT^{1/2}}+2\exp\left\{-\frac{\tilde{C}_{\chi,\alpha}z^2\mu^4}{TB^2}\right\}\leq 2\exp\left\{-\frac{Cz^2\mu^4}{TB^2}\right\}, \tag{E.7.19}$$

where $C>0$ depends on μ, α, and the initial step size η_0.

E.8 Proof of Proposition 1

In this section, we derive the regret bound for the proposed algorithm.

1. At the first round, for all $x_i\in\mathcal{A}_t$, regret $\mathrm{reg}_i=r(\theta_\star,x^*)-r(\theta_\star,x_i)\leq 2SL$.
2. At t-th ($t\geq 2$) round, for all $x_i\in\mathcal{A}_t$, regret $\mathrm{reg}_i=r(\theta_\star,x^*)-r(\theta_\star,x_i)$ can be bounded by

$$\begin{aligned}
\mathrm{reg}_i &= r(\theta_\star,x^*)-r(\theta_\star,x_i)-L\beta_{t-1}-L\beta_{t-1}+2L\beta_{t-1}\\
&= r(\theta_\star,x^*)-L\beta_{t-1}-(r(\theta_\star,x_i)+L\beta_{t-1})+2L\beta_{t-1}\\
&\overset{(a)}{\leq} r(\theta_\star,x^*)-r(\bar{\theta}_{t-1},x_i)+2L\beta_{t-1}\\
&\overset{(b)}{\leq} r(\bar{\theta}_{t-1},x^t_{\max})-r(\theta_{t-1},x_i)+2L\beta_{t-1}\\
&\overset{(c)}{\leq} 2L\beta_{t-1}+2L\beta_{t-1}\\
&\leq 4L\beta_{t-1}
\end{aligned} \tag{E.8.1}$$

Here, step (a) comes from $|r(\bar{\theta}_{t-1},x)-r(\theta_\star,x)|\leq L\beta_{t-1}$ based on (6.9)

$$\left\|\bar{\theta}_t-\theta_\star\right\|_2\leq\beta_t, \tag{E.8.2}$$

and the first condition in Assumption 4, step (b) is due to $x^t_{\max} = \arg\max_{a\in\mathcal{A}_t} r(\bar{\theta}_{t-1}, a)$, and step (c) is based on the fact $x_i \in \mathcal{A}_t$.

Thus, by selecting β_t (6.5), with probability at least $1-\delta$, we get

$$\begin{aligned}\mathrm{Reg}(TB) = \sum_{t=2}^{T}\sum_{i=(t-1)B+1}^{tB} \mathrm{reg}_i \leq 4L\beta_{t-1} &= 4BLC''_{\chi,\alpha}\sqrt{\frac{16dL_H^2}{\mu^3}\log\left(\frac{TB}{\delta}\right)}\cdot\sum_{t=2}^{T}\frac{1}{\sqrt{t}}\\ &\overset{(a)}{\leq} 32\sqrt{B}LC_{\chi,\alpha}\sqrt{\frac{dL_H^2TB}{\mu^3}\log\left(\frac{TB}{\delta}\right)}\\ &\overset{(b)}{\leq} 32\cdot\sigma C''_{\chi,\alpha}dL\sqrt{\frac{TB}{\mu^2}\log\left(\frac{TB}{\delta}\right)}. \end{aligned} \tag{E.8.3}$$

Here, step (a) comes from the fact that $\sum_{t=2}^{T}\frac{1}{\sqrt{t}} \leq 2\sqrt{T}$, and step (b) is derived by setting the batch-size as $B = \mu\sigma^2 d/L_H^2$.

Combining the above two cases, we finally get

$$\begin{aligned}\mathrm{Reg}(n) &= B\cdot 2SL + \sum_{t=2}^{T}\sum_{i=(t-1)B+1}^{tB}\mathrm{reg}_i\\ &\leq \frac{2\mu\sigma^2SLd}{L_H^2} + 32\cdot\sigma C''_{\chi,\alpha}dL\sqrt{\frac{n}{\mu^2}\log\left(\frac{n}{\delta}\right)}\end{aligned} \tag{E.8.4}$$

where for $n = TB$ and $B = \mu\sigma^2 d/L_H^2$.

E.9 Technical Lemmas

In this section, we summarize several technical lemmas that support the proof.

Lemma 24 (Chernoff-Hoeffding Inequality) *Consider a set of r independent random variables $\{x_1,\dots,x_r\}$. If we know $a_i \leq x_i \leq b_i$, then let $v_i = b_i - a_i$. Let $M = \sum_{i=1}^{r} x_i$. Then for any $\alpha \in (0, 1/2)$*

$$\mathbb{P}[|M - \boldsymbol{E}[M]| > \alpha] \leq 2\exp\left(\frac{-2\alpha^2}{\sum_{i=1}^{r} v_i^2}\right), \textit{ and}$$

$$\mathbb{P}[M - \boldsymbol{E}[M] > \alpha] \leq \exp\left(\frac{-2\alpha^2}{\sum_{i=1}^{r} v_i^2}\right).$$

Lemma 25 *For some $C > 0$, we have $\mathbb{E}[\|\theta_t - \theta_\star\|_2^2] \le \frac{C}{\mu}\eta_t$.*

Proof: *For $i \in [(t-1)B+1, tB]$, let $\nabla\ell_i(\theta_t)$ denote the sub-optimality $\ell_i(\theta_t) - \mathbb{E}[\ell_i(\theta_\star)]$. According to the second-order Taylor approximation, we have*

$$\begin{aligned}\ell_i(\theta_t) =& \ell_i(\theta_{t-1} - \eta_t \widehat{g}_t(\theta_{t-1})) \\ =& \ell_i(\theta_{t-1}) - \eta_t \nabla\ell_i(\theta_{t-1})^\top \widehat{g}_t(\theta_{t-1}) + \frac{\eta_t^2}{2}\widehat{g}_t(\theta_{t-1})^\top \nabla^2\ell_i(\theta_{t-1})\widehat{g}_t(\theta_{t-1}) \\ \overset{(a)}{\le}& \ell_i(\theta_{t-1}) - \eta_t \nabla\ell_i(\theta_{t-1})^\top \widehat{g}_t(\theta_{t-1}) + \frac{L_G}{2}\left\|\eta_t \widehat{g}_t(\theta_{t-1})\right\|_2^2,\end{aligned} \tag{E.9.1}$$

where step (a) derives from L_G-Lipschitz of loss function in Assumption 4.

