

A Study on Stochastic Combinatorial Bandit Problems
with Limited Observation

(限られた観測に基づく確率的組合せバンディットの研究)

by

Yuko Kuroki

黒木 祐子

A Doctor Thesis

博士論文

Submitted to

the Graduate School of the University of Tokyo

on December 4, 2020

in Partial Fulfillment of the Requirements

for the Degree of Doctor of Information Science and Technology

in Computer Science

Thesis Supervisor: Masashi Sugiyama 杉山 将

Professor of Computer Science

ABSTRACT

Decision making under uncertainty is common in prudent weighing of outcomes or utility because we human beings are constantly exposed to risk or ambiguity situations. In the face of uncertainty in our lives, we aim to take the best possible decision based on the past experience and conjecture. Modeling of decision making under uncertainty and its analysis have been investigated in diverse research fields such as computer science, statistics, psychology, and neuroscience.

The problem of *stochastic multi-armed bandits* is one of the classical decision making models, and it lies at the intersection of statistics and machine learning. The stochastic multi-armed bandit problem characterizes the trade-off between exploration and exploitation in stochastic environments; in this problem, an agent chooses one action (a.k.a. arm) from K given arms at each step, and obtains a reward sampled from an unknown probability distribution associated with the chosen arm.

In many real-world scenarios, our decisions are often characterized by a combinatorial and complex structure. For example, possible actions in real-world systems may be a subset of keywords in online advertisements, assignments of tasks to workers in crowdsourcing, or paths in communication networks. The problem of *stochastic combinatorial bandits* is a generalization of the multi-armed bandits, which can deal with such combinatorial actions; a subset of arms (a.k.a super arm) is an action in this model, while each single arm is an action in the multi-armed bandit problem.

In this thesis, we study stochastic combinatorial bandit problems from an algorithmic perspective. Motivated by real-world applications, we develop theory and algorithms for challenging settings, in which only limited feedback is given to an agent. In the combinatorial bandit literature, a large amount of work has studied the *semi-bandit* setting, in which an agent can directly observe a random feedback from an individual arm at each sampling trial. However, such a sampling procedure for individual arms is costly or often impossible due to the privacy issues or system constraints. To overcome this critical limitation of existing studies, we aim to propose statistically and computationally efficient methods based on only limited observation. To this end, we address several approximation algorithms for NP-hard problems and bandit algorithms that can deal with an aggregated feedback from a super arm.

This thesis focuses on the combinatorial bandit problems in the *pure exploration* setting; in this setting, an agent aims to identify the optimal super arm with the highest expected reward. The learning performance is evaluated by the number of samples required by her learning algorithm, i.e., the *sample complexity*, rather than the cumulative rewards. The main body of this thesis consists of four parts, which corresponds to Chapters 3–6. The first part is devoted to a study of pure exploration for top- k arms with *full-bandit feedback*, where underlying combinatorial structures are simply size- k subsets but available feedback is limited to a noisy evaluation for a sampled subset of arms, i.e., full-bandit feedback. The second part is devoted to a study of general combinatorial pure exploration with full-bandit feedback, in which possible combinatorial structures are size- k subsets, matchings, and paths. The third part is devoted to a study of combinatorial pure exploration over graphs with full-bandit feedback, which aims to identify a dense component in a network only together with a noisy evaluation for a sampled subgraph. The fourth part is devoted to a study of general combinatorial pure exploration with *partial-linear feedback*, which aims to develop a novel algorithmic framework for general (possibly nonlinear) reward functions and combinatorial structures including size- k subsets, matchings, and paths. Partial-linear feedback ranges from semi-bandit and full-bandit feedback; in this feedback model, the agent only observes a linear combination of rewards of the chosen super arm at each pull.

In Chapter 3, we first investigate a novel problem of pure exploration for top- k arms with full-bandit feedback. Although our problem can be regarded as an instance of linear bandit problems, a naive approach using linear bandit algorithms is computationally infeasible to the problem instance in the combinatorial setting, since the number of possible actions K is exponential with respect to the number of arms in the subset. To cope with this problem, we design a polynomial-time approximation algorithm for the 0-1 quadratic programming problem arising in confidence ellipsoid maximization. Based on our approximation algorithm, we propose a bandit algorithm whose time complexity is $O(\log K)$, thereby achieving an exponential speedup over linear bandit algorithms. We also provide an upper bound on the sample complexity that is worst-case optimal. Our experiments on large-scale crowdsourcing datasets with more than 10^{10} possible actions demonstrate the superiority of our algorithm in terms of both the computation time and the sample complexity.

In Chapter 4, we study combinatorial pure exploration with full-bandit feedback for general combinatorial structures such as matroids, matchings, and s - t paths. We devise a polynomial-time adaptive algorithm whose sample complexity has mild dependence on the *minimal gap* between the optimal value and the second optimal value. We show that the sample complexity matches (within a logarithmic factor) the information theoretic lower bound for a family of instances. We conduct a series of experiments on the matching and top- k instances and demonstrate that (a) the proposed algorithm scales to combinatorial instances while existing linear bandit algorithms require a prohibitive amount of time; (b) the number of samples required by the proposed algorithm is not sensitive to the value of the minimum gap.

In Chapter 5, we focus on a specific instance of combinatorial pure exploration with linear feedback over graphs with applications to *dense subgraph discovery under uncertainty*. We introduce a novel learning problem for online dense subgraph discovery, in which an agent queries subsets of edges rather than single edges and observes a noisy sum of edge weights in a queried subset. For this problem, we propose a polynomial-time algorithm that obtains a nearly-optimal solution with high probability. For the proposed algorithm, we provide an upper bound of the number of samples that the algorithm requires. Moreover, to deal with large-sized graphs, we design a more scalable algorithm that maximizes the quality of solution within a fixed number of queries. Computational experiments using real-world graphs demonstrate the effectiveness of our algorithms.

In Chapter 6, we study a novel model of general combinatorial pure exploration with partial-linear feedback, which includes all the problems addressed in Chapters 3–5 as special cases and finds various applications such as online ranking in recommendation systems and crowdsourcing. The model of partial-linear feedback fits in such real-world applications since it may happen that we cannot always observe outcomes from some of the chosen arms due to privacy concern or system constraints. We propose a polynomial-time algorithmic framework for this general problem, and provide a sample complexity analysis.

To summarize, we investigate the learnability of the combinatorial pure exploration problems with limited feedback and develop polynomial-time algorithms. Our results provide novel insights into online decision making problems with combinatorial action spaces and combinatorial optimization under uncertainty for incomplete inputs.

論文要旨

我々人間は、予想不可能な事態、危険性に常に晒されながら、意思決定を行なっている。我々は不確実な事象を前にして、過去の経験や推測に基づいて最善な意思決定を下そうとする。そのような不確実性の下での意思決定は、より良い結果を得るために重要な営みである。不確実性を伴う意思決定のモデル化とその分析は、情報科学、統計学、心理学、神経科学など幅広い学術分野で古くから議論されてきた。

確率的多腕バンディット問題は古典的な意思決定モデルの一つであり、統計学と機械学習の共通領域に位置している。この問題では、エージェントは K 個の行動候補からただ一つの行動を各時刻で選択しなければならない。そして、選択した行動により (確率的な) 報酬を得るが、選択しなかった行動からの報酬は観測不可能であり、ここに探索と活用のジレンマが生じている。

我々の意思決定は組合せ的な意思決定がほとんどである。例えば、オンライン広告におけるキーワードの選択や、労働者とタスクの割当、通信ネットワークにおける経路などは、組合せ的な意思決定問題である。確率的組合せバンディット問題では、各行動がある組合せ的構造を持っている行動を扱っており、この問題は上記の確率的多腕バンディット問題の一般化となっている。

本学位論文では、実世界の状況をより忠実に扱うために、限られた観測しか得られない挑戦的な設定における組合せバンディット問題に対するモデル化とアルゴリズム設計を目指す。既存研究の主流では、半バンディットと呼ばれる選択した行動に含まれる各要素を個々に直接観測する場合が扱われてきた。しかし、実世界の問題では、プライバシー保護や実応用上の制約、コストの観点から、そのような個々に関する観測を得ることは難しく、現実的な設定とは言えない。本研究ではこのような既存研究の枠組みの問題を解決すべく、理論と応用の両側面から良い性質を持った手法の開発を目標とする。この目標に向けて、いくつかの NP 困難な最適化問題に対する近似アルゴリズムの設計を提案するとともに、いかにして限られた観測情報を扱うかを議論し、統計的かつ計算量的に良い性質を持つ汎用性の高いアルゴリズム開発を目指す。

組合せバンディット問題の中でも、最適腕識別と呼ばれる枠組みに焦点を当てる。最適腕識別問題は期待報酬が最大となる行動を見つける問題で、アルゴリズムの評価指標は累積報酬ではなく、出力までに用いた標本数、つまり標本複雑度が用いられる。本学位論文の主結果は第三章から第六章までの四つの章で議論されている。第三章では限定的な観測に基づく組合せ最適腕識別問題の中でも最も単純な問題である、線形の重み和が最大となるサイズ k の集合を求める問題を扱う。ここで観測として得られるのは、選択した集合からのノイズありの重み和のみである。この観測形態を、半バンディット観測に対して全バンディット観測と呼ぶ。第四章ではマトロイド、マッチング、パスなど幅広い組合せ制約にも適用可能なアルゴリズムを提案する。第五章ではグラフ上における最も密な部分グラフを検出することを目的とした、グラフ上の組合せ最適腕識別問題を扱う。第六章では部分観測と呼ばれる、半バンディットや全バンディットを特殊ケー

スとして含む観測形態を扱う設定を議論する．さらに報酬関数として非線形関数を扱える一般的な解法を提案する．以下にそれぞれの主結果を述べる．

第三章では、サイズ k 集合を求める全バンディット組合せ最適腕識別問題を扱う．この問題では、各時刻にサイズ k 集合を一つ選択したあとその確率的な重み和を観測し、なるべく少ない標本数で最適なサイズ k 集合を求めることを目指す問題である．既存の線形モデル上の最適腕識別問題への解法は、その計算量が全ての候補数に線形に依存するため、組合せの設定に対しては指数時間アルゴリズムとなってしまう．本研究では二次計画問題として定式化される信頼区間最大化に対して多項式時間近似アルゴリズムを設計し、計算量的にも統計的にも効率的なアルゴリズムを開発する．クラウドソーシングを用いた計算機実験でその有効性を検証する．

第四章では、サイズ、マトロイド、 s - t パス、マッチングなどを含む一般的な組合せ制約を扱える枠組みを提案する．第三章での提案手法が静的サンプリング戦略に基づいていたのに対し、本章では組合せ構造を利用した動的なサンプリング戦略に基づくアルゴリズムを設計する．動的なサンプリング戦略により、標本数の上界が最適値と次に良い値の差への依存度を軽減できることを理論的に示し、計算量および統計的に妥当であることを計算機実験により示す．

第五章では、最密部分グラフ抽出問題の全バンディット観測設定を扱う．既存手法のように毎回一本の枝からの観測を得るのではなく、枝集合からの重み和のみが観測される設定を考える．この問題に対し、高確率で最適解を求める多項式時間アルゴリズムを提案し、その標本複雑度を理論的に解析する．さらに、標本数があらかじめ決まっている設定に対して、貪欲法に基づく効率的なアルゴリズムを設計し、 $1/2$ 近似解を出力する確率の下界を示す．実グラフを用いた計算機実験により提案手法の妥当性を示す．

第六章では、非線形報酬関数と部分観測を同時に扱えるより一般的なモデルを提案する．推薦システムやクラウドソーシングといった実応用では、プライバシー保護やシステムの制約上、必ずしも選んだ集合に対する観測が得られるとは限らず、部分集合に対する観測しか得られない状況があり得る．また、報酬が線形ではない設定も多く存在する．そのような場合にも適応可能な多項式時間アルゴリズムを提案し、アルゴリズムが最適解を出力するのに必要な標本数の上界を示す．

以上を要するに、本学位論文では、限定的な観測に基づく組合せ最適腕識別問題に対する多項式時間アルゴリズムによる学習可能性を示し、組合せ構造を扱う複雑なオンライン意思決定問題や、入力が不完全な場合の不確実性を伴う組合せ最適化問題に対して新たな洞察を与えた．

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my supervisor Professor Masashi Sugiyama, for his guidance and patience throughout my PhD. I first learned about him when I came across his book at Tokyo Institute of Technology, and it invited me to a fascinating world of machine learning. I was very lucky to join his laboratory from the PhD program and learned a lot from him and his group. He had a great influence on me and sharpened my research mind indeed. Thanks to him, I had opportunities to meet many wonderful students and researchers coming from around the world, which was a great experience for me. Through my difficult times, he always managed to make time and supported me despite his extremely busy schedule. His hard work and enthusiasm for research have been my great inspiration, and encouraged me to pursue a career in research. For that, I am forever grateful to him.

I would also like to express my sincere gratitude to my another advisor Senior Lecturer Junya Honda, for his continuous assistance to my research. He kindly spent a great deal of time supporting me from the beginning of research projects throughout the paper writing or presentation stages. His technical assistance was especially helpful and necessary to complete my research. Without his careful and thoughtful guidance, my PhD would not have been achievable in any sense.

Also, I am very thankful to Associate Professor Issei Sato, Senior Lecturer Naoto Yokoya, Dr. Ikko Yamane, Dr. Yoshihiro Nagano, and the secretary Ms. Yuko Kawashima for their constant support to maintain the research environment. In particular, Associate Professor Issei Sato provided me with valuable advice for not only research, but also life. I deeply appreciate their help.

I am also very grateful to Professor Tetsuo Shibuya, Professor Seiya Imoto, Associate Professor Taiji Suzuki, Associate Professor Shinya Takamaeda and Professor Yusuke Miyao for participating in the thesis committee and providing me with valuable feedback and fruitful discussions.

My special thanks are due to Professor Wei Chen at Microsoft Research Asia, who is one of the pioneers in combinatorial bandits. I have read many his papers and his work has had great influence on my PhD studies. So I feel extremely lucky for having had the chance to work with him. Although I could not visit his laboratory, he has set up discussion every week and advised me remotely from Beijing. Also, I would like to thank Ms. Yihan Du at Tsinghua University, who is the other collaborator in MSRA projects. Besides the regular meetings, I have always discussed with her by WeChat and her hard work has motivated me to continue working on research even when we got stuck. Chapters 4 and 6 are based on the joint work with them, and I cannot thank them enough for their collaboration. I really want to meet them someday.

The collaboration research with MSRA came true thanks to Ms. Mawo Kamakura, Research Program Manager at Microsoft Research. She frankly com-

municated with me and gave me confidence. She also connected me to some Japanese female students in computer science, which was really helpful for me.

I would like to thank the other collaborators, Assistant Professor Atsushi Miyauchi and Mr. Liyuan Xu, for their valuable discussions and helpful comments. Assistant Professor Atsushi Miyauchi helped me develop theory and algorithms for graph optimization problems. Mr. Liyuan Xu helped me understand the best arm identification in linear bandits, and my first work of PhD studies is inspired by his paper.

For discussions on combinatorial optimization, I wish to express my gratitude to Professor Tomomi Matsui and Associate Professor Yasushi Kawase, who were my previous advisors at Tokyo Institute of Technology and first invited me to the academic world.

Working at Sugiyama-Sato-Honda (later Sugiyama-Honda-Yokoya) Lab was one of the most wonderful time in my life. I thank all the present and past members of the laboratory for the pleasant atmosphere. In the last three years, I have enjoyed not only research but also many activities including playing badminton or tennis, playing board games or cards, going to the beach, finding for good beer and nice restaurants, and everything. I am especially thankful to Kento Nozawa, Nontawat Charoenphakdee, Han Bao, Liyuan Xu, Takeshi Teshima, Jongyeong Lee, Taira Tsuchiya, Zijian Xu, Kenshin Abe, Jeonghyun Song, Dr. Ikko Yamane, Dr. Futoshi Futami, Seiichi Kuroki, Chenri Ni, and Hideaki Imamura. They have all helped me keep smiling, laughing and happy. I will never forget the wonderful time spent with all the lab members.

Last, but not least, I wish to thank my family for their love and support. I dedicate this thesis to my late grandfathers, whose dream was to see me complete my PhD.

Contents

Abstract	i
Acknowledgments	vi
List of Figures	x
List of Tables	xi
1 Introduction	1
1.1 Sequential Decision Making under Uncertainty	1
1.2 Major Challenges and Motivation	3
1.2.1 First Challenge: Decision Making with Combinatorial Ac- tions	4
1.2.2 Second Challenge: Learning with Limited Observation and Few Samples	5
1.3 Thesis Statement and Contributions	7
1.4 Thesis Organization	13
2 Preliminaries	15
2.1 General Notation	15
2.2 Basic Concept from Combinatorial Optimization	16
2.2.1 Running Time	16
2.2.2 Approximation Algorithms for Solving NP-Hard Problems	16
2.2.3 Matroids	17
2.3 Combinatorial Pure Exploration	17
2.3.1 Best Arm Identification Problem	17
2.3.2 Formulation of Combinatorial Pure Exploration	18
2.4 Related Problems: Best Arm Identification in Linear Bandits . . .	19
3 Top-k Arm Identification with Full-bandit Feedback	21
3.1 Introduction	21
3.1.1 Background	21
3.1.2 Chapter Contribution	23
3.1.3 Chapter Organization	23
3.2 Preliminaries	24
3.2.1 Problem Definition	24
3.2.2 Confidence Bound	24
3.3 Computational Hardness	25
3.4 Static Allocation Strategies	26
3.5 Naive Method by Simplified Confidence Bounds	27

3.6	Confidence Ellipsoid Maximization	29
3.7	Static Allocation Algorithm with Quadratic Maximization	30
3.8	Heuristic Algorithms	32
3.9	Experiments	35
3.9.1	Details of Baseline Algorithms	35
3.9.2	Experimental Settings and Results	41
3.10	Proofs of Theoretical Results	44
3.10.1	Proof of Lemma 3.3	44
3.10.2	Proof of Theorem 3.1	46
3.10.3	Proof of Theorem 3.2	48
3.10.4	Proof of Theorem 3.3	48
3.11	Conclusion	51
4	Combinatorial Pure Exploration with Full-bandit Feedback under General Constraints	52
4.1	Problem Statement	52
4.2	Related Work and Chapter Contribution	53
4.3	Naive Reduction of the Top- k Case and Analysis of a UCB-based Algorithm for CPE-BL	55
4.3.1	Regularized Least Square Estimator	56
4.3.2	Analysis of CLUCB for CPE-BL	57
4.4	Adaptive Algorithm for CPE-BL	58
4.5	Theoretical Analysis of PolyALBA	60
4.5.1	Analysis of the Statistical and Computational Efficiency	61
4.5.2	Discussion on the Optimality	62
4.6	Discussion on Improving α	64
4.7	Experiments	67
4.7.1	Experiments on the Matching Instances	67
4.7.2	Experiments on the Top- k Instances	68
4.8	Proofs of Theoretical Results	69
4.8.1	Proof of Corollary 4.1	69
4.8.2	Proof of Lemma 4.2	71
4.8.3	Technical lemmas for Theorem 4.1	72
4.8.4	Proof of Theorem 4.1	74
4.8.5	Proof of Lemma 4.4	75
4.9	Conclusion	76
5	Online Dense Subgraph Discovery via Linear Feedback	77
5.1	Introduction	77
5.1.1	Dense Subgraph Discovery	77
5.1.2	Chapter Contribution	78
5.1.3	Related Work	79
5.2	Problem Statement	80
5.2.1	Densest Subgraph Problem	80
5.2.2	Densest Subgraph Bandits (DS Bandits)	81
5.3	LP-based Exact Algorithm for the Densest Subgraph Problem . .	81
5.4	Algorithm for DS Bandits with No Access to Single Edges	82
5.4.1	DS-Lin Algorithm	82
5.4.2	Sample Complexity	86

5.5	Algorithm for DS Bandits with a Fixed Budget	87
5.5.1	DS-SR Algorithm	87
5.5.2	Upper Bound on Probability of Error	89
5.6	Experiments on Real-world Graphs	89
5.6.1	Experiments for DS-Lin	95
5.6.2	Experiments for DS-SR	96
5.7	Proofs of Theopretical Results	99
5.7.1	Technical Lemmas for Theorem 5.1	99
5.7.2	Proof of Theorem 5.1	100
5.7.3	Technical lemmas for Theorem 5.2	104
5.7.4	Proof of Theorem 5.2	106
5.8	Conclusion	108
6	Combinatorial Pure Exploration with Partial-linear Feedback for Nonlinear Rewards	109
6.1	Preliminaries	109
6.1.1	Problem Statement	109
6.1.2	Assumptions	110
6.1.3	Illustration Examples	110
6.2	Related Work and Chapter Contribution	112
6.3	Static Algorithm for CPE-PL	113
6.4	Theoretical Analysis of GCB-PE	114
6.5	Application Examples	115
6.6	Experiments	116
6.7	Proofs of Theoretical Results	117
6.7.1	Proof of Lemma 6.1	117
6.7.2	Proof of Theorem 6.1	117
6.8	Conclusion	120
7	Conclusion and Future Directions	121
7.1	Summary	121
7.2	Future Directions	122
7.2.1	Lower Bounds of Polynomial-time δ -PAC Algorithms . . .	122
7.2.2	Non-Stationary Environment	123
7.2.3	Nonlinear Reward Functions and Nonlinear Feedback . . .	123
7.2.4	Optimal Experimental Design with Combinatorial Actions	124
	Bibliography	137

List of Figures

1.1	An illustration of reinforcement learning, or multi-armed bandit problems.	2
1.2	The examples of combinatorial structures in networks.	10
1.3	The process of dense subgraph discovery.	11
1.4	The roadmap of this thesis.	14
3.1	The results for approximation precision on synthetic datasets, where we set $(d, k) = (10, 5)$. Each point corresponds to an average over 10 realizations.	36
3.2	Run time in each round for synthetic datasets. Each point is an average over 10 realizations.	37
3.3	Number of samples for synthetic datasets with $(d, k) = (10, 5)$. Each point is an average over 10 realizations.	37
4.1	Experimental results of running time and sample complexity for CPE-BL.	67
4.2	Experimental results of running time and sample complexity for full-bandit top- k instances.	68
5.1	Results for the behavior of our proposed algorithm with respect to the number of iterations. Each point is an average over 10 runs of the algorithm.	92
5.2	Results for the estimation of the expected weight w . Each point is an average over 10 runs of the algorithm.	93
5.3	Fraction of the size of queried edge subsets in DS-SR (cumulative) over 100 runs.	98
6.1	Example of the crowdsourcing application.	111
6.2	Experimental results of running time and sample complexity for CPE-PL.	116
7.1	The illustration of a confidence ellipsoid, where $X = (x_1, \dots, x_d)$ is the model matrix. The shape of the ellipsoid depends on the information matrix $X^T X$	125

List of Tables

3.1	Real-world datasets on crowdsourcing. “Average” and “Best” give the average and the best accuracy rate among the workers, respectively.	42
3.2	Number of samples ($\times 10^2$) on real-world crowdsourcing datasets. Each value is an average over 10 realizations.	42
4.1	Comparison between our results and existing results for CPE-BL and BAI-LB. “General” represents that the algorithm works for combinatorial structures including size- k subsets, paths, matchings, and matroids. $\tilde{O}(\cdot)$ only omits log log factors. Main notation is defined in Section 4.1, and other specific notation are given in the footnote.	54
5.1	Notation.	82
5.2	Real-world graphs used in our experiments.	90
5.3	Comparison between DS-Lin and the baseline algorithm (Algorithm 5.5).	91
5.4	Performance of DS-SR. For DS-SR and R-Oracle, the quality of solutions, number of samples, and computation time are averaged over 100 executions.	94

Chapter 1

Introduction

The world is rife with uncertainty. We human beings are constantly exposed to risk or ambiguous situations; we live with uncertainties such as daily variations in the weather or the stock market, and we are also involved in the uncertainties of fatal events such as natural disaster, climate change, or economic recession. In the face of uncertainty in our lives, we aim to take the best possible decision based on the conjecture and past experience. *Decision making under uncertainty* is common in prudent weighing of outcomes or utility. Modeling of decision making under uncertainty has been analyzed in diverse research fields such as computer science, statistics, psychology, and neuroscience.

1.1 Sequential Decision Making under Uncertainty

The origin of *artificial intelligence* dates back to 1950 [Turing, 1950] with his initial question “Can machines think?”. Since then, there has been a big surge of interest in *machine learning*, that studies how computers can achieve human or super-human level performance on various tasks, and its great success has made new technologies such as facial recognition system [Bartlett et al., 2005], self-driving cars [Bojarski et al., 2016], conversational assistants [Lather, 1991], just to mention a few. Machine learning is a primary mechanism for extracting patterns from data, which is built on statistics and probability theory. However, there are different purposes between statistics and machine learning; statistics mainly focuses on understanding the past and statistical models are designed for indentifying inference, whereas machine learning puts emphasis on making predictions/decisions to the “unknown” in the future.

Sequential decision making under uncertainty abounds in real-world problems ranging from game playing and robotics to resource management, packet routing, financial portfolio management, medical treatment design, and online advertisement. *Reinforcement learning* [Bertsekas and Tsitsiklis, 1996, Sutton and Barto, 1998], which investigates how a computer *agent* can learn the best behavior from past experience by interacting with an unknown environment, is often considered as one of the general framework for solving such a challenging decision making problems [Mnih et al., 2015]. In reinforcement learning, an agent takes one *action* based on its own *policy* that maps each state of the environment to an action. One time step later, the *reward* signal corresponding to the taken action is provided from the environment (see Figure 1.1). The goal of reinforcement learning is to learn a good policy to maximize the cumulative

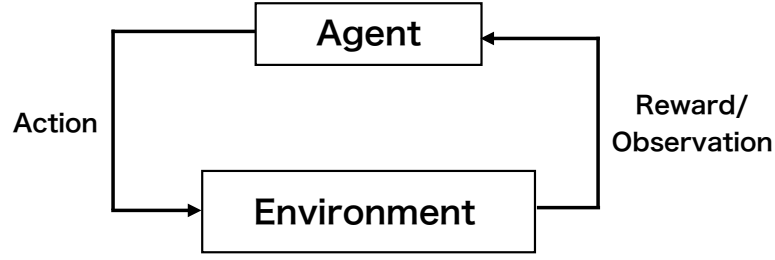


Figure 1.1: An illustration of reinforcement learning, or multi-armed bandit problems.

reward in the long run.

This is a contrast to the *supervised learning* scenario, in which training examples come all at once explicitly containing correct answers as supervision. On the other hand, in reinforcement learning, an agent has to learn from samples collected by sequential interaction with the unknown environment, and the environment does not tell explicitly what would be the best action. This characteristic naturally leads to the following *exploration-exploitation dilemma*. The agent is required to search the space of possible decisions for potential future benefit (exploration). At the same time, she must choose the seemingly optimal decision according to the past observation (exploitation). To understand such a trade-off between exploration and exploitation, the area of *multi-armed bandit problems* (MAB), which involves only one state, has received much attentions in the literature as the cornerstone of the stateful reinforcement learning.

The first idea of MAB appeared early in the 20th century [Thompson, 1933], motivated by the medical treatment design. The popularization of MAB as a sequential decision making model would have been realized by the seminal work of Robbins [1952] and Lai and Robbins [1985]. The model of MAB is described as the following learning problem. Suppose that there are K possible actions, whose utility is unknown, and each action is associated with to an unknown probability distribution. An action is usually called an *arm*, which characterizes the decision making problem of the agent in the following stochastic game with discrete time steps. At each round $t = 1, 2, \dots$, the agent must choose an arm to pull from K arms. When each arm i is pulled, the agent can observe a stochastic reward sampled from an unknown probability distribution. The most well-studied objective is to minimize the *cumulative regret*, i.e., the total loss between the expected reward of the optimal arm and the expected reward of the arm collected by the agent [Bubeck and Cesa-Bianchi, 2012, Cesa-Bianchi and Lugosi, 2006].

Another popular objective is to identify the best arm, i.e., the arm with the maximum expected reward among K arms by interacting the unknown environment. This problem, called *pure exploration* or *best arm identification*¹ in the MAB, has also received much attention recently [Even-Dar et al., 2002, 2006, Audibert et al., 2010, Jamieson et al., 2014, Chen and Li, 2015, Kaufmann et al., 2016]. In pure exploration problems, we do not care about regrets in a test phase. In the so-called *fixed confidence setting*, given a confidence level $\delta \in (0, 1)$, the

¹In general, pure exploration is a wider concept than best arm identification, but we use these terms interchangeably in this thesis.

agent must provide the optimal action with probability at least $1 - \delta$, and a learning algorithm is evaluated by its *sample complexity*, which is the number of samples an algorithm requires to output the optimal arm. In the so-called *fixed budget setting*, the number of trials must not exceed the given budget T and the agent tries to identify the optimal action. A learning algorithm is evaluated by the *probability of error*, which is the probability that an output answered by the algorithm is not the optimal action.

1.2 Major Challenges and Motivation

Despite modern developments of reinforcement learning and MAB over nearly a century, *combinatorial actions* pose a challenge to these fields. Algorithms for reinforcement learning or bandit problems typically require an action space to be small enough to enumerate, and how to deal with combinatorially large action space has been overlooked until recently in the literature [Sugiyama, 2015, Lattimore and Szepesvári, 2020]. In many real-world scenarios, indeed, our decisions are often characterized by a combinatorial and complex structure. For example, possible actions in real-world systems may be a subset of keywords in online advertisements [Rusmevichientong and Williamson, 2006], assignments of tasks to workers in crowdsourcing [Zhou et al., 2014], or channel selection in communication networks [Huang et al., 2008]. In traditional combinatorial optimization models, we commonly assume that the input parameters such as edge weights in a graph are exactly known and many algorithms have been developed under this assumption. However, input parameters in the problems might not be obtained precisely in the real world, and thus they have to be learned by collecting and observing (possibly noisy) data. To overcome this issue, *combinatorial bandit* problems, a generalization of MAB, have been investigated recently [Cesa-Bianchi and Lugosi, 2006]. However, much of the research still require strong feedback that reveals outcomes from individual arms to handle computational issues, which might not be possible in real-world applications [Chen et al., 2013, 2016b].

To deal with a complicated and combinatorially large action space, decision making algorithms should be designed to avoid computational issues. At the same time, statistical efficiency should also be taken into consideration when designing practical algorithms for real-world applications, which do not require strong observation or a prohibitively large number of samples. Therefore, major challenges are summarized as follows.

1. *Design a computationally efficient algorithm for decision making problems in combinatorially large action spaces.*
2. *Design a statistically efficient algorithm for decision making problems only from limited observation and few samples.*

To formalize these challenges, we focus on combinatorial bandit problems with limited feedback in this thesis. In what follows, we will describe our challenges above and motivation for studying combinatorial bandits in more detail.

1.2.1 First Challenge: Decision Making with Combinatorial Actions

Our decisions are often conducted in the combinatorial nature. For example, each action may be a size- k subset of keywords in online advertisement [Rusmevichientong and Williamson, 2006], an assignment between workers and tasks in crowdsourcing [Lin et al., 2014], or a spanning tree in communication networks [Huang et al., 2008]. *Combinatorial optimization* is one of the fundamental research fields that has extensively studied in theoretical computer science and operations research Korte and Vygen [2012]. A large number of combinatorial optimization problems have been proposed and its methodological design has provided an insight on how to design a computationally efficient method with combinatorial actions. However, most existing models typically require the exact parameters as inputs, which are uncertain or unknown in many modern applications such as recommender systems, crowdsourcing, and online advertisement.

Over the past decade, mathematical optimization models that are immune to data uncertainty have been studied in the field of *robust optimization* [Ben-Tal et al., 2009, Bertsimas et al., 2011]. In the robust optimization paradigm, uncertainty sets may often be modeled as deterministic sets such as boxes, polyhedra, or ellipsoids. The quality of a solution is then evaluated using the realization of the uncertainty that is most unfavorable for the decision maker. That is, robust optimization considers the worst-case scenario given uncertainty sets, and does not discuss how to design optimal sampling strategies in combinatorial action spaces to resolve the uncertainty.

Combinatorial decisions often cooccur with the notion of *networks* or *graphs*. Networks are ubiquitous in the real world and many complex phenomenon or systems in nature and society can be captured by networks. A *graph* $G = (V, E)$ is a mathematical representation of a network and it consists of a set of vertices V , and a set of edges E between pair of vertices. A *weighted graph* $G = (V, E, w)$ is a graph where each edge has a numerical weight, i.e., $w : E \rightarrow \mathbb{R}$. These mathematical notions are very general and intuitive, which represent a wide variety of real-world systems. Some examples include:

- *Technological networks* [Broder et al., 2000]: The Internet topology such as the World Wide Web and transportation networks is a graph where vertices represent web pages/routers and edges represent connection between pairs of them. The edges could be weighted by similarity between two nodes or costs for routing.
- *Social networks* [Scott and Carrington, 2011]: In friendship networks and co-workers networks, individuals are vertices and ties such as friendship or business relationship between people are edges. The edges could be weighted by the strength of connection or similarity between two users or persons.
- *Biological networks* [Alon, 2003]: In protein interaction networks, vertices correspond to proteins in a cell and edges connect two proteins if both are necessary for some biological process to occur. Edge weights represent the strength of interactions among two proteins.

With the growth of the World Wide Web and with the development of online platforms, computational efficiency in algorithm design for decision making

must be prioritized in order to handle large-sized datasets or networks. For instance, Facebook has over 2.5 billion users in 2020²; YouTube also reaches over 2 billion users and more than 500 hours of content are uploaded every minute in 2020³. While many researchers in machine learning have made effort to design sophisticated methods for various types of sequential decision making problems [Sugiyama, 2015, Lattimore and Szepesvári, 2020], most existing methods are hard to scale up to significantly large or combinatorial action spaces. Indeed, most of them typically require an action space small enough to enumerate or solving complex optimization to design a good policy [Delarue et al., 2020], which might be NP-hard in combinatorial settings.

The problem of combinatorial bandits [Cesa-Bianchi and Lugosi, 2006] is a generalization of MAB, which considers combinatorial actions; a subset of underlying arms, called a *super arm*, is an action in this model, while each single arm is an action in the MAB problem. Combinatorial bandit problems with a linear reward function can be seen as a special class of *linear bandit* problems. In linear bandits, each arm has its own feature $x \in \mathbb{R}^d$, while in combinatorial settings, each action can be associated with a vector $x \in \{0, 1\}^d$. One might think that algorithms for linear bandits can deal with combinatorial action spaces. Most linear bandit algorithms have, however, the time complexity at least proportional to the number of arms [Dani et al., 2008, Abbasi-Yadkori et al., 2011, Jun et al., 2017]. Therefore, a naive use of them is computationally infeasible since the number of actions K is exponential. To increase the applicability of decision making algorithms to the real world, there is a need for scalable systems that can deal with combinatorial actions.

1.2.2 Second Challenge: Learning with Limited Observation and Few Samples

In the stochastic combinatorial bandit framework, outcomes from super arms are assumed to follow unknown probability distributions. In each round, one super arm is pulled and the outcomes depending on the pulled super arm are revealed. This learning paradigm would be a strong tool for handling combinatorial decision making with incomplete data or inaccurate measurements, and has attracted much attention [Chen et al., 2013]. Adversarial cases, in which an adversary controls the arms and tries to defeat the learning process, have also been studied in the literature [Cesa-Bianchi and Lugosi, 2012, Combes et al., 2015]. Adversarial cases with linear objectives have also been studied as *online linear optimization* [Abernethy et al., 2008].

Early attempts in combinatorial bandits assume that an agent can observe the outcomes of all arms in the super arm, which is called *semi-bandit* feedback (e.g. [Chen et al., 2013, Kveton et al., 2015, Chen et al., 2016c, Wen et al., 2017, Perrault et al., 2019]). To be more precisely, the setup of stochastic combinatorial bandits and feedback models with a linear objective are described as follows. Let $[d] = \{1, 2, \dots, d\}$ be a set of base arms. Given combinatorial constraints, the set of arms $\mathcal{X} \subseteq \{0, 1\}^d$ is a family of super arms, where each element $x \in \{0, 1\}^d$ is an indicator vector of a super arm satisfying the given

²<https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/>

³<https://www.businessofapps.com/data/youtube-statistics/>

combinatorial structure such as a size- k subset, path, or matching. Let $X_t(e)$ for $e \in [d]$ be a random variable at round t independently sampled from the associated unknown distribution. In each round t , an agent selects a super arm $x_t \in \mathcal{X}$ and observes some feedback. Here, the reward vector $X_t \in \mathbb{R}^d$ has its e -th element with $X_t(e) = \theta(e) + \epsilon_t(e)$, where $\theta(e)$ is the expected reward and $\epsilon_t(e)$ is the zero-mean noise. The feedback can be categorized into the following types.

- (i) *Semi-bandit feedback*: After pulling a super arm x_t at round t , the component $X_t(e)$ for $e \in [d]$ is observed if and only if $x_t(e) = 1$.
- (ii) *Full-bandit feedback*: After pulling a super arm x_t at round t , only the sum of rewards $x_t^\top X_t$ is observed.
- (iii) *Partial-linear feedback*: After pulling a super arm x_t at round t , only $M_{x_t} \cdot X_t$ is observed, where M_{x_t} is a transformation matrix $\mathbb{R}^{m_{x_t} \times d}$ whose row dimension m_{x_t} depends on x_t . That is, the agent only observes a linear combination of rewards of x_t .

Most prior work have studied the semi-bandit feedback or assumed that the outcome from each individual edge can be always accessible at all rounds [Chen et al., 2014, 2016a, Gabillon et al., 2016, Jun, Chen et al., 2017, Huang et al., 2018, Cao and Krishnamurthy, 2019]. However, in most application domains, such strong feedback is not always available, since it is costly to observe a reward of individual arms, or sometimes we cannot access feedback from individual arms. For example, in crowdsourcing, we often obtain a lot of labels given by crowdworkers, but it is costly to compile labels according to labelers. In the case of transportation networks, it is not easy to observe a delay of each section of a path due to some system constraints. In social networks, due to some privacy concerns and data usage agreements, it may be impossible even for data owners to obtain the estimated number of messages exchanged by two specific users. To overcome these issues, algorithms that can deal with full-bandit feedback or partial-linear feedback are desired. However, there are very few studies even for the full-bandit feedback in the combinatorial bandit literature [Talebi Mazraeh Shahi, 2016].

Why is dealing with such limited feedback hard? Stochastic bandit problems have been analyzed by two different approaches: a frequentist approach, where the parameter is a deterministic unknown quantity, and a Bayesian approach, where the parameter is drawn from a prior distribution. Vast majority of combinatorial and linear bandit work follow a frequentist approach. They often employ the optimism principle exemplified by *upper confidence bound* (UCB) algorithm [Abbasi-Yadkori et al., 2011, Chen et al., 2016b], which uses the data observed so far to assign to each arm a value, called the UCB that is an overestimate of the unknown mean with high probability.

To handle full-bandit feedback, the frequentist approach may rely on the least-square estimator, and the combinatorial structure results in a confidence region in the form of an ellipsoid. With a confidence ellipsoid, algorithms often require complex optimization for determining the next arm to pull, which might involve quadratic optimization with combinatorial constraints. All the existing linear bandit algorithms execute a brute-force search to solve such an

optimization problem and thus they cannot be naively applied to combinatorial bandit problems. Therefore, we need to develop different techniques to deal with full-bandit feedback in combinatorial settings.

The model of partial-linear feedback addresses general limited feedback that ranges from semi-bandit to full-bandit feedback. For regret minimization, several researchers investigated UCB-based algorithms [Lin et al., 2014, Chaudhuri and Tewari, 2016]. On the other hand, for the pure exploration problems, there is no existing study and it has been open to investigate which combinatorial problems are still polynomial-time learnable.

For regret minimization, many algorithms in various settings are already shown to achieve sublinear regret bounds in time horizon T and the achievable theoretical limit is becoming now well known. On the other hand, much remains to be done for the pure exploration problems [Kaufmann et al., 2016]. The pure exploration problem models situations in which costs are not measured in terms of rewards but rather in terms of resources [Bubeck et al., 2011]. Since we aim to design statistically efficient algorithms that require less samples while guaranteeing that the output is optimal in the pure exploration problem, good algorithms for pure exploration would require quite different strategies from those for regret minimization. It is important to care about the dependence of the minimum gap Δ_{\min} , i.e., the reward gap between the best action and the second best one; this quantity is crucial in the sample complexity analysis for pure exploration problems. In many practical applications, Δ_{\min} is often too small [Kawase et al., 2019], and heavy dependence on $1/\Delta_{\min}$ means that a large number of samples are required to identify the optimal action. Therefore, we wish to develop algorithms whose sample complexity only has mild dependence on $1/\Delta_{\min}$.

1.3 Thesis Statement and Contributions

In this thesis, we study several stochastic combinatorial bandit problems on the pure exploration settings. This thesis provides evidence to support the following statement:

Thesis Statement

One can efficiently solve combinatorial decision making problems even when input parameters are initially unknown and only limited observation is available.

We believe that the results developed in this thesis will motivate future researchers in both optimization and bandit communities to develop algorithms for combinatorial decision making under uncertainty. In order to substantiate this statement, we present the following contributions.

Chapter 3: Top- k Arm Identification with Full-bandit Feedback

Problem and Motivation. We start with the simplest case in terms of combinatorial structures. We study the problem of stochastic *top- k arm identification*, where an agent sequentially explores a size- k subset of arms (super arm) from given d arms and tries to identify the best super arm. The problem is also called the *top- k selection* or *k -best arm identification*, and there is an emerging body of work in the bandit literature [Bubeck et al., 2013, Cao et al., 2015, Gabillon et al., 2011b, 2012, Kalyanakrishnan and Stone, 2010, Kalyanakrishnan et al., 2012, Kaufmann and Kalyanakrishnan, 2013, Roy Chaudhuri and Kalyanakrishnan, 2017, Chaudhuri and Kalyanakrishnan, 2019, Zhou et al., 2014]. In the above prior work, we need a strong assumption that the agent is allowed to pull a base arm and can observe its outcome directly at all rounds, which might not be possible in real-world scenarios. Our study is the very first work to analyze the top- k arm identification with full-bandit feedback, and we find a variety of applications for full-bandit settings. Some application examples are:

- *Online advertisement auctions:* Companies may want to choose multiple keywords to promote their products to consumers based on their search queries. From millions of available choices, a company aims to find the most effective set of keywords by observing the historical performance of the chosen keywords. In such application domains, we may be interested in best- k keywords and only the total sum of clicks can be sequentially observed. This decision making problem is formulated as top- k arm identification with full-bandit feedback, where each arm corresponds to each keyword and reward corresponds to the effectiveness or utility of a keyword.
- *Extracting experts in crowdsourcing:* A system-owner or employer may often obtain a lot of labels given by crowdworkers, but it is costly to compile labels according to labelers. Furthermore, in software projects, an employer may have complicated tasks that need multiple workers, in which the employer can only evaluate the quality of a completed task rather than the performance of a single worker. In such scenarios, we wish to extract top- k expert workers who can perform the task with high quality, only from a sequential access to the quality of the task completed by multiple workers.

Contribution. Although our problem can be regarded as an instance of the best arm identification problem in linear bandits, a naive approach based on linear bandits is computationally infeasible since the number of super arms K is exponential. We first show that quadratic optimization with the size constraint involved in existing linear bandit algorithms is naturally stated as a uniform knapsack problem, which is NP-hard. To cope with this computational issue, we first design a polynomial-time approximation algorithm for a 0-1 quadratic programming problem to obtain the maximum width of a confidence ellipsoid. Based on our approximation algorithm, we propose the first bandit algorithms, namely **SAQM**, that runs in $O(\log K)$ time and provide an upper bound of the sample complexity, which is still worst-case optimal. The result indicates that our algorithm provides an exponential speedup over an exhaustive search algorithm while keeping the statistical efficiency. We also design two heuristic al-

gorithms that empirically perform well: SA-FOA using first-order approximation and CLUCB-QM based on the lower-upper confidence bound algorithm. Finally, we conduct experiments on synthetic and real-world datasets with more than 10^{10} super arms, demonstrating the superiority of our algorithms in terms of both the computation time and the sample complexity.

Chapter 4: Combinatorial Pure Exploration with Full-bandit Feedback under General Constraints

Problem and Motivation. How can we go beyond the top- k setting in the full-bandit setup? We next seek how to deal with more complex combinatorial constraints such as paths, spanning trees, and matchings. In the offline settings, such structures are very common and fundamental combinatorial optimization problems such as the shortest path, the minimum spanning tree, and the weighted matching problems have been extensively studied in theoretical computer science (definitions of these problems will be introduced below). For the top- k case in the bandit setting, we find that there exists a naive reduction to the classical combinatorial pure exploration problem, in which the agent directly samples a base arm at each round. However, for more complex cases such as matroids, matchings, and s - t paths, we cannot use such a reduction due to their combinatorial constraints. To increase the applicability of bandit methods, we propose a novel algorithm for *combinatorial pure exploration with full-bandit linear feedback (CPE-BL)*, where underlying combinatorial structures are matroids, matchings, and s - t paths and the reward function is a linear objective. Some examples of offline combinatorial problems in CPE-BL are as follows.