Recall that $g(\theta)$ and $\widehat{g}_t(\theta)$ are the gradient and stochastic gradient respectively in (E.2.2). From (E.9.1), we have

$$\begin{aligned}\mathbb{E}\left[\nabla\ell_i(\theta_t)\right] &= \mathbb{E}\left[\ell_i(\theta_t) - \mathbb{E}[\ell_i(\theta_\star)]\right] \\ &\le \mathbb{E}[\nabla\ell_i(\theta_{t-1})] - \eta_t \left|\nabla f(\theta_{t-1})\right|^2 \\ &\quad + \mathbb{E}\left[\frac{L_G}{2}\left|\eta_t \widehat{g}_t(\theta_{t-1})\right|^2\right] \\ &\le \mathbb{E}[\nabla\ell_i(\theta_{t-1})] - \eta_t \left|\nabla f(\theta_{t-1})\right|^2 \\ &\quad + \frac{L_G \eta t^2 \left|g(\theta_{t-1})\right|^2}{2} \\ &\quad + \frac{L_G\eta_t^2}{2}\mathbb{E}\left[\left|g(\theta_{t-1}) - \widehat{g}_t(\theta_{t-1})\right|^2\right] \\ &\le \mathbb{E}[\nabla\ell_i(\theta_{t-1})] - \eta_t \cdot 2\mu\mathbb{E}[\nabla\ell_i(\theta_{t-1})] \\ &\quad + \frac{L_G\eta_t^2 \cdot 2L_G\mathbb{E}[\nabla\ell_i(\theta_{t-1})]}{2} \\ &\quad + \frac{L_G\eta_t^2}{2} \cdot c'\left(\left|\theta_{t-1} - \theta_\star\right|^2\right) \\ &\le \mathbb{E}[\nabla\ell_i(\theta_{t-1})] - \eta_t \cdot 2\mu\mathbb{E}[\nabla\ell_i(\theta_{t-1})] \\ &\quad + \frac{L_G\eta_t^2 \cdot 2L_G\mathbb{E}[\nabla\ell_i(\theta_{t-1})]}{2} \\ &\quad + \frac{c' L_G\eta_t^2}{2} \cdot 4S^2,\end{aligned} \tag{E.9.2}$$

where step (a) comes from the inequalities

$$2\mu\left(f(\theta)-\mathbb{E}[\ell_i(\theta_\star)]\right)\leq\|\nabla\ell_i(\theta)\|_2^2\leq 2L_G\left(\ell_i(\theta)-\mathbb{E}[\ell_i(\theta_\star)]\right),$$

and step (b) comes from $\|\theta_\star\|_2\leq S$ *in Assumption 2. Rearranging the inequality (E.9.2), we get*

$$\mathbb{E}\left[\nabla\ell_i(\theta_t)\right]\leq\left(1+L_G^2\eta_t^2-2\mu\eta_t\right)\mathbb{E}[\nabla\ell_i(\theta_{t-1})]+2L_GS^2\cdot\eta_t^2, \tag{E.9.3}$$

which is equivalent to

$$\begin{aligned}\frac{\mathbb{E}\left[\nabla\ell_i(\theta_t)\right]}{\eta_t}&\leq\frac{\eta_{t-1}\left(1+L_G^2\eta_t^2-2\mu\eta_t\right)}{\eta_t}\frac{\mathbb{E}[\nabla\ell_i(\theta_{t-1})]}{\eta_{t-1}}+2L_GS^2\cdot\eta_t\\&\overset{(a)}{\leq}\frac{\eta_{t-1}\left(1-\mu\eta_t\right)}{\eta_t}\frac{\mathbb{E}[\nabla\ell_i(\theta_{t-1})]}{\eta_{t-1}}+2L_GS^2\cdot\eta_t,\end{aligned} \tag{E.9.4}$$

where step (a) can be derived by $L_G^2\eta_t^2\leq\mu\eta_t$ *as* $\eta_t=\eta_0t^{-\alpha}\to 0$.

In the following, we verity that $\mathbb{E}\left[\nabla\ell_i(\theta_t)\right]/\eta_t$ *converges with* $\eta_t=\eta_0t^{-\alpha}$*, i.e.,*

$$\sup_{1\leq t\leq\infty}\frac{\mathbb{E}\left[\nabla\ell_i(\theta_t)\right]}{\eta_t}<\infty, \tag{E.9.5}$$

which is achieved by contradiction.

Suppose that $\sup_{1\leq t\leq\infty}\mathbb{E}\left[\nabla\ell_i(\theta_t)\right]/\eta_t=\infty$. *Consider a sequence* $\{B_t\}$ *defined as*

$$B_t=\frac{\eta_{t-1}\left(1-\mu\eta_t\right)}{\eta_t}B_{t-1}+2L_GS^2\eta_t.$$

Note that $B_t\geq\mathbb{E}\left[\nabla\ell_i(\theta_t)\right]/\eta_t$ *for all* l. *Based on the assumption* $\sup_{1\leq t\leq\infty}\mathbb{E}\left[\nabla\ell_i(\theta_t)\right]/\eta_t=\infty$, *we have* $\sup_{1\leq t\leq\infty}B_t=\infty$. *Now, we get*

$$\begin{aligned}B_t=\frac{\eta_{t-1}\left(1-\mu\eta_t\right)}{\eta_t}B_{t-1}+2L_GS^2\eta_t&=\left(1+\frac{\eta_{t-1}-\eta_t}{\eta_t}\right)\left(1-\mu\eta_t\right)B_{t-1}+2L_GS^2\eta_t\\&=\left(1-\left(\mu-c\right)\eta_t\right)B_{t-1}+2L_GS^2\eta_t\\&=B_{t-1}-\left[\left(\mu-c\right)B_{t-1}-2L_GS^2\right]\eta_t,\end{aligned} \tag{E.9.6}$$

for some constant $c>0$.

Thus, once $B_t \geq (1+c)2L_G S^2/\mu$ for some $t \in [n]$. Thus this sequence can not diverge. By contradiction, we verify (E.9.5) which is equivalent to

$$\mathbb{E}[\ell_i(\theta_t) - \mathbb{E}[\ell_i(\theta_\star)]] \leq \frac{C}{2}\eta_t, \tag{E.9.7}$$

for some constant $C > 0$. According to strong convexity in Assumption 4, we have

$$\mathbb{E}[\ell_i(\theta_t) - \mathbb{E}[\ell_i(\theta_\star)]] \geq \frac{\mu}{2}\|\theta_t - \theta_\star\|_2^2. \tag{E.9.8}$$

Combining (E.9.7) and (E.9.8), it yields

$$\mathbb{E}[\|\theta_t - \theta_\star\|_2^2] \leq \frac{C}{\mu}\eta_t, \tag{E.9.9}$$

for some $C > 0$. □

E.10 Regret Result for Finite Actions

To better characterize the relationship between the regret bound and the number of actions $K := |\mathcal{D}|$, we state the upper regret bound dependent on K in the following proposition.

Proposition 5 *Under the same condition of Theorem 11, if the action space is finite with $K := |\mathcal{D}|$, Algorithm 7 achieves the following regret, with probability at least $1-\delta$,*

$$\mathrm{Reg}(n) \leq 32C''_{\chi,\alpha}\sqrt{\frac{ndKL^2L_H^2}{\mu^3}\log\left(\frac{n}{\delta}\right)},$$

where $n = TB$ and $C''_{\chi,\alpha} \asymp C_{\chi,\alpha}$ with $C_{\chi,\alpha}$ defined in (6.4).