- The *shortest path problem*: Given a directed graph $G = (V, E)$ and a cost (or time length) θ_e assigned to each edge $e \in E$, the shortest path problem is to find the path P from origin node s to destination node t that minimizes the sum of costs $\sum_{e \in P} \theta_e$.
- The *minimum spanning tree problem*: A spanning tree on a graph is a set of edges which connects all the vertices. Given an undirected graph $G = (V, E)$ with a cost θ_e assigned to each edge $e \in E$, the minimum spanning tree problem is to find the spanning tree T that minimizes the sum of costs $\sum_{e \in T} \theta_e$.
- The *weighted matching problems*: A matching on a graph is a set of vertex-disjoint edges. Suppose that we are given an undirected graph $G = (V, E)$ with a weight θ_e assigned to each edge $e \in E$. The *minimum cost matching* problem is to find the matching that minimizes the sum of costs $\sum_{e \in P} \theta_e$. The *maximum weight matching problem* is to find the matching that maximizes the sum of weights $\sum_{e \in P} \theta_e$. A special case of the problem is the *assignment problem*, in which the given graph is a bipartite graph.

These combinatorial optimization problems are very fundamental and have a rich history, and several polynomial-time algorithms are known to these problems [Dijkstra et al., 1959, Edmonds, 1965, Korte and Vygen, 2012]. In real-world scenarios, however, input parameters such as the cost θ_e for each $e \in E$ of the combinatorial problems above are often unknown and have uncertainty.

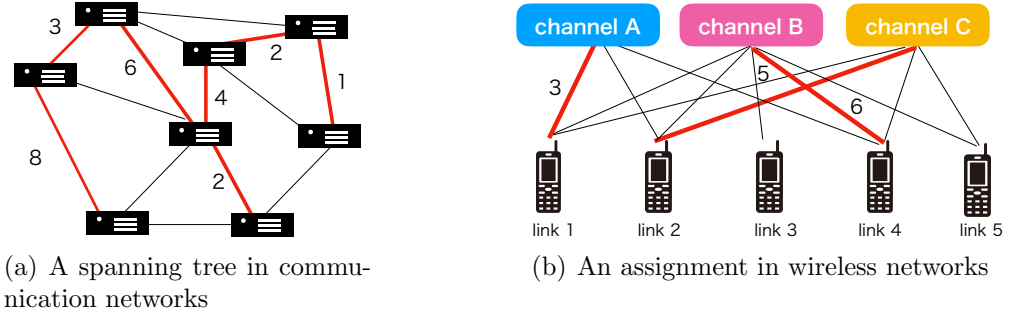


Figure 1.2: The examples of combinatorial structures in networks.

Our learning model of CPE-BL can capture these situations, in which a base arm corresponds to an edge, and rewards correspond to costs. For example, in the case of communication networks, a decision maker may not get access to the exact information of transmission delay for each arc. In this scenario, decision maker sequentially selects a routing that forms required structures such as a path, spanning tree, or matching in communication networks. Then the decision maker tries to identify the best routing as soon as possible only with a sequential access to the total cost for the chosen routing (see Figure 1.2 for application examples).

A *static* algorithm, which samples super arms based on the predetermined arm-selection ratio without any observation, might be useful for reducing computational costs. However, such a static sampling strategy may degrade the sample complexity and result in heavy dependence on the minimal gap. On the other hand, an *adaptive* algorithm, which changes sampling strategies based on past observations, may require more complex minimax optimization, which is intractable in the combinatorial action space.

Contribution. We begin with a discussion on a fully-adaptive algorithm for CPE-BL and the result depends on a non-controllable term (i.e., the minimum eigenvalue of the information matrix), indicating that fully-adaptive control among all super arms may be difficult. As a remedy for these issues, we propose an algorithm, namely **PolyALBA**, in which we have two phases: the first phase is for finding a polynomial-size set of super arms which contains the optimal super arm with high probability and discards other super arms. The second phase focuses on sampling super arms among the rest by adaptive elimination procedures. To the best of our knowledge, our **PolyALBA** is the first adaptive and polynomial-time algorithm that works for general combinatorial structures and avoids heavy dependence on the minimum gap. We show that the sample complexity of **PolyALBA** matches (within a logarithmic factor) the information theoretic lower bound for a family of instances. We conduct a series of experiments for CPE-BL on the matching and top- k instances. The experimental results demonstrate that (a) **PolyALBA** scales to combinatorial instances while existing linear bandit algorithms require prohibitive amount of time; (b) the number of samples required by **PolyALBA** is not sensitive to the value of the minimum gap.

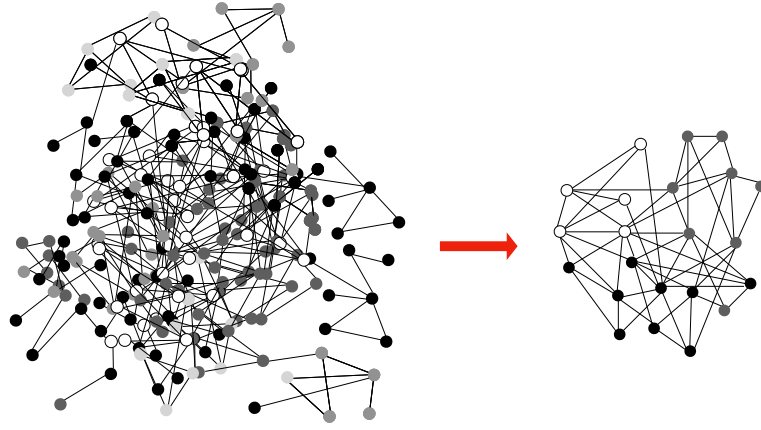


Figure 1.3: The process of dense subgraph discovery.

Chapter 5: Online Dense Subgraph Discovery via Linear Feedback

Problem and Motivation. What if the reward function is no longer linear in combinatorial pure exploration problems? Can we utilize the property of graph structures to resolve non-linearity? We study a bandit model of the *densest subgraph problem*, one of the most fundamental graph optimization problems with nonlinear objective functions, as a cornerstone to investigate these questions.

Dense subgraph discovery aims to find a dense component in edge-weighted graphs and this is a fundamental task for graph mining with a variety of applications [Dourisboure et al., 2007, Gibson et al., 2005, Bader and Hogue, 2003, Kawase et al., 2019, Angel et al., 2012] (see Figure 1.3). In the densest subgraph problem, given an edge-weighted undirected graph, we are asked to find a subset of vertices that maximizes the so-called *degree density*, which is defined as half the average degree of the subgraph induced by the subset.

Whereas most existing algorithms for the densest subgraph problem require a full input of the graph data, in many real-world applications, the edge weights need to be estimated from *uncertain* measurements. A measurement for individual edges is often quite costly or sometimes impossible due to privacy concerns or system constraints. On the other hand, it is often affordable to observe aggregated information of a subset of edges, because this procedure reveals much less information of individual users or requires less cost to conduct [Agrawal and Srikant, 2000, Zheleva and Getoor, 2011]. For such dense subgraph discovery under uncertainty, we introduce a novel learning problem by incorporating the concepts of stochastic combinatorial bandits into the densest subgraph problem. In this problem, an agent is given an undirected graph, whose edge weights are associated with unknown probability distributions. During the exploration period, the agent chooses a subset of edges (rather than a single edge) to sample, and observes the sum of noisy edge weights in the queried subset. After the exploration period is over, the agent must report one subgraph that she believes to be optimal.

Contribution. We study the both fixed confidence and fixed budget settings. For the former setting, we present an algorithm DS-Lin; for its design, we follow techniques of pure exploration in linear bandits and provide a sample complexity bound, which is the first polynomial-time sample complexity result for

this problem. Our key idea is to utilize an approximation algorithm for unconstrained quadratic optimization to compute the maximal confidence bound, thereby guaranteeing that the output by DS-Lin is an ϵ -optimal solution and the running time is polynomial in the size of a given graph. For the fixed budget setting, we design a scalable and parameter-free algorithm DS-SR that runs in $O(|V|^2T)$ time complexity for the given budget T , while DS-Lin needs $O(|E|^2)$ time for updating the estimate. Our key idea is to combine the *successive reject* strategy [Audibert et al., 2010] for the multi-armed bandits and the *greedy peeling* algorithm [Charikar, 2000] for the densest subgraph problem. We prove an upper bound on the probability that DS-SR outputs a solution whose degree density is less than $\frac{1}{2}\text{OPT} - \epsilon$, where OPT is the optimal value. In a series of experiments, we thoroughly evaluate the performance of our proposed algorithms using well-known real-world graphs. We confirm that DS-Lin obtains a nearly-optimal solution even if the minimum size of queryable subsets is larger than the size of an optimal subset, which is consistent with the theoretical analysis. In the fixed budget setting, we implement a robust optimization algorithm by Miyauchi and Takeda [2018] as a baseline and observe that it requires a large number of samples for single edges. We demonstrate that DS-SR finds nearly-optimal solutions even for large-sized instances, while significantly reducing the number of samples for single edges.

Chapter 6: Combinatorial Pure Exploration with Partial-Linear Feedback for Nonlinear Rewards

Problem and Motivation. Although full-bandit settings can capture many practical situations as demonstrated in Chapters 3–5, it may happen that we cannot always observe outcomes from some of the chosen arms due to privacy concerns or system constraints. To overcome this issue, we propose a general model of *combinatorial pure exploration with partial-monitoring linear feedback* (CPE-PL), which simultaneously models limited feedback, general (possibly nonlinear) reward and combinatorial action space. The model subsumes our previous problems addressed in Chapters 3–5. In CPE-PL, we are given a combinatorial action space $\mathcal{X} \subseteq \{0, 1\}^d$, where each dimension corresponds to a base arm and each action $x \in \mathcal{X}$ is an indicator vector of a super arm. At each round, the agent chooses an action (super arm) $x_t \in \mathcal{X}$ to pull and observes a random partial-linear feedback with expectation of $M_{x_t}\theta$, where M_{x_t} is a transformation matrix for x_t and $\theta \in \mathbb{R}^d$ is an unknown environment vector. The CPE-PL framework includes CPE-BL as its important subproblem; in CPE-BL, the agent observes full-bandit feedback (i.e., $M_x = x^\top$) and gains linear reward (with expectation of $x^\top\theta$) after each play. The model of CPE-PL appears in many practical scenarios, including:

- *Learning to rank.* Suppose that a company (agent) wishes to recommend their products to users by presenting the ranked list of items. Collecting a large amount of data on the relevance of all items might be infeasible, but the relevance of a small subset of items which are highly-ranked (or top-ranked item) is reasonable to obtain [Chaudhuri and Tewari, 2015, 2016, 2017]. In this scenario, the agent selects a ranked list of entire items at each step, and observes random partial-linear feedback on the relevance of

highly-ranked $d' \ll d$ items. The objective is to identify the best ranking of their whole items with as few samples as possible.

- *Task assignment in crowdsourcing.* Suppose that an employer wishes to assign crowdworkers to tasks with high quality performance. It might be costly for the employer and workers to provide task performance feedback for all tasks [Lin et al., 2014], and privacy issues may also arise. In this scenario, the agent sequentially chooses an assignment of workers to tasks and observes random partial-linear feedback on a small subset of completed tasks. The objective is to find the matching between workers and tasks with the highest performance using as few samples as possible.

Contribution. We focus on the fixed confidence setting and propose the first polynomial-time algorithmic framework for the general CPE-PL problem with novel sample complexity analysis. As an important assumption, we analyze nonlinear reward functions that satisfy Lipschitz continuity. We also assume that there exists a *global observer set* $\sigma = \{x_1, x_2, \dots, x_{|\sigma|}\} \subseteq \mathcal{X}$, such that the stacked $\sum_{i=1}^{|\sigma|} m_{x_i}$ -by- d transformation matrix $M_\sigma = (M_{x_1}; M_{x_2}; \dots; M_{x_{|\sigma|}})$ is of full column rank, i.e., $\text{rank}(M_\sigma) = d$. Our proposed algorithm, namely GCB-PE, samples each super arm in the global observer set to estimate the environment vector θ and constructs a global confidence bound, which is a confidence bound for the global observer set rather than one super arm. Owing to a global confidence bound and Lipschitz continuity of the expected reward function, we can determine whether the empirical best super arm is indeed the best super arm with confidence $1 - \delta$ or not by simply seeing a large enough gap between the empirical best and second best super arms. The experimental results show that our algorithm is superior in terms of speed over the existing ones, and demonstrate that GCB-PE is the first algorithm which can simultaneously deal with combinatorial action space, partial feedback, and nonlinear reward functions.

1.4 Thesis Organization

The organization of this thesis is summarized as follows. In Chapter 2, we briefly review some relevant materials on combinatorial pure exploration and necessary concept from combinatorial optimization. In Chapter 3, we start with top- k arm identifications with full-bandit feedback, the simplest case of combinatorial pure exploration with full-bandit feedback. In Chapter 4, we address a combinatorial pure exploration under general constraints such as paths, matchings, and matroids. In Chapter 5, we study a pure exploration bandit model of the densest subgraph problem, one of the most fundamental graph optimization problems with nonlinear objective functions. In Chapter 6, we study a novel model of combinatorial pure exploration with partial-linear feedback, of which our models in Chapters 3–5 are special cases. The main technical contributions of this thesis lie in Chapters 3–6 where efficient algorithms and theoretical results are provided for individual topics. Chapter 7 concludes with the summary of this thesis and discussion on future work. The roadmap of this thesis can be found in Figure 1.4.

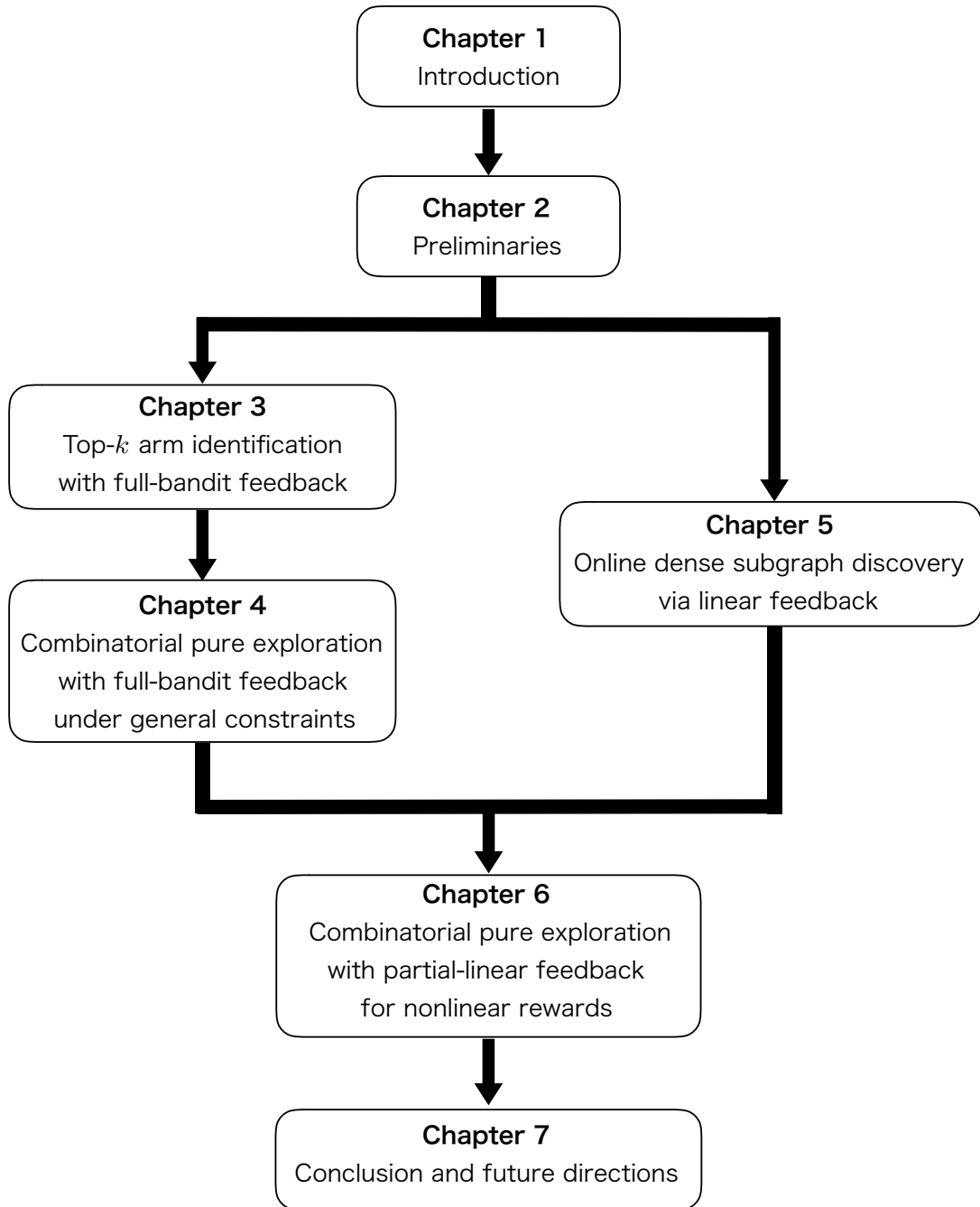


Figure 1.4: The roadmap of this thesis.

Chapter 2

Preliminaries

In this chapter, we provide a common formulation on pure exploration problems and some relevant materials which are necessary to understand the following chapters. Before stating the formal definition of combinatorial pure exploration problems, we will introduce general notation and necessary concepts from combinatorial optimization. Then, we give a description of combinatorial pure exploration problems and see the difference between existing models and those considered in this thesis. We also introduce the pure exploration problem in linear bandits as a related problem, and review existing methods, some of which are the basis of our proposed methods.

2.1 General Notation

Here, we define some notation that is used throughout this thesis.

Basic Notation. By \mathbb{R} (resp. $\mathbb{Q}, \mathbb{Z}, \mathbb{N}$), we denote the set of real (resp. rational, integral, natural) numbers. For an integer d , $[d]$ denotes $\{1, 2, \dots, d\}$. For $S \subseteq [d]$, we use χ_S to denote an indicator vector with the i -th coordinate 1 for $i \in S$ and 0 otherwise. For a matrix $A \in \mathbb{R}^{d \times d}$, $A(i, j)$ denotes the (i, j) -th entry of A for $i, j \in [d]$. For a vector $x \in \mathbb{R}^d$ and a matrix $A \in \mathbb{R}^{d \times d}$, let $\|x\|_A = \sqrt{x^\top A x}$. For a vector $\theta \in \mathbb{R}^d$ and a set $S \subseteq [d]$, we define $\theta(S) = \sum_{e \in S} \theta(e)$. For a given set \mathcal{X} , we use $\Delta(\mathcal{X})$ to denote the family of probability distributions over \mathcal{X} .

Matrices. We call a symmetric matrix $A \in \mathbb{R}^{d \times d}$ positive semidefinite if $x^\top A x \geq 0$ for all $x \in \mathbb{R}^d$ and we call A positive definite if $x^\top A x > 0$ for all $x \neq \mathbf{0}$. For any $x \in \mathbb{R}^d$, let $\text{Diag}(x)$ be the diagonal matrix whose i -th diagonal component is $x(i)$ for $i \in [d]$. We let A^+ denote the Moore-Penrose pseudoinverse of A . We let $\xi_{\max}(A)$ and $\xi_{\min}(A)$ be the maximum and minimum eigenvalues of A , respectively. The condition number of matrix A is defined by $\xi_{\max}(A)/\xi_{\min}(A)$. The identity matrix is denoted by I or, when we want to stress its dimension d , by I_d . For distribution $\lambda \in \Delta(\mathcal{X})$ for a finite set \mathcal{X} , we define $\text{supp}(\lambda) = \{x : \lambda(x) > 0\}$, $M(\lambda) = \mathbb{E}_{z \sim \lambda}[zz^\top]$ and $\widetilde{M}(\lambda) = \sum_{x \in \text{supp}(\lambda)} xx^\top$.

Graphs. A graph $G = (V, E)$ consists of a finite nonempty set V of vertices and finite set E of edges. For a subset of vertices $S \subseteq V$, let $G[S]$ denote the subgraph induced by S , i.e., $G[S] = (S, E(S))$ where $E(S) = \{\{u, v\} \in E : u, v \in S\}$.

Two vertices that are joined by an edge are called adjacent or neighbors. For $S \subseteq V$ and $v \in S$, let $N_S(v) = \{u \in S : \{u, v\} \in E\}$ be the set of neighboring vertices of v in $G[S]$ and let $E_S(v) = \{\{u, v\} \in E : u \in N_S(v)\}$ be the set of incident edges to v in $G[S]$.

Asymptotic notation. For simplicity of exposition, we use $\tilde{O}(f) := O(f) \cdot \text{polylog}(f)$ to hide the log factors.

2.2 Basic Concept from Combinatorial Optimization

In a combinatorial optimization model, given a finite set of ground elements $[d]$, combinatorial constraints $\mathcal{M} \subseteq 2^{[d]}$ and an objective function $f : 2^{[d]} \rightarrow \mathbb{R}$, a decision maker is asked to select a set $S \subseteq [d]$ such that S is feasible, i.e., $S \in \mathcal{M}$, while maximizing/minimizing the objective function $f(S)$. In what follows, we briefly introduce a definition of polynomial-time algorithms and approximation algorithms [Korte and Vygen, 2012, Vazirani, 2013]. Next we introduce the definition of matroids as an important combinatorial structure including size- k subsets and spanning trees.

2.2.1 Running Time

In principle, it is possible to run an algorithm that evaluates f for all feasible subsets to find the optimal solution since the number of subsets of $[d]$ is finite. However, such an enumerating approach is not computationally efficient and prohibitive in practical scenarios, since the size of possible subsets is exponential in the size of ground elements d . Therefore, we focus on *polynomial-time algorithms* defined as follows [Korte and Vygen, 2012].

In the Turing machine model [Turing, 1938], “running time” means the number of elementary steps in which for a valid input the computation of the algorithm reaches a certain output. The input to an algorithm usually consists of a list of numbers. For storing an integer a , we can code it in binary representation using $O(\log(|a| + 2))$ bits. The *input size* of an instance with rational data is the total number of bits needed for the binary representation.

Definition 2.1. *An algorithm with rational input is a polynomial-time algorithm if there is an integer k such that it runs in $O(n^k)$ time, where n is the input size, and all numbers in intermediate computations can be stored with $O(n^k)$ bits.*

2.2.2 Approximation Algorithms for Solving NP-Hard Problems

Next we introduce the definition of *approximation algorithms* [Vazirani, 2013], since we will use approximation algorithms to solve quadratic programming problems appearing in the bandit problems in Chapters 3–5.

A problem is *NP-hard* if it is at least as hard as any problems in class NP, where NP is defined as the set of decision problems that can be solved in polynomial time by a nondeterministic Turing machine. Unless $P = NP$, where P is the class of problems that can be solved in polynomial time, there are no efficient algorithms to find optimal solutions to NP-hard problems. Most researchers today believe that $P \neq NP$. However, important optimization problems that arise

in the real world are mostly NP-hard [Vazirani, 2013]. One of the approaches for solving NP-hard problems is to develop an approximation algorithm. An approximation algorithm is useful for finding near-optimal solutions when the optimal solution is not necessarily required. Throughout this thesis, we define an approximation algorithm as follows.

Definition 2.2. An α -approximation algorithm for a minimization problem is a polynomial-time algorithm that finds a feasible solution whose objective value (OBJ) is within a ratio α of the optimal value (OPT), i.e., $OBJ \leq \alpha \times OPT$.