Define $\Delta_i = r(\theta_\star, x^*) - r(\theta_\star, x_i)$. From (E.8.1), we conclude that a sub-optimal action $x_i \in \mathcal{D}$ is eliminated when

$$\Delta_i = r(\theta_\star, x^*) - r(\theta_\star, x_i) \geq 4L\beta_t \tag{E.10.1}$$

holds with probability at least $1-\delta$. Equivalently, it reaches a time $2 \leq t_i \leq n$ such that

$$t_i \geq (C''_{\chi,\alpha})^2\frac{256dL^2L_H^2}{\Delta_i^2\mu^3}\log\left(\frac{n}{\delta}\right). \tag{E.10.2}$$

Thus the regret concerning sub-optimal action x_i is

$$\mathrm{reg}_{x_i} \leq (C''_{\mathcal{X},\alpha})^2 \frac{256dL^2L_H^2}{\Delta_i \mu^3} \log\left(\frac{n}{\delta}\right). \tag{E.10.3}$$

To elude arbitrarily small Δ_i in the denominator, we decompose the regret to actions into two parts: action $x_i \in \mathcal{D}$ with $\Delta_i \leq \Delta$ and action $x_i \in \mathcal{D}$ with $\Delta_i > \Delta$ where $\Delta > 0$. Combining these two parts, we can get

$$\mathrm{Reg}(n) \leq n\Delta + (C''_{\mathcal{X},\alpha})^2 \frac{256dKL^2L_H^2}{\Delta \mu^3} \log\left(\frac{n}{\delta}\right). \tag{E.10.4}$$

By choosing $\Delta = C''_{\mathcal{X},\alpha}\sqrt{\frac{256dKL^2L_H^2}{n\mu^3}\log\left(\frac{n}{\delta}\right)}$, we get the regret bound as

$$\mathrm{Reg}(n) \leq 32C''_{\mathcal{X},\alpha}\sqrt{\frac{ndKL^2L_H^2}{\mu^3}\log\left(\frac{n}{\delta}\right)}. \tag{E.10.5}$$

E.11 Reward and Loss Functions for Different Models

Several reward functions and loss functions are presented in the following. Our paper includes but is not limited to these examples.

- The function $r(\theta, x) = (\theta^\top x)^2$ is $2\|x\|_2^2$-smooth with respect to $\theta \in \mathbb{R}^d$ for a fixed $x \in \mathcal{D}$ since $\nabla_\theta r(\theta, x) = 2\theta^\top x \cdot x$. The loss function of estimating θ can be given by

$$\ell_i(\theta) = \frac{1}{2n}\left((\theta^\top x_i)^2 - y_i\right)^2 + \frac{\mu}{2}\|\theta\|_2^2 \tag{E.11.1}$$

  for $i \in [n]$ and $\mu > 0$, which obeys Assumption 4.

- The function $r(\theta, x) = \log\left(1 + e^{\theta^\top x}\right)$ is $\frac{1}{4}\|x\|_2^2$-smooth with respect to $\theta \in \mathbb{R}^d$ for a fixed $x \in \mathcal{D}$ since $\nabla_\theta r(\theta, x) = \frac{x}{1+e^{-\theta^\top x}}$ and

$$\nabla_\theta^2 r(\theta, x) = \frac{e^{-\theta^\top x} \cdot xx^\top}{\left(1 + e^{-\theta^\top x}\right)^2} = \frac{xx^\top}{\left(1 + e^{-\theta^\top x}\right)\left(1 + e^{\theta^\top x}\right)} \preceq \frac{1}{4}\|x\|_2^2 I_d.$$

  The loss function of estimating θ can be given by

$$\ell_i(\theta) = \frac{1}{2n}\left(\log(1 + e^{\theta^\top x}) - y_i\right)^2 + \frac{\mu}{2}\|\theta\|_2^2 \tag{E.11.2}$$

  for $i \in [n]$ and $\mu > 0$, which obeys Assumption 4.

- The loss functions in (E.11.1) and (E.11.2) are used for minimizing mean squared error (MSE) between the prediction reward $r(\theta, x_i)$ and the observation reward y_i, where the regularized terms ensure the strong convexity. Besides, mean squared error, there are other measurement metrics such as mean squared logarithmic error (MSLE), i.e.,

  $$\frac{1}{2n}\sum_{i=1}^{n}\left(\log(y_i) - \log\left(r(\theta, x_i)\right)\right)^2, \tag{E.11.3}$$

  and the LogCosh loss [194], i.e.,

  $$\sum_{i=1}^{n}\log\left(\frac{e^{z_i}+e^{-z_i}}{2}\right), \quad \text{where} \quad z_i = r(\theta, x_i) - y_i, \tag{E.11.4}$$

  which computes the logarithm of the hyperbolic cosine of the prediction error. Corresponding to different bandit models, loss functions satisfying Assumption 4 can be established via choosing proper measurement metrics with the reasonable domain of parameters. Moreover, regularized terms can guarantee strong convexity of loss functions.

REFERENCES

[1] J. Dong, D. Zheng, L. F. Yang, and G. Karypis, "Global neighbor sampling for mixed cpu-gpu training on giant graphs," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 289–299, 2021.

[2] J. Dong and L. Yang, "Does sparsity help in learning misspecified linear bandits?," in *International Conference on Machine Learning*, pp. 8317–8333, PMLR, 2023.

[3] J. Dong, J. Wang, and L. F. Yang, "Delayed mdps with feature mapping," in *2024 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2024.

[4] J. Dong, J. Wang, and L. F. Yang, "Provably correct sgd-based exploration for generalized stochastic bandit problem," in *2024 International Conference on Smart Applications, Communications and Networking (SmartNets)*, pp. 1–6, IEEE, 2024.

[5] J. Dong, B. Fatemi, B. Perozzi, L. F. Yang, and A. Tsitsulin, "Don't forget to connect! improving rag with graph-based reranking," *arXiv preprint arXiv:2405.18414*, 2024.

[6] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations (ICLR-17)*, 2017.

[7] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *6th International Conference on Learning Representations (ICLR-18)*, 2018.

[8] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 1025–1035, 2017.

[9] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Graphsaint: Graph sampling based inductive learning method," *arXiv preprint arXiv:1907.04931*, 2019.

[10] J. Chen, T. Ma, and C. Xiao, "FastGCN: Fast learning with graph convolutional networks via importance sampling," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[11] D. Zou, Z. Hu, Y. Wang, S. Jiang, Y. Sun, and Q. Gu, "Layer-dependent importance sampling for training deep and large graph convolutional networks," in *Advances in Neural Information Processing Systems*, pp. 11249–11259, 2019.

[12] Z. Liu, Z. Wu, Z. Zhang, J. Zhou, S. Yang, L. Song, and Y. Qi, "Bandit samplers for training graph neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6878–6888, 2020.