2.2.3 Matroids

A *matroid* is a combinatorial structure that abstracts many notions of independence such as linearly independent vectors in a set of vectors (called the *linear matroid*) and spanning trees in a graph (called the *graphical matroid*) [Whitney, 1935]. Formally, a matroid is a pair $J = ([d], \mathcal{I})$, where $[d]$ is a finite set called a *ground set* and $\mathcal{I} \subseteq 2^{[d]}$ is a family of subsets of $[d]$ called *independent sets*, that satisfies the following axioms:

1. $\emptyset \in \mathcal{I}$;
2. $X \subseteq Y \in \mathcal{I} \implies X \in \mathcal{I}$;
3. $\forall X, Y \in \mathcal{I}$ such that $|X| < |Y|$, $\exists e \in Y \setminus X$ such that $X \cup \{e\} \in \mathcal{I}$.

A *weighted matroid* is a matroid that has a weight function $w: [d] \rightarrow \mathbb{R}$. Let us consider the following problem: given a weighted matroid $J = (E, \mathcal{I})$ with $w: E \rightarrow \mathbb{R}$, we are asked to find an independent set with the maximum weight, i.e., $\arg\max_{F \in \mathcal{I}} w(F)$. This problem can be solved exactly by the following simple greedy algorithm [Karger, 1998]. The algorithm initially sets F to the empty set. Then, the algorithm sorts the elements in E with the decreasing order by weight, and for each element e in this order, the algorithm adds e to F if $F \cup \{e\} \in \mathcal{I}$. Letting $g(d)$ be the computation time for checking whether F is independent, we see that the running time of the above algorithm is $O(d \log d + ng(d))$.

We will use such a polynomial-time algorithm as a maximization oracle for finding a maximum weight basis of a matroid in our proposed method in Chapter 4.

2.3 Combinatorial Pure Exploration

In this section, we start with the description of the best arm identification (BAI) problem in MAB [Audibert et al., 2010]. Then, we provide the formulation of combinatorial pure exploration problems which is our focus in this thesis.

2.3.1 Best Arm Identification Problem

Let \mathcal{X} be a set of d arms and $\theta \in \mathbb{R}^d$ be the unknown latent vector. In each round t , an agent selects an arm $x_t \in \mathcal{X}$ and observes random outcomes from x_t . Let $\text{Out} \in \mathcal{X}$ be an output of an algorithm and x^* be the optimal arm with the highest expected reward. The goal is to find $\text{Out} \in \mathcal{X}$ while guaranteeing that $\text{Out} = x^*$ with high probability. In the literature, there are two different settings: the *fixed confidence* and *fixed budget* settings defined as follows.

- *Fixed confidence setting*: The agent can determine when to stop the game. After the game is over, she needs to report $\text{Out} \in \mathcal{X}$ satisfying $\Pr[\text{Out} = x^*] \geq 1 - \delta$ for given confidence parameter δ . The agent’s performance is evaluated by her *sample complexity*, i.e., the number of pulls used by her in the game.
- *Fixed budget setting*: The agent needs to minimize the *probability of error*, which is formally $\Pr[\text{Out} \neq x^*]$, within a fixed number of rounds T .

In this thesis, we mainly focus on the fixed confidence setting. We also address the fixed budget setting in Chapter 5, where the quality of output is not necessarily optimal but a $1/2$ -approximate solution.

In the fixed confidence setting, an algorithm that satisfies $\Pr[\text{Out} = x^*] \geq 1 - \delta$ for given $\delta \in (0, 1)$ is called δ -PAC. Any algorithms for BAI in the fixed confidence setting consist of the following three components:

1. A *stopping rule*: which controls when the agent stops the sampling procedure for data acquisition.
2. A *sampling rule*: which determines, based on past observations, which arm x_t is chosen at round t .
3. A *recommendation rule*: which chooses the arm from \mathcal{X} that is to be reported as the optimal arm.

2.3.2 Formulation of Combinatorial Pure Exploration

Existing model of combinatorial pure exploration. To deal with a combinatorial action space, the model of *combinatorial pure exploration of multi-armed bandits* (CPE-MB) was first proposed by Chen et al. [2014]. In this model, there are d base arms, and an action that generates reward is a *super arm* that is a subset of base arms and satisfies the given combinatorial constraint. An agent plays a base arm at each step and observes its random feedback, and the goal is to identify the best super arm with the highest sum of the expected rewards at the end of exploration. They investigated two algorithms called CLUCB and CSAR for the fixed confidence setting and fixed budget setting, respectively, which can work for general combinatorial structures such as matroids, matchings and paths. Recently, a line of work has established near-optimal sampling methods for CPE-MB [Cao and Krishnamurthy, 2019, Chen et al., 2016a, 2017, Gabillon et al., 2016].

Proposed model with limited feedback. Notice that in CPE-MB, the output Out by a recommendation rule is a super arm, but x_t chosen by a sampling rule at any round t is still an element of $[d]$. By contrast to CPE-MB, we consider a model where the agent samples a super arm $x_t \in \mathcal{X}$ at any round t , where $\mathcal{X} \subseteq \{0, 1\}^d$ is a family of super arms rather than a set of base arms $[d]$. In our model, we address two types of feedback: full-bandit and partial-linear feedback. In the full-bandit setting, the agent can only observe the sum of rewards $r_{x_t} = x_t^\top(\theta + \epsilon_t)$ at each pull, where each element $\epsilon_t(e)$ of noise vector ϵ_t is a zero-mean random variable. In the partial-linear setting, the agent only observes

Algorithm 2.1: A general procedure of combinatorial pure exploration problems

Input : Confidence level $\delta \in (0, 1)$, a set of base arms $[d]$

1 while a stopping rule is *False* **do**

2 $t \leftarrow t + 1$;

3 Pull a super arm $x_t \in \mathcal{X}$ by a sampling rule;

4 Observe a random reward depending on x_t as its feedback;

5 Update statistics;

6 return a super arm *Out* by a recommendation rule.

a linear combination of rewards of x_t , i.e., $M_{x_t} \cdot (\theta + \epsilon_t)$ is observed, where M_{x_t} is a transformation matrix $\mathbb{R}^{m_{x_t} \times d}$ whose row dimension m_{x_t} depends on x_t . The summary of a general procedure of our model is given in Algorithm 2.1.

2.4 Related Problems: Best Arm Identification in Linear Bandits

Next we introduce the problem of *best arm identification in linear bandits* (BAI-LB) and review existing algorithms for BAI-LB, by which some of our proposed methods are inspired.

Auer [2003] first introduced the *linear bandit*, an important variant of MAB. In the linear bandit, for a dimension $d > 0$, an agent has the set of arms $\mathcal{X} \subseteq \mathbb{R}^d$. The expected reward for arm $x \in \mathcal{X}$ is written by $x^\top \theta$, where $\theta \in \mathbb{R}^d$ is an unknown parameter. It should be noted that the linear bandit is a generalized model of the ordinary MAB and combinatorial bandits with linear objectives. When $\mathcal{X} = \{e_1, e_2, \dots, e_d\}$ where e_1, e_2, \dots, e_d are the standard basis of the d -dimensional Euclidean space, the linear bandit is reduced to the ordinary MAB. When $\mathcal{X} \subseteq \{0, 1\}^d$ represents a combinatorial action set, the linear bandit coincides with the combinatorial bandits with a linear reward function.

In BAI-LB, at each round t , the agent pulls an arm $x_t \in \mathcal{X}$, and then observes a reward $r_t = x_t^\top \theta + \eta_t$, where η_t is a zero-mean random variable. The goal is to identify the best arm with the highest expected rewards.

In the literature of BAI-LB, Soare et al. [2014] addressed BAI-LB in the fixed confidence setting and first provided a static allocation algorithm for BAI-LB, whose sampling strategy is independent on any past observation. For its design, Soare et al. [2014] introduced the connection between BAI-LB and the *G-optimal experimental design* [Pukelsheim, 2006]. Tao et al. [2018] analyzed the novel randomized estimator based on the convex relaxation of the G-optimal design, and devised a phased-adaptive algorithm whose sample complexity depends linearly on the dimension d . Xu et al. [2018] proposed a fully-adaptive algorithm inspired by UGapE [Gabillon et al., 2012]. Karnin [2016] analyzed the explore-verify algorithms for several settings of BAI including linear, dueling, unimodal, and graphical bandits. Fiez et al. [2019] introduced the transductive BAI-LB, and proposed the first algorithm that nearly achieves the information-theoretic lower bound. Zaki et al. [2019] proposed a generalized LUCB algorithm for BAI-LB with sample complexity analysis for the special cases of two and three arms. Degenne et al. [2020] designed the sampling rule for BAI-LB, and showed that

their sample complexity is asymptotically optimal. Katz-Samuels et al. [2020] proposed near-optimal algorithms for both fixed confidence and fixed budget settings by leveraging the theory of suprema of empirical processes. Zaki et al. [2020] proposed the explicitly described algorithm by using tools from game theory and no-regret learning to solve minimax games.

Despite recent advances in BAI-LB, no existing algorithms for BAI-LB can solve any problems addressed in this thesis efficiently since they implicitly assume that $|\mathcal{X}|$ is small enough to enumerate. However, we will follow some techniques established in the literature of linear bandits in order to deal with full-bandit feedback in our problems. In Chapter 3, we will use the ordinary least squares estimator for unknown vector θ and a high probability bound proposed in Soare et al. [2014]. In Chapter 4, we will use the randomized estimator proposed in Tao et al. [2018] and invoke their algorithm as a subroutine.

We remark that no existing algorithms for BAI-LB can deal with the nonlinear reward functions or partial linear feedback. In Chapter 6, we will use a novel confidence bound and an estimator, which are different from existing algorithms for BAI-LB.

Chapter 3

Top- k Arm Identification with Full-bandit Feedback

In this chapter, we start with stochastic combinatorial pure exploration problems where each super arm is a size- k subset of base arms, called the *top- k arm identification* problems. The developed approach in this chapter will be the basis for solving more complex problems addressed in later chapters.

In *top- k arm identification* problems, an agent sequentially explores a size- k subset of arms from given d arms and tries to identify the top- k arms with the highest expected rewards. In this chapter, we consider the *full-bandit* setting, where only a noisy observation of the total sum of rewards of a super arm is given at each pull. Although our problem can be regarded as an instance of best arm identification in linear bandits, a naive approach based on linear bandits is computationally infeasible since the number of super arms K is exponential. To cope with this problem, we first design a polynomial-time approximation algorithm for a 0-1 quadratic programming problem arising in confidence ellipsoid maximization. Based on our approximation algorithm, we propose a bandit algorithm whose computation time is $O(\log K)$, thereby achieving an exponential speedup over linear bandit algorithms. We provide a sample complexity upper bound that is still worst-case optimal. Finally, we conduct experiments on large-scale datasets with more than 10^{10} super arms, demonstrating the superiority of our algorithms in terms of both the computation time and the sample complexity.

3.1 Introduction

In this section, we will introduce the motivation to study the problem of top- k arm identification and related work. Then, we will summarize our contribution in this chapter.

3.1.1 Background

An important variant of the MAB is the *multiple-play* MAB problem (MP-MAB), in which the agent pulls k (≥ 1) different arms at each round [Anantharam et al., 1987]. In many application domains, we need to make a decision to take multiple actions among a set of all possible choices. For example, in online advertisement auctions, companies want to choose multiple keywords to promote their products to consumers based on their search queries [Rusmevichientong and Williamson, 2006]. From millions of available choices, a company aims to find the most

effective set of keywords by observing the historical performance of the chosen keywords. This decision making is formulated as the MP-MAB, where each arm corresponds to each keyword. In addition, MP-MAB has further applications such as channel selection in cognitive radio networks [Huang et al., 2008], ranking web documents [Radlinski et al., 2008], and crowdsourcing [Zhou et al., 2014]. Owing to various applications, MP-MAB has received much attention and several algorithms have been proposed for regret minimization [Agrawal et al., 1990, Anantharam et al., 1987, Komiyama et al., 2015, Lagr  e et al., 2016]. Adversarial case has been also studied in the literature [see, e.g., Cesa-Bianchi and Lugosi, 2012, Combes et al., 2015, for instance].

In this chapter, we study the *top-k arm identification* that corresponds to the pure exploration in the MP-MAB. In this problem, the goal is to find the size- k subset (a.k.a. a super-arm) with the maximum expected rewards. The problem is also called the *top-k selection* or *k-best arm identification*, and has been extensively studied recently [Bubeck et al., 2013, Cao et al., 2015, Gabillon et al., 2012, 2011b, Kalyanakrishnan and Stone, 2010, Kalyanakrishnan et al., 2012, Kaufmann and Kalyanakrishnan, 2013, Roy Chaudhuri and Kalyanakrishnan, 2017, Chaudhuri and Kalyanakrishnan, 2019, Zhou et al., 2014]. The above prior work has considered the *semi-bandit* setting, in which we can observe a reward of each single-arm in the pulled super-arm, or assumed that a single-arm can be queried. However, in many application domains, it is costly to observe a reward of individual arms, or sometimes we cannot access feedback from individual arms. For example, in crowdsourcing, we often obtain a lot of labels given by crowdworkers, but it is costly to compile labels according to labelers. Furthermore, in software projects, an employer may have complicated tasks that need multiple workers, in which the employer can only evaluate the quality of a completed task rather than a single worker performance [Retelny et al., 2014, Tran-Thanh et al., 2014]. In such scenarios, we wish to extract expert workers who can perform the task with high quality, only from a sequential access to the quality of the task completed by multiple workers.

In this chapter, we tackle the top- k arm identification with *full-bandit* feedback, where only a noisy observation of the total sum of a super-arm is given at each pull rather than a reward of each pulled single-arm. This setting is more challenging since estimators of expected rewards of single-arms are no longer independent of each other. To solve this problem, one might use an algorithm for the non-combinatorial top- k identification problem based on the idea such that the difference in mean reward between two single-arms i and j can be estimated by pulling super-arms $\{i\} \cup A$ and $\{j\} \cup A$ for any size- $(k - 1)$ set $A \subseteq [d]$. However, such an approach fails to reduce the number of samples as shown in Section 5.6, since it cannot fully exploit the information from $k - 1$ arms at each pull.

We can see our problem as an instance of the pure exploration in *linear bandits*, which has received increasing attention [Lattimore and Szepesvari, 2017, Soare et al., 2014, Tao et al., 2018, Xu et al., 2018]. In linear bandits, each arm has its own feature $x \in \mathbb{R}^d$, while in our problem, each super-arm can be associated with a vector $x \in \{0, 1\}^d$. Most linear bandit algorithms have, however, the time complexity at least proportional to the number of arms. Therefore, a naive use of them is computationally infeasible since the number of super-arms $K = \binom{d}{k}$ is exponential. A modicum of research on linear bandits addressed the

time complexity [Jun et al., 2017, Tao et al., 2018]; Jun et al. [2017] proposed efficient algorithms for regret minimization, which results in the sublinear time complexity $O(K^\rho)$ for $\rho \in (0, 1)$. Nevertheless, in our setting, they still have to spend $O(d^{\rho k})$ time, where $\rho \in (0, 1)$ is a constant, which is exponential. Thus, to perform top- k arm identification with full-bandit feedback in practice, the computational infeasibility needs to be overcome since fast decisions are required in real-world applications.

3.1.2 Chapter Contribution

In this chapter, we design algorithms, which are efficient in terms of both the time complexity and the sample complexity. Our contributions are summarized as follows:

(i) We propose a polynomial-time approximation algorithm (Algorithm 3.2) for an NP-hard 0-1 quadratic programming problem arising in confidence ellipsoid maximization. In the design of the approximation algorithm, we utilize algorithms for a classical combinatorial optimization problem called the *densest k -subgraph problem (DkS)* [Feige et al., 2001]. Importantly, we provide a theoretical guarantee for the approximation ratio of our algorithm (Theorem 3.1).

(ii) Based on our approximation algorithm, we propose bandit algorithms (Algorithm 3.3) that runs in $O(\log K)$ time (Theorem 3.2), and provide an upper bound of the sample complexity (Theorem 3.3) that is still worst-case optimal. This result means that our algorithm achieves an exponential speedup over linear bandit algorithms while keeping the statistical efficiency. Moreover, we design two heuristic algorithms, which empirically perform well. We propose an algorithm (Algorithm 3.4) that employs the first-order approximation of confidence ellipsoids, and another algorithm (Algorithm 3.5) that is based on lower-upper confidence bound algorithm.

(iii) We conduct a series of experiments on both synthetic and real-world datasets. First, we run our proposed algorithms on synthetic datasets and verify that our algorithms give good approximation to an *exhaustive* search algorithm. Next, we evaluate our algorithms on large-scale crowdsourcing datasets with more than 10^{10} super arms, demonstrating the superiority of our algorithms in terms of both the time complexity and the sample complexity.

3.1.3 Chapter Organization

The rest of this chapter is organized as follows: In Section 3.2, we formally define the problem addressed in this chapter. Also, we describe the confidence bound for static algorithms. In Section 3.3, we briefly introduce the computational hardness which arises in our problems. In Section 3.4, we introduce a basic of static allocation strategies. In Section 3.5, we discuss a naive algorithm based on simplified confidence bounds that runs in polynomial time and provide the sample complexity. In Section 3.6, we propose approximation algorithm for the 0-1 maximization problem for computing the maximal confidence ellipsoidal norm. In Section 3.7, we design a static algorithm that runs in polynomial time and provide the sample complexity. In Section 3.8, we propose two adaptive algorithms which output the optimal super arm but does not have a sample complexity bound. The results of computational experiments are reported in

Section 5.6. All the proofs in this section is provided in Section 3.10.

3.2 Preliminaries

3.2.1 Problem Definition

Suppose that there are d base arms associated with unknown reward distributions $\{\phi_1, \dots, \phi_d\}$. The reward from ϕ_e for each base arm $e \in [d]$ is expressed as $X_t(e) = \theta(e) + \epsilon_t(e)$, where $\theta(e)$ is the expected reward and $\epsilon_t(e)$ is the zero-mean noise bounded in $[-R, R]$ for some $R > 0$. The agent chooses a size- k subset from n base arms at each round t for an integer $k > 0$. In the well-studied *semi-bandit* setting, the agent pulls a subset M_t , and then she can observe $X_t(e)$ for each $e \in M_t$ independently sampled from the associated unknown distribution ϕ_e . However, in the *full-bandit* setting, she can only observe the sum of rewards $r_{M_t} = \theta(M_t) + \sum_{e \in M_t} \epsilon_t(e)$ at each pull, which means that estimators of expected rewards of base arms are no longer independent of each other.

In this chapter, we call a size- k subset of base arms a *super arm*. We define a *decision class* \mathcal{M} as a finite set of super arms that satisfies the size constraint, i.e., $\mathcal{M} = \{M \in 2^{[d]} : |M| = k\}$; thus, the size of the decision class is given by $K = \binom{d}{k}$. We also use $\mathcal{X} \in \{0, 1\}^d$ to denote a set of super arms where each element x is an indicator vector of a super arm. Let M^* be the optimal super arm in the decision class \mathcal{M} , i.e., $M^* = \arg \max_{M \in \mathcal{M}} \theta(M)$. In this chapter, we focus on the (ε, δ) -PAC setting, where the goal is to design an algorithm to output the super arm $\text{Out} \in \mathcal{M}$ that satisfies for $\delta \in (0, 1)$ and $\varepsilon > 0$, $\Pr[\theta(M^*) - \theta(\text{Out}) \leq \varepsilon] \geq 1 - \delta$. An algorithm is called (ε, δ) -PAC if it satisfies this condition. In the *fixed confidence* setting, the agent's performance is evaluated by her *sample complexity*, i.e., the number of rounds until the agent terminates.

3.2.2 Confidence Bound

In order to handle full-bandit feedback, we utilize approaches for best arm identification in linear bandits. In best arm identification problems, the agent sequentially estimates θ from past observations and confirm that the estimation error is small or not. We introduce the necessary notation as follows. Let $\mathbf{M}_t = (M_1, M_2, \dots, M_t) \in \mathcal{M}^t$ be a sequence of super arms and $(r_{M_1}, \dots, r_{M_t}) \in \mathbb{R}^t$ be the corresponding sequence of observed rewards. Let $\chi_M \in \{0, 1\}^d$ denote the indicator vector of super arm $M \in \mathcal{M}$; for each $e \in [d]$, $\chi_M(e) = 1$ if $e \in M$ and $\chi_M(e) = 0$ otherwise.

Given the sequence of super arm selections $\mathbf{x}_t = (\chi_{M_1}, \dots, \chi_{M_t})$, an unbiased least-squares estimator for $\theta \in \mathbb{R}^d$ can be obtained by

$$\hat{\theta}_t = A_{\mathbf{x}_t}^{-1} b_{\mathbf{x}_t} \in \mathbb{R}^d, \quad (3.1)$$

where

$$A_{\mathbf{x}_t} = \sum_{i=1}^t \chi_{M_i} \chi_{M_i}^\top \in \mathbb{R}^{d \times d} \quad \text{and} \quad b_{\mathbf{x}_t} = \sum_{i=1}^t \chi_{M_i} r_{M_i} \in \mathbb{R}^d. \quad (3.2)$$

It suffices to consider the case where $A_{\mathbf{x}_t}$ is invertible, since we shall exclude a redundant feature when any sampling strategy cannot make $A_{\mathbf{x}_t}$ invertible. For

any $\mathcal{X} \subseteq \mathbb{R}^d$ and \mathbf{x}_t fixed beforehand, Soare et al. [2014] provided the following proposition on the confidence ellipsoid for ordinary least-square estimator $\hat{\theta}_t$. In our problem, the proposition holds for $\sigma = kR$.

Proposition 3.1 (Soare et al. [2014, Proposition 1]). *Let ϵ_t be a noise variable bounded as $\epsilon_t \in [-\sigma, \sigma]$ for $\sigma > 0$. Let $c = 2\sqrt{2}\sigma$ and $c' = 6/\pi^2$ and fix $\delta \in (0, 1)$. Then, for any fixed sequence \mathbf{x}_t , with probability at least $1 - \delta$, the inequality*

$$|x^\top \theta - x^\top \hat{\theta}_t| \leq C_t \|x\|_{A_{\mathbf{x}_t}^{-1}} \quad (3.3)$$

holds for all $t \in \{1, 2, \dots\}$ and $x \in \mathbb{R}^d$, where $C_t = c\sqrt{\log(c't^2K/\delta)}$.

3.3 Computational Hardness

In any (ε, δ) -PAC algorithms, the agent continues sampling a super arm until a certain stopping condition is satisfied. In order to check the stopping condition, existing algorithms for best arm identification in linear bandits involve the following confidence ellipsoid maximization:

$$\text{CEM: } \max. \|\chi_M\|_{A_{\mathbf{x}_t}^{-1}} \text{ s.t. } M \in \mathcal{M}, \quad (3.4)$$

where recall that $\|\chi_M\|_{A_{\mathbf{x}_t}^{-1}} = \sqrt{\chi_M^\top A_{\mathbf{x}_t}^{-1} \chi_M}$. All existing algorithms in linear bandits implicitly assume that an optimal solution to CEM can be exhaustively searched. However, since the number of super arms K is exponential in our setting, it is computationally intractable to exactly solve it.

Let $W \in \mathbb{R}^{d \times d}$ be a symmetric matrix. CEM introduced in (3.4) can be naturally represented by the following 0-1 quadratic programming problem:

$$\begin{aligned} \text{QP: } \max. & \sum_{i=1}^d \sum_{j=1}^d w_{ij} x_i x_j \\ \text{s.t. } & \sum_{i=1}^d x_i = k, x_i \in \{0, 1\}, \forall i \in [d]. \end{aligned} \quad (3.5)$$

Notice that QP can be seen as an instance of the *uniform quadratic knapsack problem*, which is known to be NP-hard [Taylor, 2016], and there are few results of polynomial-time approximation algorithms even for a special case. The uniform quadratic knapsack problem (UQKP) is defined as follows. Assume that we have n items, each of which has weight 1. In addition, we are given an $n \times n$ non-negative integer matrix $W = (w_{ij})$, where w_{ii} is the profit achieved if item i is selected and $w_{ij} + w_{ji}$ is a profit achieved if both items i and j are selected for $i < j$. The UQKP calls for selecting a subset of items whose overall weight does not exceed a given knapsack capacity k , so as to maximize the overall profit. The UQKP can be formulated as the following 0-1 integer quadratic programming:

$$\begin{aligned} \max. & \sum_{i=1}^d \sum_{j=1}^d w_{ij} x_i x_j \\ \text{s.t. } & \sum_{i=1}^d x_i \leq k. \end{aligned}$$

The UQKP is an NP-hard problem. Indeed, the maximum clique problem, which is also NP-hard, can be reduced to it; Given a graph $G = (V, E)$, we set $w_{ii} = 0$ for all i and $w_{ij} = 1$ for all $\{i, j\} \in E$. Solving this problem, it allows us to find a clique of size k if and only if the optimal solution of the problem has value $k(k-1)$ [Taylor, 2016].

Therefore, we need approximation algorithms for CEM or a totally different approach for solving the top- k arm identification with full-bandit feedback, since we may involve a computational hard optimization if we naively use similar algorithms in linear bandits.

3.4 Static Allocation Strategies

First, we deal with *static* allocation strategies, which sequentially sample a super arm from a fixed sequence of super arms. In general, adaptive strategies will perform better than static ones, but due to the computational hardness, we focus on static ones to analyze the worst-case optimality in terms of the *minimum gap* $\Delta_{\min} = \operatorname{argmin}_{M \in \mathcal{M} \setminus \{M^*\}} \theta(M^*) - \theta(M)$. Suppose that we are given a static allocation $\lambda \in \Delta(\mathcal{M})$, which is a distribution with the polynomial-sized support. In static algorithms, the agent pulls a super arm from a fixed set of super arms based on λ until a certain stopping condition is satisfied. Therefore, it is important to construct a stopping condition guaranteeing that the estimate $\hat{\theta}_t$ belongs to a set of parameters that admits the empirical best super arm $\hat{M}_t^* = \operatorname{argmax}_{M \in \mathcal{M}} \hat{\theta}_t(M)$ as an optimal super arm M^* as quickly as possible.

To make confidence bounds shrink fast, the static allocation strategy named as *G-allocation* is proposed in Soare et al. [2014] for linear bandits. In the *G-allocation* strategy, we make the sequence of selection \mathbf{x}_t to be

$$\mathbf{x}_t^G = \operatorname{argmin}_{\mathbf{x}_t \in \mathbb{R}^{d \times t}} \max_{x \in \mathcal{X}} \|x\|_{A_{\mathbf{x}_t}^{-1}},$$

which is NP-hard optimization problem. There are massive studies that proposed approximate solutions to solve it in the experimental design literature [Bouhtou et al., 2010, Sagnol, 2013, Pukelsheim, 2006]. We let $\lambda^* \in \Delta(\mathcal{X})$ be the optimal distribution obtained by its convex relaxation problem, i.e.,

$$\lambda^* = \operatorname{argmin}_{\lambda \in \Delta(\mathcal{X})} \max_{x \in \mathcal{X}} \|x\|_{\Lambda(\lambda)^{-1}},$$

where $\Lambda(\lambda) = \sum_{x \in \mathcal{X}} \lambda_x x x^\top$ and λ_x is the probability that arm x is chosen. Approximating the optimal G-allocation can be done via an efficient rounding procedure using the distribution λ^* [Pukelsheim, 2006].

Note that in linear bandits or experimental design literature, the size of action set $K = |\mathcal{X}|$ is assumed to be small enough to enumerate. To adapt G-allocation to combinatorial settings, we need to restrict the support of λ to the polynomial size in d . Let $\mathcal{M}' \subseteq \mathcal{M}$ be any subset of super arms where the size of \mathcal{M}' is polynomial in d . Let $\Delta(\mathcal{M}')$ be a probability simplex on \mathcal{M}' . In the rest of this chapter, we define a given allocation strategy λ as $\lambda = (\lambda_M)_{M \in \mathcal{M}'} \in \Delta(\mathcal{M}')$, where λ_M prescribes the proportions of pulls to super arm M , and let $\operatorname{supp}(\lambda) = \{M \in \mathcal{M}' : \lambda_M > 0\}$ be its support.

3.5 Naive Method by Simplified Confidence Bounds

In this section, we see the fundamental observation of employing a simplified confidence bound as a naive method to obtain a computationally efficient algorithm. We provide a sample complexity and see the trade-off between the statistical efficiency and computational efficiency. The (ϵ, δ) -PAC algorithm proposed in this section, named ICB, is also evaluated as a simple benchmark strategy in our experiments.

For a matrix $B \in \mathbb{R}^{d \times d}$, let $B(i, j)$ denote the (i, j) -th entry of B . We construct a simplified confidence bound, named an *independent confidence bound*, which is obtained by diagonal approximation of confidence ellipsoids. We start with the following lemma, which shows that θ lies in an independent confidence region centered at $\hat{\theta}_t$ with high-probability.

Lemma 3.1. *Let $c' = 6/\pi^2$. Let ϵ_t be a noise variable bounded as $\epsilon_t \in [-\sigma, \sigma]$ for $\sigma > 0$. Then, for any fixed sequence \mathbf{x}_t , any $t \in \{1, 2, \dots\}$, and $\delta \in (0, 1)$, with probability at least $1 - \delta$, the inequality*

$$|x^\top \theta - x^\top \hat{\theta}_t| \leq C_t \sum_{i=1}^d |x_i| \sqrt{A_{\mathbf{x}_t}^{-1}(i, i)} \quad (3.6)$$

holds for all $x \in \{-1, 0, 1\}^d$, where

$$C_t = \sigma \sqrt{2 \log(c' t^2 d / \delta)}.$$

This lemma can be derived from Proposition 3.1 and the triangle inequality. The RHS of (3.6) only has linear terms of $\{x_i\}_{i \in [d]}$, whereas that of (3) in Proposition 3.1 has the matrix norm $\|x\|_{A_{\mathbf{x}_t}^{-1}}$, which results in a difficult instance. As long as we assume that linear maximization oracle is available, maximization of this value can be also done in polynomial time. For example, maximization of the RHS of (3.6) under matroid constraints can be solved by using the simple greedy procedure [Karger, 1998]. Based on the independent confidence bounds, we propose ICB, which is detailed in Algorithm 3.1. At each round t , ICB computes the empirical best super arm \widehat{M}_t^* , and then solves the following maximization problem:

$$\begin{aligned} P_1 : \quad & \max. \quad \widehat{\theta}_t(M) + C_t \sum_{i=1}^d |\chi_M(i) - \chi_{\widehat{M}_t^*}(i)| \sqrt{A_{\mathbf{x}_t}^{-1}(i, i)}, \\ \text{s.t.} \quad & M \in \mathcal{M} \setminus \{\widehat{M}_t^*\}. \end{aligned} \quad (3.7)$$

The second term in the objective of P_1 can be regarded as the confidence interval of the estimated gap $\widehat{\theta}_t(M) - \widehat{\theta}_t(\widehat{M}_t^*)$. ICB continues sampling a super arm until the following stopping condition is satisfied:

$$Z_t^* - \widehat{\theta}_t(\widehat{M}_t^*) < \epsilon, \quad (3.8)$$

where Z_t^* represents the optimal value of P_1 . Note that P_1 is solvable in polynomial time because P_1 is an instance of linear maximization problems. As the following lemma states, ICB is an efficient algorithm in terms of the computation time.

Algorithm 3.1: Static allocation with independent confidence bound (ICB)

Input : Accuracy $\epsilon > 0$, confidence level $\delta \in (0, 1)$, allocation strategy λ

```

1 for  $t = 1, \dots, d$  do
2    $t \leftarrow t + 1$ ;
3   Pull  $M_t \in \text{supp}(\lambda)$ ;
4   Observe  $r_t$ ;
5   Update  $A_t$  and  $b_t$ ;
6 while stopping condition (3.8) is not true do
7    $t \leftarrow t + 1$ ;
8   Pull  $M_t \leftarrow \text{argmin}_{M \in \text{supp}(\lambda)} \frac{T_M(t)}{\lambda_M}$ ;
9   Observe  $r_t$ ;
10  Update  $A_t$  and  $b_t$ ;
11   $\hat{\theta}_t \leftarrow A_{\mathbf{x}_t}^{-1} b_t$ ;
12   $\hat{M}_t^* \leftarrow \text{argmax}_{M \in \mathcal{M}} \hat{\theta}_t(M)$ ;
13   $Z_t^* \leftarrow \max_{M \in \mathcal{M} \setminus \{\hat{M}_t^*\}} \left( \hat{\theta}_t(M) + C_t \sum_{i=1}^d |(\chi_M(i) - \chi_{\hat{M}_t^*}(i))| \sqrt{A_{\mathbf{x}_t}^{-1}(i, i)} \right)$ ;
14 return  $\hat{M}^* \leftarrow \hat{M}_t^*$ 

```

Lemma 3.2. *Given any instance of top- k arm identification with full-bandit feedback, ICB (Algorithm 3.1) at each round $t \in \{1, 2, \dots\}$ runs in polynomial time.*

From the definition, we have $A_t = \sum_{M \in \mathcal{M}} T_M(t) \chi_M \chi_M^\top$, where $T_M(t)$ denotes the number of times that M is pulled before the round $t + 1$. Let $\Lambda'_\lambda = \sum_{M \in \mathcal{M}} \lambda_M \chi_M \chi_M^\top$. We define $\rho'(\lambda)$ as

$$\rho'(\lambda) = \left(\max_{M, M' \in \mathcal{M}} \sum_{i=1}^d |\chi_M(i) - \chi_{M'}(i)| \sqrt{\Lambda_\lambda^{-1}(i, i)} \right)^2. \quad (3.9)$$

Now, we give a problem-dependent sample complexity bound of ICB with allocation strategy λ as follows.

Lemma 3.3. *Given any instance of top- k arm identification with full-bandit feedback, with probability at least $1 - \delta$, ICB (Algorithm 3.1) returns an ϵ -optimal super arm \hat{M}^* and the total number of samples T is bounded as follows:*

$$T = O \left(k^2 R^2 H'_\epsilon \log \left(\frac{d}{\delta} \left(k R H'_\epsilon \left(k^2 R^2 H'_\epsilon + \log \left(\frac{d}{\delta} \right) \right) \right) \right) \right),$$

where

$$H'_\epsilon = \frac{\rho'(\lambda)}{(\Delta_{\min} + \epsilon)^2}.$$

Notice that in the MAB, this diagonal approximation is tight since $A_{\mathbf{x}_t}$ becomes a diagonal matrix. However, for combinatorial settings where the size of super arms is $k \geq 2$, there is no guarantee that this approximation is tight;

Algorithm 3.2: Quadratic Maximization

Input : Symmetric matrix $W \in \mathbb{R}^{d \times d}$
1 $V \leftarrow [d]$;
2 $E \leftarrow \{\{i, j\} : i, j \in [d], i \neq j\}$;
3 **for** $\{i, j\} \in E$ **do** $\tilde{w}_{ij} \leftarrow w_{ij} + w_{ii} + w_{jj}$;
4 Construct $\tilde{G} = (V, E, \tilde{w})$;
5 $S \leftarrow \text{DkS-Oracle}(\tilde{G})$;
6 **return** S

the approximation may degrade the sample complexity. Although the proposed algorithm here empirically performed well when the number of base arms is not large in our experiments, it is still unclear that using the simplified confidence bound should be desired instead of ellipsoids confidence bounds. This is the reason why we focus on the approach with confidence ellipsoids, which will be discussed in the following section.

3.6 Confidence Ellipsoid Maximization

In this section, we design an approximation algorithm for confidence ellipsoid maximization CEM.

By utilizing algorithms for a classical combinatorial optimization problem, called the *densest k -subgraph problem (DkS)*, we design an approximation algorithm that admits theoretical performance guarantee for QP in (3.5) with positive definite matrix W . The definition of the DkS is as follows. Let $G = (V, E, w)$ be an undirected graph with nonnegative edge weight $w = (w_e)_{e \in E}$. For a vertex set $S \subseteq V$, let $E(S) = \{\{u, v\} \in E : u, v \in S\}$ be the subset of edges in the subgraph induced by S . We denote by $w(S)$ the sum of the edge weights in the subgraph induced by S , i.e., $w(S) = \sum_{e \in E(S)} w_e$. In the DkS, given $G = (V, E, w)$ and positive integer k , we are asked to find $S \subseteq V$ with $|S| = k$ that maximizes $w(S)$. Although the DkS is NP-hard, there are a variety of polynomial-time approximation algorithms [Asahiro et al., 2000, Bhaskara et al., 2010, Feige et al., 2001]. The current best approximation result for the DkS has an approximation ratio of $\Omega(1/|V|^{1/4+\epsilon})$ for any $\epsilon > 0$ [Bhaskara et al., 2010]. The direct reduction of QP to the DkS results in an instance that has arbitrary weights of edges. Existing algorithms cannot be used for such an instance since these algorithms need an assumption that the weights of all edges are nonnegative.

Now we present our algorithm for QP, which is detailed in Algorithm 3.2. The algorithm operates in two steps. In the first step, it constructs an d -vertex complete graph $\tilde{G} = (V, E, \tilde{w})$ from a given symmetric matrix $W \in \mathbb{R}^{d \times d}$. For each $\{i, j\} \in E$, the edge weight \tilde{w}_{ij} is set to $w_{ij} + w_{ii} + w_{jj}$. Note that if W is positive definite, $\tilde{w}_{ij} \geq 0$ holds for every $\{i, j\} \in E$, which means that \tilde{G} is an instance of the DkS. In the second step, the algorithm accesses the *densest k -subgraph oracle (DkS-Oracle)*, which accepts \tilde{G} as input and returns in polynomial time an approximate solution for the DkS. Note that we can use any polynomial-time approximation algorithm for the DkS as the DkS-Oracle. Let α_{DkS} be the approximation ratio of the algorithm employed by the DkS-Oracle. By sophisticated analysis on the approximation ratio of Algorithm 3.2,

Algorithm 3.3: Static allocation algorithm with quadratic maximization (SAQM)

Input : Accuracy $\epsilon > 0$, confidence level $\delta \in (0, 1)$, allocation strategy λ

- 1 **for** $t = 1, \dots, n$ **do**
- 2 $t \leftarrow t + 1$ and pull $M_t \in \text{supp}(\lambda)$;
- 3 Observe r_{M_t} , and update $A_{\mathbf{x}_t}$ and b_t ;
- 4 **while** *stopping condition (3.10) is not true* **do**
- 5 $t \leftarrow t + 1$;
- 6 Pull $M_t \leftarrow \text{argmin}_{M \in \text{supp}(\lambda)} \frac{T_M(t)}{p_\lambda}$;
- 7 Observe r_{M_t} , and update $A_{\mathbf{x}_t}$ and b_t ;
- 8 $\hat{\theta}_t \leftarrow A_{\mathbf{x}_t}^{-1} b_t$;
- 9 $\hat{M}_t^* \leftarrow \text{argmax}_{M \in \mathcal{M}} \hat{\theta}_t(M)$;
- 10 $M'_t \leftarrow \text{Quadratic Maximization}(A_{\mathbf{x}_t}^{-1})$;
- 11 $Z_t \leftarrow C_t \|\chi_{M'_t}\|_{A_{\mathbf{x}_t}^{-1}}$;
- 12 **return** $\hat{M}^* \leftarrow \hat{M}_t^*$

we have the following theorem.

Theorem 3.1. *For QP with any positive definite matrix $W \in \mathbb{R}^{d \times d}$, Algorithm 3.2 employing α_{DkS} -approximation DkS -Oracle is a $\left(\frac{1}{k-1} \frac{\xi_{\min}(W)}{\xi_{\max}(W)} \alpha_{DkS}\right)$ -approximation algorithm, where $\xi_{\min}(W)$ and $\xi_{\max}(W)$ represent the minimum and maximum eigenvalues of W , respectively.*

3.7 Static Allocation Algorithm with Quadratic Maximization

Based on the approximation algorithm proposed in the previous section, we propose two algorithms for the top- k arm identification with full-bandit feedback. Note that we assume that $k \geq 2$ since the top- k arm identification with $k = 1$ is the same as best arm identification problem of the MAB.

Now we propose an algorithm named **SAQM**, which is detailed in Algorithm 3.3. Let $T_M(t)$ be the number of times that M is pulled before $(t + 1)$ -th round. At each round t , **SAQM** samples a super arm $M_t = \text{argmin}_{M \in \text{supp}(\lambda)} T_M(t)/\lambda_M$, and updates statistics $A_{\mathbf{x}_t}$, b_t and $\hat{\theta}_t$. Then, the algorithm computes the empirical best super arm \hat{M}_t^* , and approximately solves CEM in (3.4), using Algorithm 3.2 as a subroutine. Note that any α -approximation algorithm for QP is a $\sqrt{\alpha}$ -approximation algorithm for CEM. **SAQM** employs the following stopping condition:

$$\hat{\theta}_t(\hat{M}_t^*) - C_t \|\chi_{\hat{M}_t^*}\|_{A_{\mathbf{x}_t}^{-1}} \geq \max_{M \in \mathcal{M} \setminus \{\hat{M}_t^*\}} \hat{\theta}_t(M) + \frac{1}{\alpha_t} C_t Z_t - \epsilon, \quad (3.10)$$

where Z_t denotes the objective value of an approximate solution M'_t for CEM, and α_t denotes the approximation ratio of our algorithm for CEM at round t . Note that we can specify the value of α_t using the guarantee in Theorem 3.1, and

this stopping condition allows the output to be ε -optimal with high probability. As the following theorem states, **SAQM** provides an exponential speedup over exhaustive search algorithms.

Theorem 3.2. *Let $\text{poly}(d)_{\text{DkS}}$ be the computation time of the DkS-Oracle. At any round $t > 0$, **SAQM** (Algorithm 3.3) runs in $O(\max\{d^2, \text{poly}(d)_{\text{DkS}}\})$ time.*

Most existing approximation algorithms for the DkS have efficient computation time. For example, if we employ the algorithm by Feige et al. [2001] as the DkS-Oracle that runs in $O(d^\omega)$ time in Algorithm 3.3, the running time of **SAQM** becomes $O(d^\omega)$, where the exponent $\omega \leq 2.373$ is equal to that of the computation time of matrix multiplication (e.g., see Le Gall [2014]). If we employ the algorithm by Asahiro et al. [2000] that runs in $O(d^2)$, the running time of **SAQM** also becomes $O(d^2)$.

Recall that $\Lambda_\lambda = \sum_{M \in \mathcal{M}} \lambda_M \mathbf{x}_M \mathbf{x}_M^\top$ is a design matrix. We define the problem complexity H_ε as

$$H_\varepsilon = \frac{\rho(\lambda)}{(\Delta_{\min} + \varepsilon)^2},$$

where $\rho(\lambda) = \max_{M \in \mathcal{M}} \|\mathbf{x}_M\|_{\Lambda_\lambda^{-1}}^2$, which is also appeared in Soare et al. [2014]. The next theorem shows that **SAQM** is (ε, δ) -PAC and gives a problem-dependent sample complexity bound.

Theorem 3.3. *Given any instance of the top- k arm identification with full-bandit feedback, with probability at least $1 - \delta$, **SAQM** (Algorithm 3.3) with α -approximation of CEM returns an ε -optimal set \hat{M}^* , and the total number of samples T is bounded as follows:*

$$T \leq 8 \left(3 + \frac{1}{\alpha}\right)^2 \sigma^2 H_\varepsilon \log \left(\frac{K}{\delta}\right) + C(H_\varepsilon, \delta),$$

where

$$C(H_\varepsilon, \delta) = O \left(\sigma^2 H_\varepsilon \log \left(\frac{\sigma^2}{\alpha^2} H_\varepsilon + \log \left(\frac{K}{\delta} \right) \right) \right).$$

It is worth mentioning that if we have an α -approximation algorithm for CEM with a more general decision class \mathcal{M} (such as paths, matchings, matroids), we have the same sample complexity bound in Theorem 3.3 for the combinatorial pure exploration (CPE) with general constraints. For the top- k arm identification setting, we have $\alpha = \Omega \left(k^{-\frac{1}{2}} d^{-\frac{1}{8}} \sqrt{\frac{\xi_{\min}(\Lambda_\lambda)}{\xi_{\max}(\Lambda_\lambda)}} \right)$ in Theorem 3.3.

Soare et al. [2014] considered the *oracle sample complexity* of a linear best-arm identification problem. The oracle complexity, which is based on the optimal allocation strategy λ derived from the true parameter θ , is $O(H_\varepsilon \log(1/\delta))$ if we ignore the terms that are not related to H_ε and δ . Soare et al. [2014] showed that the sample complexity with G-allocation strategy matches the oracle sample complexity up to constants in the worst case. The sample complexity of **SAQM** is also worst-case optimal in the sense that it matches $O(H_\varepsilon \log(1/\delta))$, while **SAQM** runs in polynomial time.