[13] M. Ramezani, W. Cong, M. Mahdavi, A. Sivasubramaniam, and M. Kandemir, "GCN meets GPU: Decoupling "when to sample" from "how to sample"," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[14] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, and Z. Zhang, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," *arXiv preprint arXiv:1909.01315*, 2019.

[15] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *CoRR*, vol. abs/1903.02428, 2019.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

[18] D. Zheng, C. Ma, M. Wang, J. Zhou, Q. Su, X. Song, Q. Gan, Z. Zhang, and G. Karypis, "DistDGL: Distributed graph neural network training for billion-scale graphs," *arXiv preprint arXiv:2010.05337*, 2020.

[19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 4171–4186, Association for Computational Linguistics, June 2019.

[20] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *arXiv preprint arXiv:2005.00687*, 2020.

[21] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: Extraction and mining of academic social networks," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (New York, NY, USA), 2008.

[22] S. Siriwardhana, R. Weerasekera, E. Wen, T. Kaluarachchi, R. Rana, and S. Nanayakkara, "Improving the domain adaptation of retrieval augmented generation (rag) models for open domain question answering," *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 1–17, 2023.

[23] E. M. Voorhees and D. M. Tice, "The TREC-8 question answering track," in *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00)* (M. Gavrilidou, G. Carayannis, S. Markantonatou, S. Piperidis, and G. Stainhauer, eds.), (Athens, Greece), European Language Resources Association (ELRA), May 2000.

[24] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[25] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 7871–7880, Association for Computational Linguistics, July 2020.

[26] C. Wang, Z. Xu, Q. Guo, X. Hu, X. Bai, Z. Zhang, and Y. Zhang, "Exploiting Abstract Meaning Representation for open-domain question answering," in *Findings of the Association for Computational Linguistics: ACL 2023* (A. Rogers, J. Boyd-Graber, and N. Okazaki, eds.), (Toronto, Canada), pp. 2083–2096, Association for Computational Linguistics, July 2023.

[27] D. Yu, C. Zhu, Y. Fang, W. Yu, S. Wang, Y. Xu, X. Ren, Y. Yang, and M. Zeng, "KG-FiD: Infusing knowledge graph in fusion-in-decoder for open-domain question answering," in *ACL*, (Dublin, Ireland), pp. 4961–4974, Association for Computational Linguistics, May 2022.

[28] Google, R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen, E. Chu, J. H. Clark, L. E. Shafey, Y. Huang, K. Meier-Hellstern, G. Mishra, E. Moreira, M. Omernick, K. Robinson, S. Ruder, Y. Tay, K. Xiao, Y. Xu, Y. Zhang, G. H. Abrego, J. Ahn, J. Austin, P. Barham, J. Botha, J. Bradbury, S. Brahma, K. Brooks, M. Catasta, Y. Cheng, C. Cherry, C. A. Choquette-Choo, A. Chowdhery, C. Crepy, S. Dave, M. Dehghani, S. Dev, J. Devlin, M. Díaz, N. Du, E. Dyer, V. Feinberg, F. Feng, V. Fienber, M. Freitag, X. Garcia, S. Gehrmann, L. Gonzalez, G. Gur-Ari, S. Hand, H. Hashemi, L. Hou, J. Howland, A. Hu, J. Hui, J. Hurwitz, M. Isard, A. Ittycheriah, M. Jagielski, W. Jia, K. Kenealy, M. Krikun, S. Kudugunta, C. Lan, K. Lee, B. Lee, E. Li, M. Li, W. Li, Y. Li, J. Li, H. Lim, H. Lin, Z. Liu, F. Liu, M. Maggioni, A. Mahendru, J. Maynez, V. Misra, M. Moussalem, Z. Nado, J. Nham, E. Ni, A. Nystrom, A. Parrish, M. Pellat, M. Polacek, A. Polozov, R. Pope, S. Qiao, E. Reif, B. Richter, P. Riley, A. C. Ros, A. Roy, B. Saeta, R. Samuel, R. Shelby, A. Slone, D. Smilkov, D. R. So, D. Sohn, S. Tokumine, D. Valter, V. Vasudevan, K. Vodrahalli, X. Wang, P. Wang, Z. Wang, T. Wang, J. Wieting, Y. Wu, K. Xu, Y. Xu, L. Xue, P. Yin, J. Yu, Q. Zhang, S. Zheng, C. Zheng, W. Zhou, D. Zhou, S. Petrov, and Y. Wu, "Palm 2 technical report," 2023.

[29] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *NeurIPS*, vol. 33, pp. 9459–9474, 2020.

[30] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Online), pp. 6769–6781, Association for Computational Linguistics, 2020.

[31] R. Nogueira, Z. Jiang, R. Pradeep, and J. Lin, "Document ranking with a pretrained sequence-to-sequence model," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 708–718, 2020.

[32] R. Pradeep, Y. Liu, X. Zhang, Y. Li, A. Yates, and J. Lin, "Squeezing water from a stone: a bag of tricks for further improving cross-encoder effectiveness for reranking," in *European Conference on Information Retrieval*, pp. 655–670, Springer, 2022.

[33] H. Zhuang, Z. Qin, R. Jagerman, K. Hui, J. Ma, J. Lu, J. Ni, X. Wang, and M. Bendersky, "Rankt5: Fine-tuning T5 for text ranking with ranking losses," in *SIGIR*, pp. 2308–2313, 2023.

[34] E. Park, S.-M. Lee, D. Seo, S. Kim, I. Kang, and S.-H. Na, "Rink: reader-inherited evidence reranker for table-and-text open domain question answering," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37-11, pp. 13446–13456, 2023.

[35] M. Ju, W. Yu, T. Zhao, C. Zhang, and Y. Ye, "GRAPE: Knowledge graph enhanced passage reader for open-domain question answering," in *Findings of Empirical Methods in Natural Language Processing*, 2022.

[36] A. Asai, K. Hashimoto, H. Hajishirzi, R. Socher, and C. Xiong, "Learning to retrieve reasoning paths over wikipedia graph for question answering," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, 2020.

[37] J. O. Costa and A. Kulkarni, "Leveraging knowledge graph for open-domain question answering," in *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pp. 389–394, IEEE, 2018.

[38] L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider, "Abstract Meaning Representation for sembanking," in *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, (Sofia, Bulgaria), pp. 178–186, Association for Computational Linguistics, Aug. 2013.

[39] X. Bai, Y. Chen, L. Song, and Y. Zhang, "Semantic representation for dialogue modeling," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, (Online), pp. 4430–4445, Association for Computational Linguistics, Aug. 2021.

[40] T. Naseem, A. Blodgett, S. Kumaravel, T. J. O'Gorman, Y.-S. Lee, J. Flanigan, R. F. Astudillo, R. Florian, S. Roukos, and N. Schneider, "DocAMR: Multi-sentence AMR representation and evaluation," in *North American Chapter of the Association for Computational Linguistics*, 2021.

[41] OpenAI, "ChatGPT." `https://openai.com/research/chatgpt`.

[42] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[43] OpenAI, "GPT-4." `https://openai.com/gpt-4`.

[44] D. Huang, Z. Wei, A. Yue, X. Zhao, Z. Chen, R. Li, K. Jiang, B. Chang, Q. Zhang, S. Zhang, *et al.*, "DSQA-LLM: Domain-specific intelligent question answering based on large language model," in *International Conference on AI-generated Content*, pp. 170–180, Springer, 2023.

[45] X. Ma, L. Wang, N. Yang, F. Wei, and J. Lin, "Fine-tuning llama for multi-stage text retrieval," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2421–2425, 2024.

[46] C. Wang, S. Cheng, Z. Xu, B. Ding, Y. Wang, and Y. Zhang, "Evaluating open question answering evaluation," *arXiv preprint arXiv:2305.12421*, 2023.

[47] Y. Tan, D. Min, Y. Li, W. Li, N. Hu, Y. Chen, and G. Qi, "Can ChatGPT replace traditional KBQA models? an in-depth analysis of the question answering performance of the GPT LLM family," in *International Semantic Web Conference*, pp. 348–367, Springer, 2023.

[48] G. Izacard and É. Grave, "Leveraging passage retrieval with generative models for open domain question answering," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 874–880, 2021.

[49] X. Bai, Y. Chen, and Y. Zhang, "Graph pre-training for AMR parsing and generation," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Dublin, Ireland), pp. 6001–6015, Association for Computational Linguistics, May 2022.

[50] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, *et al.*, "Natural questions: a benchmark for question answering research," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 453–466, 2019.

[51] M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer, "TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (R. Barzilay and M.-Y. Kan, eds.), (Vancouver, Canada), pp. 1601–1611, Association for Computational Linguistics, July 2017.

[52] L. Gong and Q. Cheng, "Exploiting edge features for graph neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9211–9219, 2019.

[53] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *ICLR*, 2017.

[54] Y. Li, Y. Song, and J. Luo, “Improving pairwise ranking for multi-label image classification,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3617–3625, 2017.

[55] Z. Li, X. Zhang, Y. Zhang, D. Long, P. Xie, and M. Zhang, “Towards general text embeddings with multi-stage contrastive learning,” *arXiv preprint arXiv:2308.03281*, 2023.

[56] S. Xiao, Z. Liu, P. Zhang, N. Muennighoff, D. Lian, and J.-Y. Nie, “C-pack: Packaged resources to advance general chinese embedding,” *arXiv preprint arXiv:2309.07597*, 2023.

[57] Z. Liu and Y. Shao, “RetroMAE: Pre-training retrieval-oriented transformers via masked auto-encoder,” *arXiv preprint arXiv:2205.12035*, 2022.

[58] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, *et al.*, “Huggingface’s transformers: State-of-the-art natural language processing,” *arXiv preprint arXiv:1910.03771*, 2019.

[59] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.

[60] M. Glass, G. Rossiello, M. F. M. Chowdhury, A. Naik, P. Cai, and A. Gliozzo, “Re2G: Retrieve, rerank, generate,” in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (Seattle, United States), pp. 2701–2715, Association for Computational Linguistics, July 2022.

[61] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, “Deep reinforcement learning for autonomous driving: A survey,” *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[62] A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. DePristo, K. Chou, C. Cui, G. Corrado, S. Thrun, and J. Dean, “A guide to deep learning in healthcare,” *Nature medicine*, vol. 25, no. 1, pp. 24–29, 2019.

[63] D. Bouneffouf, A. Bouzeghoub, and A. L. Gançarski, “A contextual-bandit algorithm for mobile context-aware recommender system,” in *International conference on neural information processing*, pp. 324–331, Springer, 2012.

[64] E. M. Schwartz, E. T. Bradlow, and P. S. Fader, “Customer acquisition via display advertising using multi-armed bandit experiments,” *Marketing Science*, vol. 36, no. 4, pp. 500–522, 2017.

[65] S. S. Du, S. M. Kakade, R. Wang, and L. F. Yang, "Is a good representation sufficient for sample efficient reinforcement learning?," in *International Conference on Learning Representations*, 2020.

[66] T. Lattimore, C. Szepesvari, and G. Weisz, "Learning with good feature representations in bandits and in rl with a generative model," in *International Conference on Machine Learning*, pp. 5662–5670, PMLR, 2020.

[67] P. Bühlmann and S. Van De Geer, *Statistics for high-dimensional data: methods, theory and applications*. Springer Science & Business Media, 2011.

[68] M. J. Wainwright, *High-dimensional statistics: A non-asymptotic viewpoint*, vol. 48. Cambridge University Press, 2019.

[69] V. Sivakumar, S. Wu, and A. Banerjee, "Structured linear contextual bandits: A sharp and geometric smoothed analysis," in *International Conference on Machine Learning*, pp. 9026–9035, PMLR, 2020.

[70] Y. Abbasi-Yadkori, D. Pal, and C. Szepesvari, "Online-to-confidence-set conversions and application to sparse stochastic bandits," in *Artificial Intelligence and Statistics*, pp. 1–9, PMLR, 2012.

[71] H. Bastani and M. Bayati, "Online decision making with high-dimensional covariates," *Operations Research*, vol. 68, no. 1, pp. 276–294, 2020.

[72] X. Wang, M. Wei, and T. Yao, "Minimax concave penalized multi-armed bandit model with high-dimensional covariates," in *International Conference on Machine Learning*, pp. 5200–5208, PMLR, 2018.

[73] Y. Su, M. Dimakopoulou, A. Krishnamurthy, and M. Dudík, "Doubly robust off-policy evaluation with shrinkage," in *International Conference on Machine Learning*, pp. 9167–9176, PMLR, 2020.

[74] T. Lattimore, K. Crammer, and C. Szepesvári, "Linear multi-resource allocation with semi-bandit feedback," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[75] A. Carpentier and R. Munos, "Bandit theory meets compressed sensing for high dimensional stochastic linear bandit," in *Artificial Intelligence and Statistics*, pp. 190–198, PMLR, 2012.

[76] I. Bogunovic and A. Krause, "Misspecified gaussian process bandit optimization," *Advances in Neural Information Processing Systems*, vol. 34, pp. 3004–3015, 2021.

[77] K. Takemura, S. Ito, D. Hatano, H. Sumita, T. Fukunaga, N. Kakimura, and K.-i. Kawarabayashi, "A parameter-free algorithm for misspecified linear contextual bandits," in *International Conference on Artificial Intelligence and Statistics*, pp. 3367–3375, PMLR, 2021.

[78] A. Zanette, A. Lazaric, M. Kochenderfer, and E. Brunskill, “Learning near optimal policies with low inherent Bellman error,” in *International Conference on Machine Learning*, pp. 10978–10989, PMLR, 2020.