Note that if we use Proposition 3.2 that will be specified in the next section instead of Proposition 3.1, we have a better sample complexity bound in terms

Algorithm 3.4: Static allocation algorithm with first-order approximation (SA-FOA)

Input : Accuracy $\epsilon > 0$, confidence level $\delta \in (0, 1)$, allocation strategy λ

```

1 for  $t = 1, \dots, d$  do
2    $t \leftarrow t + 1$  and pull  $M_t \in \text{supp}(\lambda)$ ;
3   Observe  $r_{M_t}$ , and update  $A_{\mathbf{x}_t}$  and  $b_t$ ;
4 while  $\frac{\epsilon}{2} < Z'_t - \hat{\theta}_t(\hat{M}_t^*)$  do
5    $t \leftarrow t + 1$ ;
6   Pull  $M_t \leftarrow \text{argmin}_{M \in \text{supp}(\lambda)} \frac{T_M(t)}{\lambda_M}$ ;
7   Observe  $r_{M_t}$ , and then update  $A_{\mathbf{x}_t}$ ,  $b_t$  and  $\hat{\theta}_t \leftarrow A_{\mathbf{x}_t}^{-1} b_t$ ;
8    $\hat{M}_t^* \leftarrow \text{argmax}_{M \in \mathcal{M}} \hat{\theta}_t(M)$ ;
9    $m \leftarrow \ell d$  for some positive integer  $\ell$ ;
10  for  $i = 1, \dots, m$  do
11     $F \leftarrow \{\emptyset\}$ ;
12    Choose a super arm  $M_i \in \text{supp}(\lambda)$  uniformly at random;
13     $\gamma \leftarrow \frac{C_t}{2\|\mathbf{x}_{M_i} - \mathbf{x}_{\hat{M}_t^*}\|_{A_{\mathbf{x}_t}^{-1}}}$ ;
14     $B_t \leftarrow \gamma A_{\mathbf{x}_t}^{-1} - \text{Diag}(2\gamma(A_{\mathbf{x}_t}^{-1} \mathbf{x}_{\hat{M}_t^*})) + \text{Diag}(\hat{\theta}_t)$ ;
15     $F \leftarrow F \cup \{\text{QuadraticMaximization}(B_t)\}$ ;
16   $Z'_t \leftarrow \max_{M \in F} (\hat{\theta}_t(M) + C_t \|\mathbf{x}_M - \mathbf{x}_{\hat{M}_t^*}\|_{A_{\mathbf{x}_t}^{-1}})$ ;
17 return  $\hat{M}^* \leftarrow \hat{M}_t^*$ 

```

of k (or d). However, the sample complexity with Proposition 3.2 becomes complicated since it will depend on regularization parameter λ . Therefore, we analyze the sample complexity bound based on Proposition 3.1 to clarify how the sample complexity depends on α , H_ϵ and δ , rather than k (or d).

Although the quantity $\rho(\lambda)$ is unbounded in the worst case, it is upper bounded by d as pointed out in Soare et al. [2014], if we use G-allocation strategy as sampling strategy λ . However, the exact G-allocation might be hard to compute if d is large, since the number of variable can be exponential.

Remark 3.1. Suppose that SAQM (Algorithm 3.4) employs G-allocation strategy λ^G , i.e., $\lambda^G = \text{argmin}_{p \in \mathcal{P}} \max_{M \subseteq [d]} \mathbf{x}_M \Lambda_p^{-1} \mathbf{x}_M$. Then, from the theorem in [Kiefer and Wolfowitz [1960, Section 2]], we have $\rho(\lambda^G) = d$.

3.8 Heuristic Algorithms

In this section, we propose two algorithms which output the optimal super arm but does not have a sample complexity bound. The first algorithm employs the first-order approximation of confidence ellipsoids for checking stopping condition efficiently. The second one is based on an adaptive sampling strategy. Empirically, we evaluated these algorithms and observed that they perform well in our experiments.

First order approximation for confidence ellipsoid maximization. In SAQM, we compute an upper confidence bound of the expected reward of each super arm. However, in order to reduce the number of required samples, we wish to directly construct a tight confidence bound for the gap of the reward between two super arms. For this reason, we propose another algorithm SA-FOA. The procedure of SA-FOA is shown in Algorithm 3.4. Given an allocation strategy λ , this algorithm continues sampling until the stopping condition $\frac{\varepsilon}{2} \geq Z'_t - \hat{\theta}_t(\widehat{M}_t^*)$ is satisfied, where Z'_t denotes the objective value of an approximate solution of the following maximization problem:

$$\max_{M \in \mathcal{M} \setminus \{\widehat{M}_t^*\}} \left(\hat{\theta}_t(M) + C_t \|\chi_M - \chi_{\widehat{M}_t^*}\|_{A_{\mathbf{x}_t}^{-1}} \right). \quad (3.11)$$

The second term of (3.11) can be regarded as the confidence interval of the estimated gap $\hat{\theta}_t(M) - \hat{\theta}_t(\widehat{M}_t^*)$. In order to simultaneously maximize the estimated reward $\hat{\theta}_t(M)$ and the matrix norm $\|\chi_M - \chi_{\widehat{M}_t^*}\|_{A_{\mathbf{x}_t}^{-1}}$, we employ a first-order approximation technique. For a fixed super arm M_i , we approximate $\|\chi_M - \chi_{\widehat{M}_t^*}\|_{A_{\mathbf{x}_t}^{-1}}$ using the following bound:

$$\|\chi_M - \chi_{\widehat{M}_t^*}\|_{A_{\mathbf{x}_t}^{-1}} \leq \frac{\|\chi_M - \chi_{\widehat{M}_t^*}\|_{A_{\mathbf{x}_t}^{-1}}^2}{2\|\chi_{M_i} - \chi_{\widehat{M}_t^*}\|_{A_{\mathbf{x}_t}^{-1}}} + \frac{\|\chi_{M_i} - \chi_{\widehat{M}_t^*}\|_{A_{\mathbf{x}_t}^{-1}}}{2}, \quad (3.12)$$

which follows from $\sqrt{a+x} \leq \sqrt{a} + \frac{x}{2\sqrt{a}}$ for any $a, x > 0$. Using (3.12), the objective function in (3.11) can be bounded as follows:

$$\begin{aligned} & \hat{\theta}_t(M) + C_t \|\chi_M - \chi_{\widehat{M}_t^*}\|_{A_{\mathbf{x}_t}^{-1}} \\ & \leq \hat{\theta}_t^\top \chi_M + C_t \left(\frac{\|\chi_M - \chi_{\widehat{M}_t^*}\|_{A_{\mathbf{x}_t}^{-1}}^2}{2\|\chi_{M_i} - \chi_{\widehat{M}_t^*}\|_{A_{\mathbf{x}_t}^{-1}}} + \frac{\|\chi_{M_i} - \chi_{\widehat{M}_t^*}\|_{A_{\mathbf{x}_t}^{-1}}}{2} \right) \end{aligned} \quad (3.13)$$

$$= \hat{\theta}_t^\top \chi_M + C_t \left(\frac{\|\chi_M\|_{A_{\mathbf{x}_t}^{-1}}^2 - 2(\chi_{\widehat{M}_t^*}^\top A_{\mathbf{x}_t}^{-1})^\top \chi_M + \|\chi_{\widehat{M}_t^*}\|_{A_{\mathbf{x}_t}^{-1}}^2}{2\|\chi_{M_i} - \chi_{\widehat{M}_t^*}\|_{A_{\mathbf{x}_t}^{-1}}} \right) \quad (3.14)$$

$$+ \frac{C_t \|\chi_{M_i} - \chi_{\widehat{M}_t^*}\|_{A_{\mathbf{x}_t}^{-1}}}{2}. \quad (3.15)$$

For any $y \in \mathbb{R}^n$, let $\text{Diag}(y)$ be a diagonal matrix whose i -th diagonal component is $y(i)$ for $i \in [d]$. Note that for a fixed round t and M_i , the term $\|\chi_{M_i} - \chi_{\widehat{M}_t^*}\|_{A_{\mathbf{x}_t}^{-1}}$ and $\|\chi_{\widehat{M}_t^*}\|_{A_{\mathbf{x}_t}^{-1}}^2$ do not depend on χ_M . Therefore, the above first-order approximation allows us to transform the original problem to QP, where the objective function is

$$\chi_M^\top \left(\gamma A_{\mathbf{x}_t}^{-1} - \text{Diag}(2\gamma(A_{\mathbf{x}_t}^{-1} \chi_{\widehat{M}_t^*})) + \text{Diag}(\hat{\theta}_t) \right) \chi_M,$$

with a positive constant $\gamma = \frac{C_t}{2\|\chi_{M_i} - \chi_{\widehat{M}_t^*}\|_{A_{\mathbf{x}_t}^{-1}}}$. We can approximately solve it by Algorithm 3.2, and choose the best approximate solution that maximizes the original objective among ℓn super arms. Notice that SA-FOA is an (ϵ, δ) -PAC algorithm since we compute the upper bound of the objective function in (3.11) and thus it will not stop earlier. In our experiments, it works well although we have no theoretical results on the sample complexity. We will also observe in the experiments that the approximation error of SA-FOA for (3.11) becomes smaller as the number of rounds increases.

Algorithm 3.5: Combinatorial lower-upper confidence bound with quadratic maximization (CLUCEB-QM)

Input : Accuracy $\epsilon > 0$, confidence level $\delta \in (0, 1)$

1 **Initialization** For each $e \in [d]$, pull $M_e \in \mathcal{M}$ such that $e \in M_e$ once.
Initialize $A_{\mathbf{x}_t}^\omega$ and b_t and set $T_e(t) \leftarrow 1$ for all $e \in [d]$. Fix $e' \in [d]$.

2 **while** *stopping condition* (3.10) **is not true** **do**

3 $t \leftarrow t + 2$;

4 $\widehat{M}_t^* \leftarrow \operatorname{argmax}_{M \in \mathcal{M}} \widehat{\theta}_t(M)$;

5 Compute confidence radius $\operatorname{rad}_t(e) = R \sqrt{\frac{2k \log\left(\frac{4dt^3}{\delta}\right)}{T_e(t)}}$ for all $e \in [d]$;

6 **for** $e = 1, \dots, n$ **do**

7 **if** $e \in \widehat{M}_t^*$ **then** $\widetilde{\theta}_t(e) = \widehat{\theta}_t(e) - \operatorname{rad}_t(e)$;

8 **else** $\widetilde{\theta}_t(e) \leftarrow \widehat{\theta}_t(e) + \operatorname{rad}_t(e)$;

9 $\widetilde{M}_t \leftarrow \operatorname{argmax}_{M \in \mathcal{M}} \widetilde{\theta}_t(M)$;

10 $e_t \leftarrow \operatorname{argmax}_{e \in (\widetilde{M}_t \setminus \widehat{M}_t^*) \cup (\widehat{M}_t^* \setminus \widetilde{M}_t)} \operatorname{rad}_t(e)$;

11 $T_t(e_t) \leftarrow T_{t-2}(e_t) + 1$;

12 Choose $M_t \in \mathcal{M}$ such that $e_t \in M_t$ and $e' \notin M_t$ uniformly at random;

13 Pull $M_t \in \mathcal{M}$ and $M'_t = M_t \setminus \{e_t\} \cup \{e'\}$;

14 Observe r_{M_t} and $r_{M'_t}$, and then update $A_{\mathbf{x}_t}^\omega$, b_t and $\widehat{\theta}_t$;

15 $M'_t \leftarrow \text{Quadratic Maximization}((A_{\mathbf{x}_t}^\omega)^{-1})$;

16 $Z_t \leftarrow C_t \|\chi_{M'_t}\|_{(A_{\mathbf{x}_t}^\omega)^{-1}}$;

17 **return** $\widehat{M}^* \leftarrow \widehat{M}_t^*$

Adaptive algorithm. In the previous section, using the approximation algorithm for confidence ellipsoid maximization, we proposed algorithms based on static allocation strategies. However, in order to design a near-optimal sampling algorithm, we should focus on adaptive algorithms, which adaptively changes arm selection strategies based on the past observations at every round. In this section, we propose an adaptive algorithm making use of existing classical methods.

If \mathbf{x}_t is adaptively determined based on past observations, we consider the regularized least-square estimator is given by

$$\widehat{\theta}_t = (A_{\mathbf{x}_t}^\omega)^{-1} b_{\mathbf{x}_t}, \quad (3.16)$$

where $A_{\mathbf{x}_t}^\omega$ is defined by

$$A_{\mathbf{x}_t}^\omega = \omega I + \sum_{i=1}^t \chi_{M_i} \chi_{M_i}^\top,$$

for regularization parameter $\lambda > 0$ and the identity matrix I . In the remaining part of this section, we assume that the reward distribution ϕ_e is a R -sub-Gaussian distribution for all $e \in [d]$. For the regularized least-square estimator, we have another confidence bound which is also valid for adaptive strategies. In Abbasi-Yadkori et al. [2011], for linear bandits, they proposed another confidence bound which can be used even if \mathbf{x}_t is adaptively defined. Plugging their confidence bound into our setting, we have the following proposition.

Proposition 3.2 (Adapted from Abbasi-Yadkori et al. [2011, Theorem 2]). *Let ϵ_t be a R -sub-Gaussian noise. If l_2 -norm of parameter θ is less than S , then statement*

$$|x^\top \theta - x^\top \hat{\theta}_t| \leq C_t \|x\|_{(A_{\mathbf{x}_t}^\omega)^{-1}} \quad (3.17)$$

holds for all $t \in \{1, 2, \dots\}$ and $x \in \mathbb{R}^n$ with probability at least $1 - \delta$, where

$$C_t = R \sqrt{2k \log \frac{\det(A_{\mathbf{x}_t}^\omega)^{\frac{1}{2}}}{\omega^{\frac{d}{2}}}} + \omega^{\frac{1}{2}} S. \quad (3.18)$$

Moreover, if $\|x\| \leq \sqrt{k}$ holds for all $t > 0$, then

$$C_t \leq R \sqrt{2kd \log \frac{1 + tk/\omega}{\delta}} + \omega^{\frac{1}{2}} S. \quad (3.19)$$

We propose an algorithm named (**CLUCB-QM**) that employs an adaptive strategy based on Combinatorial Lower-Upper Confidence Bound (CLUCB) algorithm [Chen et al., 2014]. The CLUCB algorithm is originally designed for semi-bandit settings. We can modify the algorithm to solve full-bandit settings; whereas the original algorithm queries one base arm in one period, we can use it for the full-bandit settings by replacing the one base arm with two super arms that are different only by one base arm. However, with the original stopping condition used in Chen et al. [2014], the algorithm works poorly in terms of number of samples as we will observe in our experiments. In **CLUCB-QM** we propose, we use the stopping criterion as in **SAQM** that maintains the least-squares estimator and involves confidence ellipsoid maximization.

The entire procedure of **CLUCB-QM** is detailed in Algorithm 3.5. This algorithm keeps up confidence radius $\text{rad}_t(e)$ for each base arm $e \in [d]$. The vector $\tilde{\theta}_t(e)$ penalizes base arms belonging to the current empirical best super arm \widehat{M}_t^* and encourages exploring single-arms out of \widehat{M}_t^* . **CLUCB-QM** chooses a base arm e_t that has the largest confidence radius among the symmetric difference \widehat{M}_t^* and $\widetilde{M}_t = \arg\max_{M \in \mathcal{M}} \tilde{\theta}_t(M)$. The algorithm queries two super arms with one arm difference: **CLUCB-QM** pulls $M_t \in \mathcal{M}$ such that $e_t \in M_t$ and $e' \notin M_t$, and then pulls $M'_t = M_t \setminus \{e_t\} \cup \{e'\}$, where e' is a fixed base arm. Since **CLUCB-QM** employs the stopping condition (3.10) as in **SAQM**, we can prove that **CLUCB-QM** algorithm outputs ε -optimal super arm with probability at least $1 - \delta$. Although we do not have a sample complexity bound for **CLUCB-QM**, it performs better than **SAQM** in our experimental results.

3.9 Experiments

In this section, we evaluate the empirical performance of our algorithms, namely **SAQM** (Algorithm 3.3), **SA-FOA** (Algorithm 3.4), and **CLUCB-QM** (Algorithm 3.5).

3.9.1 Details of Baseline Algorithms

We implement a baseline algorithm namely **ICB** that works in polynomial-time. **ICB** employs simplified confidence bounds obtained by diagonal approximation of confidence ellipsoids.

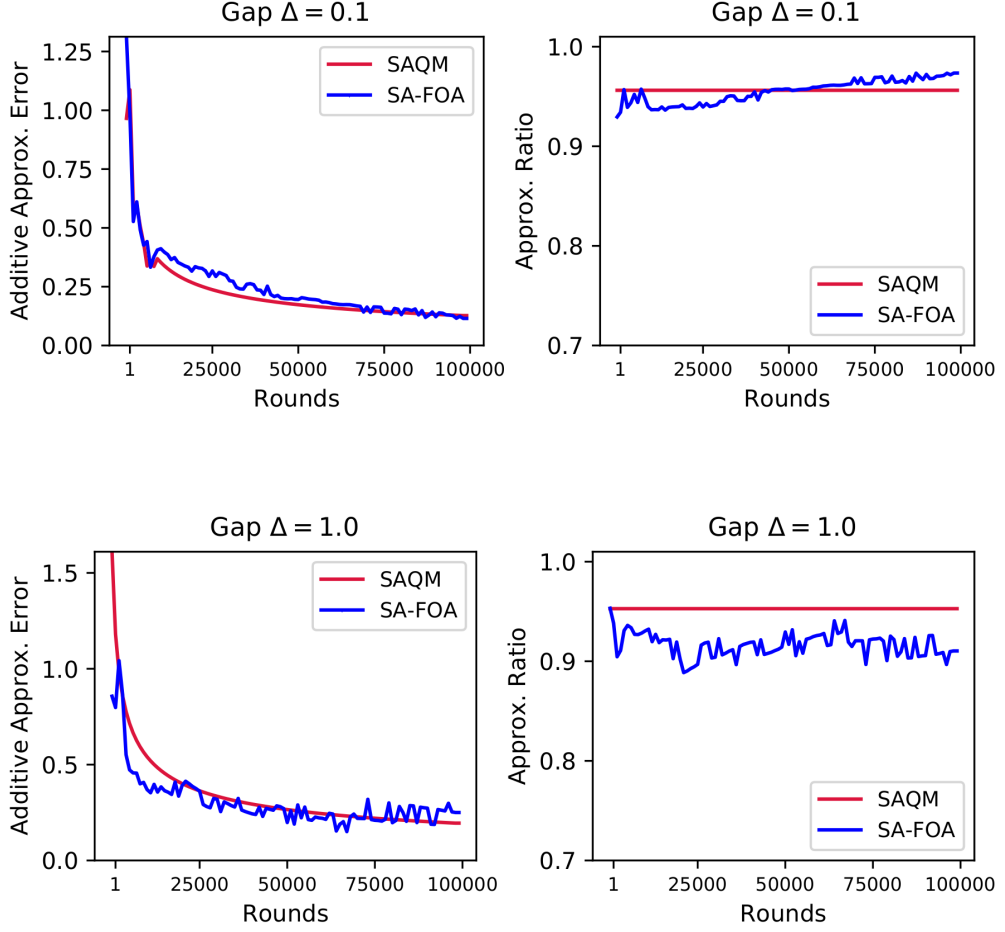


Figure 3.1: The results for approximation precision on synthetic datasets, where we set $(d, k) = (10, 5)$. Each point corresponds to an average over 10 realizations.

To verify the effectiveness of our novel stopping rule which uses confidence ellipsoids, we implement **CLUCB** (Algorithm 3.6); its stopping rule is different from **CLUCB-QM** but its sampling rule is same as **CLUCB-QM**. **CLUCB** estimates the gap between each base arm and a fixed arm by pulling two super-arms with one single-arm difference. **CLUCB** is based on the lower upper confidence bound algorithm for solving general constraints proposed by Chen et al. [2014] as in **CLUCB-QM**, and it employs a stopping condition that does not require confidence ellipsoid maximization. Notice that **CLUCB** is (ϵ, δ) -PAC.

We implement two baseline algorithms that invoke non-combinatorial top- k identification algorithms: one is an elimination-based algorithm **ME** (Algorithm 3.7 with **ME-Subroutine** Algorithm 3.8), the other is a confidence-bound-based algorithm **LUCB** (Algorithm 3.7 with **LUCB-Subroutine** Algorithm 3.9). **ME** employs the median elimination algorithm by Kalyanakrishnan and Stone [2010] for the non-combinatorial setting as a subroutine with simple modification; we sample $\{i\} \cup A$ when base arm $i \in [d]$ should be sampled for a fixed size- $(k - 1)$ subset $A \subseteq [d]$. **LUCB** is a counterpart of **ME** that employs the lower upper confidence based algorithm proposed by Kalyanakrishnan et al. [2012]. Note that **LUCB** and **ME** are also (ϵ, δ) -PAC.

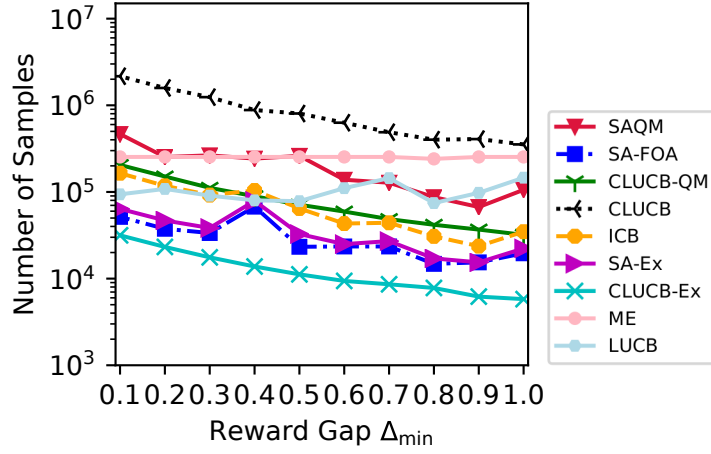


Figure 3.2: Run time in each round for synthetic datasets. Each point is an average over 10 realizations.

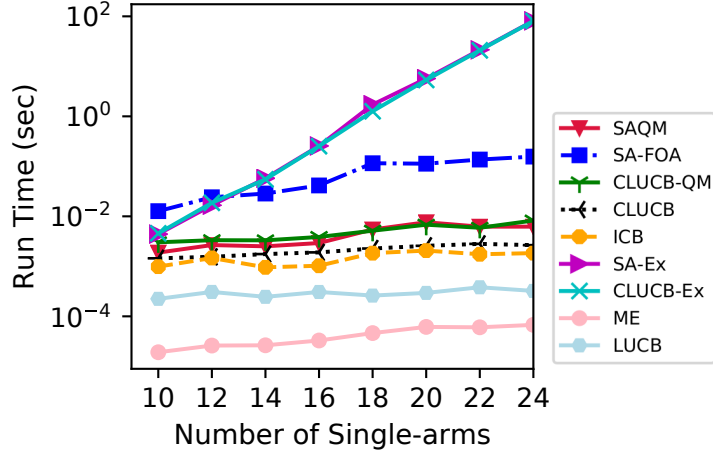


Figure 3.3: Number of samples for synthetic datasets with $(d, k) = (10, 5)$. Each point is an average over 10 realizations.

We compare our algorithms with two exponential time algorithms namely **SA-Ex** and **CLUCB-Ex**, which reduce our problem to the pure exploration problem in the linear bandit.

Now we will explain the details of above baseline algorithms and provide its sample complexity for some of them.

The detail of CLUCB. A baseline strategy **CLUCB** is given in Algorithm 3.6. **CLUCB** was proposed for combinatorial pure exploration problem where we can query each base arm [Chen et al., 2014]. With simple modification, we can use the original algorithm for the full-bandit setting. For a fixed base arm e' , **CLUCB** estimates the gap between each base arm and fixed e' by comparing two super-arm queries with one single-arm difference. The vector $\hat{\theta}'_t$ represents the empirical gap between each arm and fixed e' . The stopping condition employed

Algorithm 3.6: Combinatorial Lower-Upper Confidence Bound (CLUCB)

Input : Accuracy $\epsilon > 0$, confidence level $\delta \in (0, 1)$

- 1 **Initialization** Fix $e' \in [d]$. For each $e \in [d]$, pull $M_e \in \mathcal{M}$ such that $e \in M_e$ and $e' \notin M_e$ and pull $M'_t = M_t \setminus \{e_t\} \cup \{e'\}$. Initialize empirical gap $\widehat{\theta}'_t$ and set $T_e(t) \leftarrow 1$ for all $e \in [d]$.
- 2 **while** $\widetilde{\theta}'_t(\widetilde{M}_t) - \widehat{\theta}'_t(\widehat{M}_t^*) \leq \epsilon$ **is not true** **do**
- 3 $t \leftarrow t + 2$;
- 4 $\widehat{M}_t^* \leftarrow \operatorname{argmax}_{M \in \mathcal{M}} \widehat{\theta}'_t(M)$;
- 5 Compute confidence radius $\operatorname{rad}_t(e) = R \sqrt{\frac{2k \log\left(\frac{4dt^3}{\delta}\right)}{T_e(t)}}$ for all $e \in [d]$;
- 6 **for** $e = 1, \dots, d$ **do**
- 7 **if** $e \in \widehat{M}_t^*$ **then** $\widetilde{\theta}'_t(e) = \widehat{\theta}'_t(e) - \operatorname{rad}_t(e)$;
- 8 **else** $\widetilde{\theta}'_t(e) \leftarrow \widehat{\theta}'_t(e) + \operatorname{rad}_t(e)$;
- 9 $\widetilde{M}_t \leftarrow \operatorname{argmax}_{M \in \mathcal{M}} \widetilde{\theta}'_t(M)$;
- 10 $e_t \leftarrow \operatorname{argmax}_{e \in \widetilde{M}_t \setminus \widehat{M}_t^* \cup \widehat{M}_t^* \setminus \widetilde{M}_t} \operatorname{rad}_t(e)$;
- 11 $T_t(e_t) \leftarrow T_{t-2}(e_t) + 1$;
- 12 Pull $M_t \in \mathcal{M}$ such that $e_t \in M_t$ and $e' \notin M_t$;
- 13 Pull $M'_t = M_t \setminus \{e_t\} \cup \{e'\}$;
- 14 Observe r_{M_t} and $r_{M'_t}$, and then update $\widehat{\theta}'_t$;
- 15 **return** $\widehat{M}^* \leftarrow \widehat{M}_t^*$

Algorithm 3.7: Baseline framework

Input : Accuracy $\epsilon > 0$, confidence level $\delta \in (0, 1)$,

- 1 Put $k - 1$ arms uniformly at random in set A and the remaining $n - k + 1$ arms in set B ;
- 2 Find ϵ -optimal k -arms in B by Subroutine($B, k, \frac{\epsilon}{2k}, \frac{\delta}{2}, A$) (Algorithm 3.8 or 3.9) and let B^* be its output;
- 3 Choose $k - 1$ arms from $B \setminus B^*$ uniformly at random and let them be a set B_{bad} ;
- 4 Find ϵ -optimal k -arms in $B^* \cup A$ by Subroutine($B^* \cup A, k, \frac{\epsilon}{2k}, \frac{\delta}{2}, B_{\text{bad}}$) (Algorithm 3.8 or 3.9) and let M^* be its output;
- 5 **return** M^* ;

by CLUCB is

$$\widetilde{\theta}'_t(\widetilde{M}_t) - \widehat{\theta}'_t(\widehat{M}_t^*) \leq \epsilon,$$

as in the original algorithm [Chen et al., 2014]. The variance becomes $O(k)$ times larger than the one in the setting where single-arm can be queried. Therefore, CLUCB is (ϵ, δ) -PAC algorithm with sample complexity bound as follows.

Proposition 3.3 (Chen et al. [2014, Theorem 1]). *With probability at least $1 - \delta$, CLUCB algorithm (Algorithm 3.6) returns ϵ -optimal super-arm \widehat{M}^* and the*

Algorithm 3.8: ME-Subroutine($B, k, \epsilon', \delta', A$)

Input : a set A , a set B , size k , accuracy $\epsilon' > 0$, confidence level $\delta' \in (0, 1)$

- 1 $d' \leftarrow |B|$
- 2 $M_1 \leftarrow B$;
- 3 $\epsilon_1 \leftarrow \epsilon'$ and $\delta_1 \leftarrow \delta'/2$;
- 4 **for** $l = 1, 2, \dots, \lceil \log(d'/k) \rceil$ **do**
- 5 **for** each $e \in M_l$ **do**
- 6 Sample arm $e \cup A$ for $\lceil \frac{2\sqrt{k}}{\epsilon_l^2} \ln \left(\frac{3k}{\delta_l} \right) \rceil$ times;
- 7 Let \hat{p}_e be its average reward;
- 8 Find $M'_l \subset M_l$ such that $|M'_l| = \max(\lceil |M_l|/2 \rceil, k)$ and $\hat{p}_i \geq \hat{p}_j$ for all $i \in M'_l$ and $j \in M_l \setminus M'_l$;
- 9 $M_{l+1} \leftarrow M'_l$;
- 10 $\epsilon_{l+1} \leftarrow \frac{3}{4}\epsilon_l$ and $\delta_{l+1} \leftarrow \frac{1}{2}\delta_l$;
- 11 **return** $M_{\lceil \log(d'/k) \rceil}$.

Algorithm 3.9: LUCB-Subroutine($B, k, \epsilon', \delta', A$)

Input : a set A , a set B , size k , accuracy $\epsilon' > 0$, confidence level $\delta' \in (0, 1)$

- 1 Pull $\{e\} \cup A$ and initialize its empirical mean $\hat{\theta}_e^A$ for each $e \in B$.
- 2 **while** $(\hat{\theta}_{l_*^t}^A + \beta(l_*^t, t)) - (\hat{\theta}_{h_*^t}^A - \beta(h_*^t, t)) \geq \epsilon'$ **do**
- 3 $t \leftarrow t + 2$;
- 4 Compute confidence bound $\beta(e, t) \leftarrow \sqrt{\frac{k}{2T_e(t)} \ln \left(\frac{5|B|t^4}{4\delta'} \right)}$ for all $e \in B$;
- 5 Find $\text{High}^t \subset B$ such that $|\text{High}^t| = k$ and $\hat{\theta}_i^A \geq \hat{\theta}_j^A$ for all $i \in \text{High}^t$ and $j \in B \setminus \text{High}^t$;
- 6 $\text{Low}^t \leftarrow B \setminus \text{High}^t$;
- 7 $h_*^t \leftarrow \operatorname{argmin}_{h \in \text{High}^t} \{\hat{\theta}_h^A - \beta(h, t)\}$;
- 8 $l_*^t \leftarrow \operatorname{argmax}_{l \in \text{Low}^t} \{\hat{\theta}_l^A + \beta(l, t)\}$;
- 9 Pull $\{h_*^t\} \cup A$ and $\{l_*^t\} \cup A$, and update its empirical mean;
- 10 **return** High^t

number of samples T is bounded as

$$T \leq O \left(kR^2 \sum_{e \in [d]} \min \left\{ \frac{2}{\Delta_e^2}, \frac{k^2}{\epsilon^2} \right\} \log \left(\frac{kR^2}{\delta} \sum_{e \in [d]} \min \left\{ \frac{2}{\Delta_e^2}, \frac{k^2}{\epsilon^2} \right\} \right) \right),$$

where for each base arm $e \in [d]$, the gap Δ_e is defined as

$$\Delta_e = \begin{cases} \theta(M^*) - \max_{M \in \mathcal{M}: e \in M} \sum_{e \in M} \theta_e & (\text{if } e \notin M^*) \\ \theta(M^*) - \max_{M \in \mathcal{M}: e \notin M} \sum_{e \in M} \theta_e & (\text{if } e \in M^*). \end{cases} \quad (3.1)$$

CLUCB algorithm is efficient in terms of computational time since the algorithm runs in polynomial time. However, as observed in our experiments, this

Algorithm 3.10: Static Allocation with Exhaustive Search (SA-Ex)

Input : Accuracy $\varepsilon > 0$, confidence level $\delta \in (0, 1)$, allocation strategy p

```

1 for  $t = 1, \dots, d$  do
2   Pull  $M_t \in \text{supp}(p)$ ;
3   Observe  $r_t$ ;
4   Update  $A_t$  and  $b_t$ ;
5 while  $Z_t^* \geq \varepsilon$  do
6    $t \leftarrow t + 1$ ;
7   Pull  $M_t \leftarrow \text{argmin}_{M \in \text{supp}(p)} \frac{T_M(t)}{p_M}$ ;
8   Observe  $r_t$ ;
9   Update  $A_t$  and  $b_t$ ;
10   $\widehat{M}_t^* \leftarrow \text{argmax}_{M \in \mathcal{M}} \widehat{\theta}_t(M)$ ;
11   $Z_t^* \leftarrow \max_{M \in \mathcal{M} \setminus \{\widehat{M}^*\}} \left( \widehat{\theta}_t(M) + C_t \|\chi_M - \chi_{M^*}\|_{A_{\mathbf{x}_t}^{-1}} \right) - \widehat{\theta}_t(\widehat{M}^*)$ ;
12 return  $\widehat{M}^* \leftarrow \widehat{M}_t^*$ ;

```

naive baseline algorithm does not work well especially for real-world datasets, since it cannot exploit the information from $k - 1$ arms at each pull.

The detail of ME and LUCB. The entire procedure of **ME** is detailed in Algorithm 3.7 with ME-subroutine Algorithm 3.8. First, we choose any subset A of $k - 1$ arms and find the best k subset B^* among $B = [d] \setminus A$ by the median elimination algorithm proposed by Kalyanakrishnan and Stone [2010]. When $i \in B$ should be pulled, we pull the super arm $\{i\} \cup A$ instead of i . By this procedure, we can get the k arms $i \in B$ to maximize $\theta(\{i\} \cup A)$, which are equal to the arms $i \in B$ to maximize $\theta(i)$. Clearly, the best k subset in $[d]$ can be obtained by finding the best k subset in $A \cup B^*$. **ME** is (ϵ, δ) -PAC and its sample complexity is $O\left(\frac{dk^2}{\epsilon^2} \log\left(\frac{k}{\delta}\right)\right)$.

LUCB is a counterpart of **ME**, which is detailed in Algorithm 3.7 with LUCB-Subroutine Algorithm 3.9. **LUCB** employs the lower upper confidence algorithm proposed by Kalyanakrishnan et al. [2012] whose sample complexity is

$$O\left(\sum_{e \in [d]} \frac{1}{\max\{\Delta_e, \epsilon\}^2} \log\left(\frac{1}{\max\{\Delta_e, \epsilon\}^2 \delta}\right)\right),$$

where Δ_e is defined by (3.1). Since this subroutine is (ϵ, δ) -PAC, we see that **LUCB** is also (ϵ, δ) -PAC. However, the sample complexity can be very large when the gap between the best k -th and $(k + 1)$ -th arm in the instance for its first subroutine is much smaller than Δ_{\min} .

ME and **LUCB** are computationally efficient since they run in polynomial time. However, they must invoke a subroutine two times, which will increase the number of samples.

The detail of exponential algorithms. The entire procedure of **SA-Ex** is detailed in Algorithm 3.10. In this algorithm and **SAQM**, the difference is only

Algorithm 3.11: Combinatorial Lower-Upper Confidence Bound with Exhaustive Search (CLUCB-Ex)

Input : Accuracy $\varepsilon > 0$, confidence level $\delta \in (0, 1)$,

- 1 **Initialization** Fix $e' \in [d]$. For each $e \in [d]$, pull $M_e \in \mathcal{M}$ such that $e \in M_e$ and $e' \notin M_e$ and pull $M'_t = M_t \setminus \{e_t\} \cup \{e'\}$. Initialize empirical gap $\widehat{\theta}'_t$ and set $T_e(t) \leftarrow 1$ for all $e \in [d]$.
- 2 **while** $Z_t^* \geq \varepsilon$ **do**
- 3 $t \leftarrow t + 2$;
- 4 $\widehat{M}_t^* \leftarrow \operatorname{argmax}_{M \in \mathcal{M}} \widehat{\theta}'_t(M)$;
- 5 Compute confidence radius $\operatorname{rad}_t(e) = R \sqrt{\frac{2k \log\left(\frac{4dt^3}{\delta}\right)}{T_e(t)}}$ for all $e \in [d]$;
- 6 **for** $e = 1, \dots, d$ **do**
- 7 **if** $e \in \widehat{M}_t^*$ **then** $\widetilde{\theta}'_t(e) = \widehat{\theta}'_t(e) - \operatorname{rad}_t(e)$;
- 8 **else** $\widetilde{\theta}'_t(e) \leftarrow \widehat{\theta}'_t(e) + \operatorname{rad}_t(e)$;
- 9 $\widetilde{M}_t \leftarrow \operatorname{argmax}_{M \in \mathcal{M}} \widetilde{\theta}'_t(M)$;
- 10 $e_t \leftarrow \operatorname{argmax}_{e \in \widetilde{M}_t \setminus \widehat{M}_t^* \cup \widehat{M}_t^* \setminus \widetilde{M}_t} \operatorname{rad}_t(e)$;
- 11 $T_t(e_t) \leftarrow T_{t-2}(e_t) + 1$;
- 12 Choose $M_t \in \mathcal{M}$ such that $e_t \in M_t$ and $e' \notin M_t$ uniformly at random;
- 13 Pull $M_t \in \mathcal{M}$ and $M'_t = M_t \setminus \{e_t\} \cup \{e'\}$;
- 14 Observe r_{M_t} and $r_{M'_t}$, and then update $\widehat{\theta}'_t$;
- 15 $\widehat{M}_t^* \leftarrow \operatorname{argmax}_{M \in \mathcal{M}} \widehat{\theta}'_t(M)$;
- 16 $Z_t^* \leftarrow \max_{M \in \mathcal{M} \setminus \{\widehat{M}^*\}} \left(\widehat{\theta}_t(M) + C_t \|\chi_M - \chi_{M^*}\|_{A_{\mathbf{x}_t}^{-1}} \right) - \widehat{\theta}_t(\widehat{M}^*)$;
- 17 **return** $\widehat{M}^* \leftarrow \widehat{M}_t^*$;

the stopping condition. In **SAQM**, we approximately solve confidence ellipsoid maximization, whereas **SA-Ex** conducts an exhaustive search to obtain the exact solution. Thus, **SA-Ex** runs in exponential time, i.e, $O(d^k)$. The entire procedure of **CLUCB-Ex** is detailed in Algorithm 3.11. This algorithm also reduces our problem to the pure exploration problem in the linear bandit, and thus runs in exponential time. The stopping condition used in **CLUCB-Ex** and **SA-Ex** is same. **CLUCB-Ex** is adaptively pulls super-arms based on CLUCB strategy as in **CLUCB** and **CLUCB-QM**.

3.9.2 Experimental Settings and Results

We conduct the experiments on small synthetic datasets and large-scale real-world datasets. All experiments were conducted on a Macbook with a 1.3 GHz Intel Core i5 and 8GB memory. All codes were implemented by using Python. In all experiments, we employed the approximation algorithm called the *greedy peeling* [Asahiro et al., 2000] as the DkS -Oracle. Specifically, the greedy peeling algorithm iteratively removes a vertex with the minimum weighted degree in the currently remaining graph until we are left with the subset of vertices with size k . The algorithm runs in $O(d^2)$.

Table 3.1: Real-world datasets on crowdsourcing. “Average” and “Best” give the average and the best accuracy rate among the workers, respectively.

Dataset	#task	#worker	Average	Best	Δ_{\min}
<i>IT</i>	25	36	0.54	0.84	0.04
<i>Medicine</i>	36	45	0.48	0.92	0.03
<i>Chinese</i>	24	50	0.37	0.79	0.04
<i>Pokémon</i>	20	55	0.28	1.00	0.05
<i>English</i>	30	63	0.26	0.70	0.03
<i>Science</i>	20	111	0.29	0.85	0.05

Table 3.2: Number of samples ($\times 10^2$) on real-world crowdsourcing datasets. Each value is an average over 10 realizations.

Dataset	SAQM	SA-FOA	CLUCB-QM	CLUCB	ME	LUCB	ICB
<i>IT</i>	62,985	1,437	9,896	405,313	111,603	91,442	43,773
<i>Medicine</i>	96,174	865	15,678	400,953	139,504	109,124	66,468
<i>Chinese</i>	88,209	1,060	19,438	754,439	301,635	129,795	99,424
<i>Pokémon</i>	83,209	328	1,994	151,748	331,799	89,674	19,705
<i>English</i>	121,890	1,023	31,300	671,274	380,060	117,611	114,406
<i>Science</i>	276,325	1,505	100,950	1,825,106	1,292,074	224,494	418,155

Synthetic datasets. To see the dependence of the performance on the minimum gap Δ_{\min} , we generate synthetic instances as follows. We first set the expected rewards for the top- k base arms uniformly at random from $[0, 1]$. Let $\theta_{\min-k}$ be the minimum expected reward in the top- k base arms. We set the expected reward of the $(k + 1)$ -th best base arm to $\theta_{\min-k} - \Delta_{\min}$ for the predetermined parameter $\Delta_{\min} \in [0, 1]$. Then, we generate the expected rewards of the rest of base arms by uniform samples from $[-1, \theta_{\min-k} - \Delta_{\min}]$ so that expected rewards of the best super arm is larger than those of the rest of super arms by at least Δ_{\min} . We set the additive noise distribution $\mathcal{N}(0, 1)$. In all instances we set $\delta = 0.05$ and $\varepsilon = 0.5$. For a regularization parameter in CLUCB-QM, we set $\lambda = 1$ as in Xu et al. [2018]. SAQM, SA-FOA and ICB employ G-allocation strategy. In SA-FOA, ℓ is set to 2 for all experiments.

Approximation error. First, we examine the approximation precision of our approximation algorithms. The results are reported in Figure 3.1. SAQM and SA-FOA employ some approximation mechanisms to test the stopping condition in polynomial time. Recall that SAQM approximately solves CEM in (3.4) to attain an objective value of Z_t , and SA-FOA approximately solves the maximization problem in (3.11) to attain an objective value of Z'_t . We set up the experiments with $d = 10$ base arms and $k = 5$. We run the experiments for the small gap ($\Delta_{\min} = 0.1$) and large gap ($\Delta_{\min} = 1.0$). We plot the approximation ratio and the additive approximation error of SAQM and SA-FOA in the first 100,000 rounds. From the results, we can see that the approximation ratios of them are almost

always greater than 0.9, which are far better than the worst-case guarantee proved in Theorem 3.1. In particular, the approximation ratio of **SA-FOA** in the small gap case is surprisingly good (around 0.95) and grows as the number of rounds increases. This result implies that there is only a slight increase of the sample complexity caused by the approximation, especially when the expected rewards of base arms are close to each other.

Running time. Next, we conduct the experiments to compare the running time of algorithms. We set $d = 10, 12, \dots, 24$ and $k = d/2$ on synthetic datasets. We report the computational time in one round in Figure 3.2. Since **ME** is an elimination-based algorithm, we report the computational time which is overall time t divided by the number of samples required by the algorithm. As can be seen, **SA-Ex** and **CLUCB-Ex** are prohibitive on instances with large number of super arms, while our algorithms can run fast even if n becomes larger, which matches our theoretical analysis. The results indicate that polynomial-time algorithms are of crucial importance for practical use.

Number of samples. Finally, we evaluate the number of samples required to identify the best super arm for varying Δ_{\min} . Based on the above observation, we set $\alpha = 0.9$. The result is shown in Figure 3.3. We observed that our algorithms always output the optimal super arm. The result indicates that the numbers of samples of our algorithms are comparable to that of **SA-Ex** and **CLUCB-Ex**. Notice that **LUCB** does not show a decreasing trend in Figure 3.3. The reason may be that **LUCB** reduces the problem into two instances, where the gap between the best k -th and $(k + 1)$ -th arm can be no longer Δ_{\min} .

Performance on real-world crowdsourcing datasets. We use the crowdsourcing datasets compiled by Li et al. [2017] whose basic information is shown in Table 5.2. The task is to identify the top- k workers with the highest accuracy only from a sequential access to the accuracy of part of labels given by some workers. Notice that the number of super arms is more than 10^{10} in all experiments. All datasets are hard instances as Δ_{\min} is less than 0.05. We set $k = 10$ and $\varepsilon = 0.5$. Since **SA-Ex** and **CLUCB-Ex** are prohibitive, we compare the other algorithms. **SAQM**, **SA-FOA** and **ICB** employ uniform allocation strategy.

The result is shown in Table 3.2, which indicates the applicability of our algorithms to the instances with a massive number of super arms. Moreover, all algorithms found the optimal subset of crowdworkers. In all datasets, **SA-FOA** outperformed the other algorithms. Recall that **ICB** uses the simplified confidence bound for the gap between two super arms. On the other hand, **SA-FOA** uses the approximation of the confidence ellipsoids for the gap between two super arms, which results in better performance than **ICB**. **SAQM** approximately computes the maximal confidence ellipsoid bound for the reward of one super arm rather than the gap between two super arms, which may result in worse performance than **SA-FOA**. **CLUCB-QM**, which employs the same sampling rule as **CLUCB** and the same stopping rule as **SAQM**, performed better than **CLUCB** and **SAQM**. This result may indicate that an adaptive sampling rule is more desired than a static sampling rule, and using a confidence ellipsoid is more desired than considering an individual confidence bound. **ME**, **LUCB** and **CLUCB** discard the in-

formation from $k - 1$ arms at each pull, which may cause the unfavorable results. LUCB worked better than CLUCB, since the original version of CLUCB was designed for very general combinatorial constraints while LUCB was designed only for the top- k setting. Notice that ME is phased adaptive while LUCB is fully adaptive; ME performed poorly in all instances although ME is the counterpart of LUCB.

3.10 Proofs of Theoretical Results

In this section, we give proofs of all the theorems and lemmas in this chapter.

First, we introduce the notation. For $M, M' \in \mathcal{M}$, let $\Delta(M, M')$ be the *value gap* between two super arms, i.e., $\Delta(M, M') = |\theta(M) - \theta(M')|$. Also, let $\widehat{\Delta}(M, M')$ be the *empirical gap* between two super arms, i.e., $\widehat{\Delta}(M, M') = |\widehat{\theta}_t(M) - \widehat{\theta}_t(M')|$.