[79] R. Wang, R. R. Salakhutdinov, and L. Yang, “Reinforcement learning with general value function approximation: Provably efficient approach via bounded Eluder dimension,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 6123–6135, 2020.

[80] Q. Ding, C.-J. Hsieh, and J. Sharpnack, “An efficient algorithm for generalized linear bandit: Online stochastic gradient descent and thompson sampling,” in *International Conference on Artificial Intelligence and Statistics*, pp. 1585–1593, PMLR, 2021.

[81] D. Russo and B. Van Roy, “Eluder dimension and the sample complexity of optimistic exploration,” *Advances in Neural Information Processing Systems*, vol. 26, 2013.

[82] V. Dani, T. P. Hayes, and S. M. Kakade, “Stochastic linear optimization under bandit feedback,” in *COLT*, pp. 355–366, 2008.

[83] W. Chu, L. Li, L. Reyzin, and R. Schapire, “Contextual bandits with linear payoff functions,” in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 208–214, 2011.

[84] C. Jin, Z. Yang, Z. Wang, and M. I. Jordan, “Provably efficient reinforcement learning with linear function approximation,” in *Conference on Learning Theory*, pp. 2137–2143, PMLR, 2020.

[85] Q. Cai, Z. Yang, C. Jin, and Z. Wang, “Provably efficient exploration in policy optimization,” in *International Conference on Machine Learning*, pp. 1283–1294, PMLR, 2020.

[86] A. Zanette, A. Lazaric, M. J. Kochenderfer, and E. Brunskill, “Provably efficient reward-agnostic navigation with linear value iteration,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 11756–11766, 2020.

[87] A. Agarwal, S. Kakade, A. Krishnamurthy, and W. Sun, “Flambe: Structural complexity and representation learning of low rank MDPs,” *Advances in neural information processing systems*, vol. 33, pp. 20095–20107, 2020.

[88] G. Neu and C. Pike-Burke, “A unifying view of optimism in episodic reinforcement learning,” *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[89] Y. Wang, R. Wang, S. S. Du, and A. Krishnamurthy, “Optimism in reinforcement learning with generalized linear function approximation,” in *International Conference on Learning Representations*, 2020.

[90] I. Osband and B. Van Roy, “Model-based reinforcement learning and the eluder dimension,” *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[91] N. Jiang, A. Krishnamurthy, A. Agarwal, J. Langford, and R. E. Schapire, "Contextual decision processes with low Bellman rank are PAC-learnable," in *International Conference on Machine Learning*, pp. 1704–1713, PMLR, 2017.

[92] L. Li, Y. Lu, and D. Zhou, "Provably optimal algorithms for generalized linear contextual bandits," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2071–2080, JMLR. org, 2017.

[93] B. Kveton, M. Zaheer, C. Szepesvari, L. Li, M. Ghavamzadeh, and C. Boutilier, "Randomized exploration in generalized linear bandits," in *International Conference on Artificial Intelligence and Statistics*, pp. 2066–2076, 2020.

[94] S. Filippi, O. Cappe, A. Garivier, and C. Szepesvári, "Parametric bandits: The generalized linear case," in *Advances in Neural Information Processing Systems*, pp. 586–594, 2010.

[95] K.-S. Jun, A. Bhargava, R. Nowak, and R. Willett, "Scalable generalized linear bandits: Online computation and hashing," in *Advances in Neural Information Processing Systems*, pp. 99–109, 2017.

[96] D. Foster, A. Rakhlin, D. Simchi-Levi, and Y. Xu, "Instance-dependent complexity of contextual bandits and reinforcement learning: A disagreement-based perspective," in *Conference on Learning Theory*, pp. 2059–2059, PMLR, 2021.

[97] D. J. Foster, C. Gentile, M. Mohri, and J. Zimmert, "Adapting to misspecification in contextual bandits," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11478–11489, 2020.

[98] D. Vial, A. Parulekar, S. Shakkottai, and R. Srikant, "Improved algorithms for misspecified linear markov decision processes," in *International Conference on Artificial Intelligence and Statistics*, pp. 4723–4746, PMLR, 2022.

[99] C.-Y. Wei, C. Dann, and J. Zimmert, "A model selection approach for corruption robust reinforcement learning," in *International Conference on Algorithmic Learning Theory*, pp. 1043–1096, PMLR, 2022.

[100] T. Lattimore and C. Szepesvári, *Bandit algorithms*. Cambridge University Press, 2020.

[101] G.-S. Kim and M. C. Paik, "Doubly-robust lasso bandit," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[102] B. Hao, T. Lattimore, and M. Wang, "High-dimensional sparse linear bandits," *Advances in Neural Information Processing Systems*, vol. 33, pp. 10753–10763, 2020.

[103] J. Z. Kolter and A. Y. Ng, "Regularization and feature selection in least-squares temporal difference learning," in *Proceedings of the 26th annual international conference on machine learning*, pp. 521–528, 2009.

[104] M. Geist and B. Scherrer, "ℓ_1-penalized projected bellman residual," in *European Workshop on Reinforcement Learning*, pp. 89–101, Springer, 2012.

[105] C. Painter-Wakefield and R. Parr, "Greedy algorithms for sparse reinforcement learning," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pp. 867–874, 2012.

[106] B. Liu, S. Mahadevan, and J. Liu, "Regularized off-policy td-learning," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[107] M. Ghavamzadeh, A. Lazaric, R. Munos, and M. Hoffman, "Finite-sample analysis of lasso-td," in *International Conference on Machine Learning*, 2011.

[108] M. Geist, B. Scherrer, A. Lazaric, and M. Ghavamzadeh, "A dantzig selector approach to temporal difference learning," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pp. 347–354, 2012.

[109] B. Hao, Y. Duan, T. Lattimore, C. Szepesvári, and M. Wang, "Sparse feature selection makes batch reinforcement learning more sample efficient," in *International Conference on Machine Learning*, pp. 4063–4073, PMLR, 2021.

[110] M. Ibrahimi, A. Javanmard, and B. Roy, "Efficient reinforcement learning for high dimensional linear quadratic systems," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[111] B. Hao, T. Lattimore, C. Szepesvári, and M. Wang, "Online sparse reinforcement learning," in *International Conference on Artificial Intelligence and Statistics*, pp. 316–324, PMLR, 2021.

[112] A. C.-C. Yao, "Probabilistic computations: Toward a unified measure of complexity," in *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pp. 222–227, IEEE Computer Society, 1977.

[113] M. J. Todd, *Minimum-volume ellipsoids: Theory and algorithms*. SIAM, 2016.

[114] W. B. Johnson and Lindenstrauss, "Extensions of lipschitz mappings into a hilbert space," *Contemp. Math.*, vol. 26, pp. 189–206, 1984.

[115] D. M. Kane and J. Nelson, "Sparser johnson-lindenstrauss transforms," *Journal of the ACM (JACM)*, vol. 61, no. 1, pp. 1–23, 2014.

[116] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[117] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[118] N. Li, D. W. Oyler, M. Zhang, Y. Yildiz, I. Kolmanovsky, and A. R. Girard, "Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems," *IEEE Transactions on control systems technology*, vol. 26, no. 5, pp. 1782–1797, 2017.

[119] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 3, pp. 653–664, 2016.

[120] D. P. Bertsekas, "Feature-based aggregation and deep reinforcement learning: A survey and some new implementations," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 1, pp. 1–31, 2018.

[121] A. Bar-Ilan and A. Sulem, "Explicit solution of inventory problems with delivery lags," *Mathematics of Operations Research*, vol. 20, no. 3, pp. 709–720, 1995.

[122] L. Dugard and E. I. Verriest, *Stability and control of time-delay systems*, vol. 228. Springer, 1998.

[123] J.-P. Richard, "Time-delay systems: an overview of some recent advances and open problems," *automatica*, vol. 39, no. 10, pp. 1667–1694, 2003.

[124] E. Fridman, *Introduction to time-delay systems: Analysis and control*. Springer, 2014.

[125] B. Bruder and H. Pham, "Impulse control problem on finite horizon with execution delay," *Stochastic Processes and their Applications*, vol. 119, no. 5, pp. 1436–1469, 2009.

[126] L. Yang and M. Wang, "Reinforcement learning in feature space: Matrix bandit, kernels, and regret bound," in *International Conference on Machine Learning*, pp. 10746–10756, PMLR, 2020.

[127] K. V. Katsikopoulos and S. E. Engelbrecht, "Markov decision processes with delays and asynchronous cost collection," *IEEE transactions on automatic control*, vol. 48, no. 4, pp. 568–574, 2003.

[128] T. J. Walsh, A. Nouri, L. Li, and M. L. Littman, "Learning and planning in environments with delayed feedback," *Autonomous Agents and Multi-Agent Systems*, vol. 18, no. 1, pp. 83–105, 2009.

[129] P. Joulani, A. Gyorgy, and C. Szepesvári, "Online learning under delayed feedback," in *International Conference on Machine Learning*, pp. 1453–1461, 2013.

[130] J. S. Campbell, S. N. Givigi, and H. M. Schwartz, "Multiple model q-learning for stochastic asynchronous rewards," *Journal of Intelligent & Robotic Systems*, vol. 81, no. 3-4, pp. 407–422, 2016.

[131] B. Chen, M. Xu, Z. Liu, L. Li, and D. Zhao, "Delay-aware multi-agent reinforcement learning," *arXiv preprint arXiv:2005.05441*, 2020.

[132] E. Derman, G. Dalal, and S. Mannor, "Acting in delayed environments with non-stationary markov policies," in *International Conference on Learning Representations*, 2021.

[133] T. Lancewicki, A. Rosenberg, and Y. Mansour, "Learning adversarial markov decision processes with delayed feedback," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36-7, pp. 7281–7289, 2022.

[134] S. Ramstedt and C. Pal, "Real-time reinforcement learning," in *Advances in Neural Information Processing Systems*, pp. 3073–3082, 2019.

[135] T. Xiao, E. Jang, D. Kalashnikov, S. Levine, J. Ibarz, K. Hausman, and A. Herzog, "Thinking while moving: Deep reinforcement learning with concurrent control," in *International Conference on Learning Representations*, 2020.

[136] T. Hester and P. Stone, "Texplore: real-time sample-efficient reinforcement learning for robots," *Machine learning*, vol. 90, no. 3, pp. 385–429, 2013.

[137] A. Modi, N. Jiang, A. Tewari, and S. Singh, "Sample complexity of reinforcement learning using linearly combined model ensembles," in *International Conference on Artificial Intelligence and Statistics*, pp. 2010–2020, PMLR, 2020.

[138] A. Ayoub, Z. Jia, C. Szepesvari, M. Wang, and L. Yang, "Model-based reinforcement learning with value-targeted regression," in *International Conference on Machine Learning*, pp. 463–474, PMLR, 2020.

[139] R. Wang, R. Salakhutdinov, and L. F. Yang, "Provably efficient reinforcement learning with general value function approximation," *arXiv preprint arXiv:2005.10804*, 2020.

[140] Z. Yang, C. Jin, Z. Wang, M. Wang, and M. I. Jordan, "On function approximation in reinforcement learning: Optimism in the face of large state spaces," *arXiv preprint arXiv:2011.04622*, 2020.

[141] L. Yang and M. Wang, "Sample-optimal parametric Q-learning using linearly additive features," in *International Conference on Machine Learning*, pp. 6995–7004, PMLR, 2019.

[142] D. Zhou, J. He, and Q. Gu, "Provably efficient reinforcement learning for discounted MDPs with feature mapping," in *International Conference on Machine Learning*, PMLR, 2021.

[143] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman, "PAC model-free reinforcement learning," in *Proceedings of the 23rd international conference on Machine learning*, pp. 881–888, 2006.

[144] I. Szita and C. Szepesvari, “Model-based reinforcement learning with nearly tight exploration complexity bounds,” in *International Conference on Machine Learning*, pp. 1031–1038, 2010.

[145] T. Lattimore and M. Hutter, “PAC bounds for discounted MDPs,” in *International Conference on Algorithmic Learning Theory*, pp. 320–334, Springer, 2012.

[146] Y. Wang, K. Dong, X. Chen, and L. Wang, “Q-learning with ucb exploration is sample efficient for infinite-horizon mdp,” in *International Conference on Learning Representations*, 2019.

[147] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári, “Improved algorithms for linear stochastic bandits,” in *Proceedings of the 24th International Conference on Neural Information Processing Systems*, pp. 2312–2320, 2011.

[148] W. Xia, T. Q. Quek, K. Guo, W. Wen, H. H. Yang, and H. Zhu, “Multi-armed bandit-based client scheduling for federated learning,” *IEEE Transactions on Wireless Communications*, vol. 19, no. 11, pp. 7108–7123, 2020.

[149] A. Slivkins *et al.*, “Introduction to multi-armed bandits,” *Foundations and Trends® in Machine Learning*, vol. 12, no. 1-2, pp. 1–286, 2019.

[150] S. Maghsudi and S. Stańczak, “Channel selection for network-assisted d2d communication via no-regret bandit learning with calibrated forecasting,” *IEEE Transactions on Wireless Communications*, vol. 14, no. 3, pp. 1309–1322, 2014.

[151] F. Li, D. Yu, H. Yang, J. Yu, H. Karl, and X. Cheng, “Multi-armed-bandit-based spectrum scheduling algorithms in wireless networks: A survey,” *IEEE Wireless Communications*, vol. 27, no. 1, pp. 24–30, 2020.

[152] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge university press, 2006.

[153] S. Bubeck, N. Cesa-Bianchi, and S. M. Kakade, “Towards minimax policies for online linear optimization with bandit feedback,” in *Conference on Learning Theory*, pp. 41–1, JMLR Workshop and Conference Proceedings, 2012.

[154] N. Korda, L. Prashanth, and R. Munos, “Fast gradient descent for drifting least squares regression, with application to bandits,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29-1, 2015.

[155] Y. Han, Z. Liang, Y. Wang, and J. Zhang, “Generalized linear bandits with local differential privacy,” *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[156] P. Auer, N. Cesa-Bianchi, and P. Fischer, “Finite-time analysis of the multiarmed bandit problem,” *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[157] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proceedings of the 19th international conference on World wide web*, pp. 661–670, 2010.

[158] M. Abeille, A. Lazaric, *et al.*, "Linear thompson sampling revisited," *Electronic Journal of Statistics*, vol. 11, no. 2, pp. 5165–5197, 2017.

[159] S. Dong, T. Ma, and B. V. Roy, "On the performance of thompson sampling on logistic bandits.," in *COLT*, pp. 1158–1160, 2019.

[160] O. Chapelle and L. Li, "An empirical evaluation of thompson sampling," in *Advances in neural information processing systems*, pp. 2249–2257, 2011.

[161] D. Russo and B. Van Roy, "Learning to optimize via posterior sampling," *Mathematics of Operations Research*, vol. 39, no. 4, pp. 1221–1243, 2014.

[162] B. Awerbuch and R. D. Kleinberg, "Adaptive routing with end-to-end feedback: Distributed learning and geometric approaches," in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pp. 45–53, 2004.

[163] H. B. McMahan and A. Blum, "Online geometric optimization in the bandit setting against an adaptive adversary," in *International Conference on Computational Learning Theory*, pp. 109–123, Springer, 2004.

[164] V. Dani, S. M. Kakade, and T. Hayes, "The price of bandit information for online optimization," *Advances in Neural Information Processing Systems*, vol. 20, 2007.

[165] A. S. Nemirovskij and D. B. Yudin, *Problem complexity and method efficiency in optimization*. Wiley-Interscience, 1983.

[166] J. D. Abernethy, E. Hazan, and A. Rakhlin, "Competing in the dark: An efficient algorithm for bandit linear optimization.," in *COLT*, pp. 263–274, Citeseer, 2008.

[167] S. Bubeck, N. Cesa-Bianchi, *et al.*, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Foundations and Trends® in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[168] A. Bietti, A. Agarwal, and J. Langford, "A contextual bandit bake-off," *Journal of Machine Learning Research*, vol. 22, no. 133, pp. 1–49, 2021.

[169] C. Riquelme, G. Tucker, and J. Snoek, "Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling," in *International Conference on Learning Representations*, 2018.

[170] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, pp. 177–186, Springer, 2010.

[171] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the trade*, pp. 421–436, Springer, 2012.

[172] B. T. Polyak and A. B. Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM journal on control and optimization*, vol. 30, no. 4, pp. 838–855, 1992.

[173] A. Rakhlin, O. Shamir, and K. Sridharan, "Making gradient descent optimal for strongly convex stochastic optimization," in *Proceedings of the 29th International Conference on International Conference on Machine Learning*, pp. 1571–1578, 2012.

[174] M. Zinkevich, M. Weimer, L. Li, and A. Smola, "Parallelized stochastic gradient descent," *Advances in neural information processing systems*, vol. 23, 2010.

[175] Z. Lou, W. Zhu, and W. B. Wu, "Beyond sub-gaussian noises: Sharp concentration analysis for stochastic gradient descent," *Journal of Machine Learning Research*, vol. 23, pp. 1–22, 2022.

[176] N. J. Harvey, C. Liaw, Y. Plan, and S. Randhawa, "Tight analyses for non-smooth stochastic gradient descent," in *Conference on Learning Theory*, pp. 1579–1613, PMLR, 2019.

[177] E. Hazan and S. Kale, "Beyond the regret minimization barrier: optimal algorithms for stochastic strongly-convex optimization," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 2489–2512, 2014.

[178] H. Cardot, P. Cénac, and A. Godichon-Baggioni, "Online estimation of the geometric median in hilbert spaces: Nonasymptotic confidence balls," *The Annals of Statistics*, vol. 45, no. 2, pp. 591–614, 2017.

[179] P. Jain, D. Nagaraj, and P. Netrapalli, "Making the last iterate of sgd information theoretically optimal," in *Conference on Learning Theory*, pp. 1752–1755, PMLR, 2019.

[180] V. Feldman and J. Vondrak, "High probability generalization bounds for uniformly stable algorithms with nearly optimal rate," in *Conference on Learning Theory*, pp. 1270–1279, PMLR, 2019.

[181] W. Mou, C. J. Li, M. J. Wainwright, P. L. Bartlett, and M. I. Jordan, "On linear stochastic approximation: Fine-grained polyak-ruppert and non-asymptotic concentration," in *Conference on Learning Theory*, pp. 2947–2997, PMLR, 2020.

[182] R. S. Sutton, A. G. Barto, *et al.*, *Introduction to reinforcement learning*, vol. 135. MIT press Cambridge, 1998.

[183] J. M. Kohler and A. Lucchi, "Sub-sampled cubic regularization for non-convex optimization," in *International Conference on Machine Learning*, pp. 1895–1904, 2017.

[184] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[185] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” *Advances in neural information processing systems*, vol. 30, 2017.

[186] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?,” in *International Conference on Learning Representations*, 2018.

[187] S. M. Ross, *Introduction to probability models*. Academic press, 2014.

[188] T. Jaksch, R. Ortner, and P. Auer, “Near-optimal regret bounds for reinforcement learning.,” *Journal of Machine Learning Research*, vol. 11, no. 4, 2010.

[189] I. Osband, B. Van Roy, D. J. Russo, and Z. Wen, “Deep exploration via randomized value functions.,” *Journal of Machine Learning Research*, vol. 20, no. 124, pp. 1–62, 2019.

[190] I. Osband, B. Van Roy, and Z. Wen, “Generalization and exploration via randomized value functions,” in *International Conference on Machine Learning*, pp. 2377–2386, PMLR, 2016.

[191] Y. Abbasi-Yadkori and C. Szepesvari, “Bayesian optimal control of smoothly parameterized systems,” in *Uncertainty in Artificial Intelligence: Proceedings of the 31st Conference, UAI 2015*, pp. 2–11, AUAI Press (Association for Uncertainty in Artificial Intelligence), 2015.

[192] E. G. Birgin, J. M. Martínez, and M. Raydan, “Nonmonotone spectral projected gradient methods on convex sets,” *SIAM Journal on Optimization*, vol. 10, no. 4, pp. 1196–1211, 2000.

[193] A. Anastasiou, K. Balasubramanian, and M. A. Erdogdu, “Normal approximation for stochastic gradient descent via non-asymptotic rates of martingale clt,” in *Conference on Learning Theory*, pp. 115–137, PMLR, 2019.

[194] S. Jadon, “A survey of loss functions for semantic segmentation,” in *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pp. 1–7, IEEE, 2020.