3.10.1 Proof of Lemma 3.3

Proof. First we define random event \mathcal{E}_t as follows:

$$\mathcal{E}_t = \left\{ \forall M, M' \in \mathcal{M}, |\widehat{\theta}_t(M) - \widehat{\theta}_t(M')| \leq C_t \sum_{i=1}^n |\chi_M(i) - \chi_{M'}(i)| \sqrt{A_{\mathbf{x}_t}^{-1}(i, i)} \right\}.$$

We notice that random event \mathcal{E}_t implies that the event that the confidence intervals of all super arm $M \in \mathcal{M}$ are valid at round t . From Lemma 3.1, we see that the probability that event $\mathcal{E} = \cap_{t=1}^{\infty} \mathcal{E}_t$ occurs is at least $1 - \delta$. Under the event \mathcal{E} , we see that the output \widehat{M}^* is an ε -optimal super arm. In the rest of the proof, we shall assume that event \mathcal{E} holds. Next, we focus on bounding the sample complexity T . By recalling the stopping condition (3.8), a sufficient condition for stopping is that for M^* and for $t > d$,

$$\varepsilon > \max_{M \in \mathcal{M} \setminus \{M^*\}} \left(\widehat{\theta}_t(M) + C_t \sum_{i=1}^n |\chi_M(i) - \chi_{M^*}(i)| \sqrt{A_{\mathbf{x}_t}^{-1}(i, i)} \right) - \widehat{\theta}_t(M^*). \quad (3.1)$$

We define \overline{M} as

$$\overline{M} = \operatorname{argmax}_{M \in \mathcal{M} \setminus \{M^*\}} \left(\widehat{\theta}_t(M) + C_t \sum_{i=1}^n |\chi_M(i) - \chi_{M^*}(i)| \sqrt{A_{\mathbf{x}_t}^{-1}(i, i)} \right).$$

Eq. (3.1) is satisfied if

$$\widehat{\Delta}(M^*, \overline{M}) > C_t \sqrt{\frac{\rho'(\lambda)}{t}} - \varepsilon. \quad (3.2)$$

From Lemma 1 with $x = \chi_{M^*} - \chi_{\overline{M}}$, with probability at least $1 - \delta$, we have

$$\begin{aligned}\widehat{\Delta}(M^*, \overline{M}) &\geq \Delta(M^*, \overline{M}) - C_t \sum_{i=1}^n |\chi_{M^*}(i) - \chi_{\overline{M}}(i)| \sqrt{A_{\mathbf{x}_t}^{-1}(i, i)} \\ &\geq \Delta(M^*, \overline{M}) - C_t \sqrt{\frac{\rho'(\lambda)}{t}}.\end{aligned}\tag{3.3}$$

Combining (3.2) and (3.3), we see that a sufficient condition for stopping is given by $\Delta(M^*, \overline{M}) \geq \Delta_{\min} \geq 2C_t \sqrt{\frac{\rho(\lambda)}{t}} - \varepsilon$. Therefore, we have $t \geq 4C_t^2 H_\varepsilon$ as a sufficient condition to stop. Let $\tau > d$ be the stopping time of the algorithm. From the above discussion, we see that $\tau \leq 4C_\tau^2 H_\varepsilon$. Recalling that $C_t = \sigma \sqrt{2 \log(c't^2 d/\delta)}$, we have $\tau \leq 8\sigma^2 \log(c'\tau^2 d/\delta) H_\varepsilon$. Let τ' be a parameter that satisfies

$$\tau = 8\sigma^2 \log(c'\tau'^2 d/\delta) H_\varepsilon.\tag{3.4}$$

Then, it is obvious that $\tau' \leq \tau$ holds. For N defined as $N = 8\sigma^2 \log(c'd/\delta) H_\varepsilon$, we have

$$\tau' \leq \tau = 16\sigma^2 \log(\tau') H_\varepsilon + N \leq 16\sigma^2 \sqrt{\tau'} H_\varepsilon + N$$

Transforming this inequality, we obtain

$$\sqrt{\tau'} \leq 8\sigma^2 H_\varepsilon + \sqrt{64\sigma^4 H_\varepsilon^2 + N} \leq 2\sqrt{64\sigma^4 H_\varepsilon^2 + N}.\tag{3.5}$$

Let $L = 2\sqrt{64\sigma^4 H_\varepsilon^2 + N}$, which equals the RHS of (3.5). We see that

$$\log L = O\left(\log\left(\sigma H_\varepsilon \left(\sigma^2 H_\varepsilon + \log\left(\frac{d}{\delta}\right)\right)\right)\right).$$

Then, using this upper bound of τ' in (3.4), we have

$$\tau \leq 16\sigma^2 H_\varepsilon \log\left(\frac{c'd}{\delta}\right) + C(H_\varepsilon, \delta),$$

where

$$C(H_\varepsilon, \delta) = O\left(\sigma^2 H_\varepsilon \log\left(\sigma H_\varepsilon \left(\sigma^2 H_\varepsilon + \log\left(\frac{d}{\delta}\right)\right)\right)\right).$$

Recalling that $\sigma = kR$, we obtain

$$\tau = O\left(k^2 R^2 H_\varepsilon \log\left(\frac{d}{\delta} \left(k R H_\varepsilon \left(k^2 R^2 H_\varepsilon + \log\left(\frac{d}{\delta}\right)\right)\right)\right)\right).$$

□

3.10.2 Proof of Theorem 3.1

We begin by showing the following three lemmas.

Lemma 3.4. *Let $W \in \mathbb{R}^{d \times d}$ be any positive definite matrix. Then $\tilde{G} = (V, E, \tilde{w})$ constructed by Algorithm 3.2 is a non-negative weighted graph.*

Proof. For any $(i, j) \in V^2$, we have $w_{ii} \geq 0$ and $w_{jj} \geq 0$ since W is a positive definite matrix. If $w_{ij} \geq 0$, it is obvious that $\tilde{w}_{ij} = w_{ij} + w_{ii} + w_{jj} \geq 0$. We consider the case $w_{ij} < 0$. In the case, we have $w_{ij} + w_{ii} + w_{jj} > 2w_{ij} + w_{ii} + w_{jj} \geq 0$, where the last inequality holds from the definition of positive definite matrix W . Thus, we obtain the desired result. \square

Lemma 3.5. *Let $W \in \mathbb{R}^{d \times d}$ be any positive definite matrix and $\tilde{W} = (\tilde{w}_{ij})$ be the adjacency matrix of the complete graph constructed by Algorithm 3.2. Then, for any $S \subseteq V$ such that $|S| \geq 2$, we have $w(S) \leq \tilde{w}(S)$.*

Proof. We have

$$\begin{aligned} w(S) &= \sum_{e \in E(S)} w_e = \sum_{\{i,j\} \in E(S): i \neq j} w_{ij} + \sum_{i \in S} w_{ii} \\ &\leq \sum_{\{i,j\} \in E(S): i \neq j} w_{ij} + (|S| - 1) \sum_{i \in S} w_{ii} = \tilde{w}(S), \end{aligned}$$

where the last inequality holds since each diagonal component w_{ii} is positive for all $i \in V$ from the definition of the positive definite matrix. \square

Lemma 3.6. *Let $W \in \mathbb{R}^{d \times d}$ be any positive definite matrix and $\tilde{W} = (\tilde{w}_{ij})$ be the adjacency matrix of the complete graph constructed in Algorithm alg:QM. Then, for any subset of vertices $S \subseteq V$, we have $\frac{\tilde{w}(S)}{w(S)} \leq (|S| - 1) \frac{\xi_{\max}(W)}{\xi_{\min}(W)}$, where $\xi_{\min}(W)$ and $\xi_{\max}(W)$ represent the minimum and maximum eigenvalues of W , respectively.*

Proof. We consider the following two cases: Case (i) $\sum_{\{i,j\} \in E(S): i \neq j} w_{ij} \geq 0$ and Case (ii) $\sum_{\{i,j\} \in E(S): i \neq j} w_{ij} < 0$.

Case (i) Since $W = (w_{ij})_{1 \leq i,j \leq d}$ is positive definite matrix, we see that diagonal component w_{ii} is positive for all $i \in V$. Thus, we have

$$\begin{aligned} \tilde{w}(S) &= \sum_{(i,j) \in E(S): i \neq j} w_{ij} + (|S| - 1) \sum_{i \in S} w_{ii} \\ &\leq (|S| - 1) \left(\sum_{(i,j) \in E(S): i \neq j} w_{ij} + \sum_{i \in S} w_{ii} \right) = (|S| - 1)w(S). \end{aligned}$$

Since W is positive definite, we have $w(S) > 0$. That gives us the desired result.

Case (ii) In this case, we see that

$$\tilde{w}(S) = \sum_{(i,j) \in E(S): i \neq j} w_{ij} + (|S| - 1) \sum_{i \in S} w_{ii} \leq (|S| - 1) \sum_{i \in S} w_{ii}.$$

For any diagonal component w_{ii} we have that $w_{ii} \leq \max_{1 \leq i, j \leq d} w_{ij}$. For the largest component $\max_{1 \leq i, j \leq d} w_{ij}$, we have

$$\max_{1 \leq i, j \leq d} w_{ij} \leq \max_{1 \leq i, j \leq d} \frac{1}{2} e_i^\top W e_i + \frac{1}{2} e_j^\top W e_j \leq \xi_{\max}(W),$$

where the first inequality is satisfied since W is positive definite. Thus, we obtain

$$\tilde{w}(S) < |S|(|S| - 1) \xi_{\max}(W). \quad (3.6)$$

For the lower bound of $w(S)$, we have

$$w(S) = \chi_s^\top W \chi_s = \frac{\chi_s^\top W \chi_s}{\|\chi_s^\top \chi_s\|_2^2} |S| > \xi_{\min}(W) |S|. \quad (3.7)$$

Combining (3.6) and (3.7), we obtain

$$\frac{\tilde{w}(S)}{w(S)} < (|S| - 1) \frac{\xi_{\max}(W)}{\xi_{\min}(W)},$$

which completes the proof. □

We are now ready to prove Theorem 3.1.

Proof of Theorem 3.1. For any round $t > d$, let χ_{s_t} be the approximate solution obtained by Algorithm 3.2 and Z be its objective value. Let $\tilde{w} : 2^{[d]} \rightarrow \mathbb{R}$ be the weight function defined by Algorithm 3.2. We denote the optimal value of QP by OPT. Let us denote optimal solution of the DkS for $G([d], E, \tilde{w})$ by \tilde{S}_{OPT} . Adjacency matrix W is a symmetric positive definite matrix; thus, Lemma 3.4, 3.5 and 3.6 hold for W . We have

$$\begin{aligned} \tilde{w}(S_t) &\geq \alpha_{\text{DkS}} \tilde{w}(\tilde{S}_{\text{OPT}}) & (\because S_t \text{ is an } \alpha_{\text{DkS}}\text{-approximate solution for DkS}(\tilde{G}).) \\ &\geq \alpha_{\text{DkS}} \tilde{w}(S_{\text{OPT}}) \\ &\geq \alpha_{\text{DkS}} w(S_{\text{OPT}}). & (\because \text{Lemma 3.5}). \end{aligned} \quad (3.8)$$

Thus, we obtain

$$\begin{aligned} Z &= \chi_{s_t}^\top A_{\mathbf{x}_t}^{-1} \chi_{s_t} = w(S_t) \\ &\geq \frac{1}{|S| - 1} \frac{\xi_{\min}(W)}{\xi_{\max}(W)} \tilde{w}(S_t) & (\because \text{Lemma 3.6}) \end{aligned}$$

$$\begin{aligned}
&\geq \frac{1}{|S| - 1} \frac{\xi_{\min}(W)}{\xi_{\max}(W)} \alpha_{\text{DkS}} w(S_{\text{OPT}}) \quad (\because (3.8)) \\
&= \frac{1}{|S| - 1} \frac{\xi_{\min}(W)}{\xi_{\max}(W)} \alpha_{\text{DkS}} \text{OPT}.
\end{aligned}$$

Therefore, we obtain $Z \geq \left(\frac{1}{k-1} \frac{\xi_{\min}(W)}{\xi_{\max}(W)} \alpha_{\text{DkS}} \right) \text{OPT}$. \square

3.10.3 Proof of Theorem 3.2

Proof. Updating $A_{\mathbf{x}_t}^{-1}$ can be done in $O(d^2)$ time, and computing the empirical best super arm can be done in $O(d)$ time. Moreover, confidence maximization CEM can be approximately solved in polynomial-time, since quadratic maximization QP is solved in polynomial-time as long as we employ polynomial time algorithm as the DkS-Oracle. Let $\text{poly}(d)_{\text{DkS}}$ be the computation time of DkS-Oracle. Then, we can guarantee that **SAQM** runs in $O(\max\{d^2, \text{poly}(d)_{\text{DkS}}\})$ time. \square

3.10.4 Proof of Theorem 3.3

Before stating the proof of Theorem 3.3, we give the technical lemmas.

For any $t > 0$, let us define random event \mathcal{E}'_t as

$$\left\{ \forall M \in \mathcal{M}, |\theta(M) - \hat{\theta}_t(M)| \leq C_t \|\chi_M\|_{A_{\mathbf{x}_t}^{-1}} \right\}. \quad (3.9)$$

We note that random event \mathcal{E}'_t characterizes the event that the confidence bounds of all super arm $M \in \mathcal{M}$ are valid at round t . Next lemma indicates that, if the confidence bounds are valid, then **SAQM** always outputs ε -optimal super arm \widehat{M}^* when it stops.

Lemma 3.7. *Given any $t > d$, assume that \mathcal{E}'_t occurs. Then, if **SAQM** (Algorithm 3.3) terminates at round t , we have $\theta(M^*) - \theta(\widehat{M}^*) \leq \varepsilon$.*

Proof of Lemma 3.7. If $\widehat{M}^* = M^*$, we have the desired result. Then, we shall assume $\widehat{M}^* \neq M^*$. We have the following inequalities:

$$\begin{aligned}
\theta(\widehat{M}^*) &\geq \hat{\theta}_t(\widehat{M}^*) - C_t \|\chi_{\widehat{M}^*}\|_{A_{\mathbf{x}_t}^{-1}} \\
&\geq \max_{M \in \mathcal{M} \setminus \{\widehat{M}^*\}} \hat{\theta}_t(M) + \frac{1}{\alpha_t} Z_t - \varepsilon \\
&\geq \max_{M \in \mathcal{M} \setminus \{\widehat{M}^*\}} \hat{\theta}_t(M) + \max_{M \in \mathcal{M}} C_t \|\chi_M\|_{A_{\mathbf{x}_t}^{-1}} - \varepsilon \\
&\geq \hat{\theta}_t(M^*) + C_t \|\chi_{M^*}\|_{A_{\mathbf{x}_t}^{-1}} - \varepsilon \\
&\geq \theta(M^*) - \varepsilon.
\end{aligned}$$

The first inequality is from the event \mathcal{E}' and the second one is from the stopping condition. The third inequality holds as $Z_t \geq \alpha_t \max_{M \in \mathcal{M}} C_t \|\chi_M\|_{A_{\mathbf{x}_t}^{-1}}$. The last

inequality is again from the event \mathcal{E}' . \square

We see that approximation ratio of CEM $\alpha_\tau = \Omega\left(k^{-1/2}d^{-1/8}\sqrt{\frac{\xi_{\min}(\Lambda_\lambda)}{\xi_{\max}(\Lambda_\lambda)}}\right)$ for sufficiently large τ from Theorem 3.1 and Lemma 3.8 below, if we use the best approximation algorithm for the DkS as the DkS-Oracle [Bhaskara et al., 2010].

Lemma 3.8. *For $t \gg |\text{supp}(\lambda)|$, the condition number of $A_{\mathbf{x}_t}$ is given by*

$$\frac{\xi_{\max}(A_{\mathbf{x}_t})}{\xi_{\min}(A_{\mathbf{x}_t})} \simeq \frac{\xi_{\max}(\Lambda_\lambda)}{\xi_{\min}(\Lambda_\lambda)}.$$

Proof. Recall that we assume an arm selection strategy to be

$$M_t = \underset{M \in \text{supp}(\lambda)}{\text{argmin}} T_M(t-1)/\lambda_M,$$

which makes $T_M(t-1)$ to be proportional to the fixed arm selection ratio λ_M . It is easy to see that $\lambda_M t + 1 \geq T_M(t) \geq \lambda_M t - |\text{supp}(\lambda)|$ when such an arm selection strategy is employed. If $T_M(t) \simeq \lambda_M t$, then

$$\frac{\xi_{\max}(A_{\mathbf{x}_t})}{\xi_{\min}(A_{\mathbf{x}_t})} \simeq \frac{\xi_{\max}(t\Lambda_\lambda)}{\xi_{\min}(t\Lambda_\lambda)} = \frac{\xi_{\max}(\Lambda_\lambda)}{\xi_{\min}(\Lambda_\lambda)},$$

where $\Lambda_\lambda = \sum_{M \in \text{supp}(\lambda)} \lambda_M \mathbf{x}_M \mathbf{x}_M^\top$. \square

We are now ready to prove Theorem 3.3.

Proof of Theorem 3.3. We define event \mathcal{E}' as $\bigcap_{t=1}^{\infty} \mathcal{E}'_t$. We can see that the probability that event \mathcal{E}' occurs is at least $1 - \delta$ from Proposition 3.1. In the rest of the proof, we shall assume that this event holds. By Lemma 3.7 and the assumption on \mathcal{E}' , we see that the output \widehat{M}^* is ε -optimal super arm. Next, we focus on bounding the sample complexity.

A sufficient condition for stopping is that for M^* and for $t > d$,

$$\widehat{\theta}(M^*) - C_t \|\mathbf{x}_{M^*}\|_{A_{\mathbf{x}_t}^{-1}} \geq \max_{M \in \mathcal{M} \setminus \{M^*\}} \widehat{\theta}(M) + \frac{1}{\alpha_t} Z_t - \varepsilon. \quad (3.10)$$

From the definition of $A_{\mathbf{x}_t}^{-1}$, we have $\Lambda_\lambda = \frac{A_{\mathbf{x}_t}}{t}$. Using

$$\|\mathbf{x}_{M^*}\|_{A_{\mathbf{x}_t}^{-1}} \leq \max_{M \in \mathcal{M}} \|\mathbf{x}_M\|_{A_{\mathbf{x}_t}^{-1}}$$

and $Z_t \leq \max_{M \in \mathcal{M}} \|\mathbf{x}_M\|_{A_{\mathbf{x}_t}^{-1}}$, a sufficient condition for (3.10) is equivalent to:

$$\widehat{\Delta}(M^*, \overline{M}) \geq \left(1 + \frac{1}{\alpha_t}\right) C_t \sqrt{\frac{\rho(\lambda)}{t}} - \varepsilon, \quad (3.11)$$

where $\overline{M} = \operatorname{argmax}_{M \in \mathcal{M}} \widehat{\Delta}_t(M^*, M)$. On the other hand, we have

$$\|\chi_{M^*} - \chi_{\overline{M}}\|_{A_{\mathbf{x}_t}^{-1}} \leq 2 \max_{M \in \mathcal{M}} \|\chi_M\|_{A_{\mathbf{x}_t}^{-1}} \leq 2\sqrt{\frac{\rho(\lambda)}{t}}.$$

Therefore, from Proposition 1 with $\mathbf{x} = \chi_{M^*} - \chi_{M_i}$, with at least probability $1 - \delta$, we have

$$\widehat{\Delta}_t(M^*, \overline{M}) \geq \Delta(M^*, \overline{M}) - C_t \|\chi_{M^*} - \chi_{\overline{M}}\|_{A_{\mathbf{x}_t}^{-1}} \geq \Delta(M^*, \overline{M}) - 2C_t \sqrt{\frac{\rho(\lambda)}{t}}. \quad (3.12)$$

Combining (3.11) and (3.12), we see that a sufficient condition for stopping becomes the following inequality.

$$\Delta_{\min} - 2C_t \sqrt{\frac{\rho(\lambda)}{t}} \geq \left(1 + \frac{1}{\alpha_t}\right) C_t \sqrt{\frac{\rho(\lambda)}{t}} - \varepsilon.$$

Therefore, we have that a sufficient condition to stop is $t \geq \left(3 + \frac{1}{\alpha_t}\right)^2 C_t^2 H_\varepsilon$, where $H_\varepsilon = \frac{\rho(\lambda)}{(\Delta_{\min} + \varepsilon)^2}$. Let $\tau > d$ be the stopping time of the algorithm. From the above discussion, we see that

$$\tau \leq \left(3 + \frac{1}{\alpha_\tau}\right)^2 C_\tau^2 H_\varepsilon. \quad (3.13)$$

Recalling that $C_\tau = c\sqrt{\log(c'\tau^2 K/\delta)}$, we have that

$$\tau \leq (3 + 1/\alpha_\tau)^2 C_\tau^2 H_\varepsilon = (3 + 1/\alpha_\tau)^2 c^2 \log(c'\tau^2 K/\delta) H_\varepsilon.$$

Let τ' be a parameter that satisfies

$$\tau = (3 + 1/\alpha_\tau)^2 c^2 \log(c'\tau'^2 K/\delta) H_\varepsilon. \quad (3.14)$$

Then, it is obvious that $\tau' \leq \tau$ holds. For N defined as

$$N = (3 + 1/\alpha_\tau)^2 c^2 \log(c'K/\delta) H_\varepsilon,$$

we have

$$\tau' \leq \tau = (3 + 1/\alpha_\tau)^2 c^2 \log(\tau') H_\varepsilon + N \leq (3 + 1/\alpha_\tau)^2 c^2 \sqrt{\tau'} H_\varepsilon + N.$$

By solving this inequality with $c = 2\sqrt{2}\sigma$, we obtain

$$\sqrt{\tau'} \leq 4(3 + 1/\alpha_\tau)^2 \sigma^2 H_\varepsilon + \sqrt{16(3 + 1/\alpha_\tau)^4 \sigma^4 H_\varepsilon^2 + N}$$

$$\leq 2\sqrt{16(3 + 1/\alpha_\tau)^4 \sigma^4 H_\varepsilon^2 + N}.$$

Let $L = 2\sqrt{16(3 + 1/\alpha_\tau)^4 \sigma^4 H_\varepsilon^2 + N}$, which is equal to the RHS of the inequality. We see that $\log L = O\left(\log\left(\left(\frac{\sigma}{\alpha_\tau}\right)^2 H_\varepsilon + \log\left(\frac{K}{\delta}\right)\right)\right)$. Then, using this upper bound of τ' in (3.14), we have

$$\tau \leq 8 \left(3 + \frac{1}{\alpha_\tau}\right)^2 \sigma^2 H_\varepsilon \log\left(\frac{c'K}{\delta}\right) + C(H_\varepsilon, \delta),$$

where

$$\begin{aligned} C(H_\varepsilon, \delta) &= 16 \left(3 + \frac{1}{\alpha_\tau}\right)^2 \sigma^2 H_\varepsilon \log(L) \\ &= O\left(\sigma^2 H_\varepsilon \log\left(\frac{\sigma^2}{\alpha_\tau^2} H_\varepsilon + \log\left(\frac{K}{\delta}\right)\right)\right). \end{aligned}$$

□

3.11 Conclusion

We have studied the top- k arm identification with full-bandit feedback, where we cannot observe a reward of each base arm, but only the sum of the rewards. Although we can regard our problem as a special case of pure exploration in linear bandits, an approach based on linear bandits is not computationally feasible since the number of super arms may be exponential. To overcome the computational challenges, we have designed a novel approximation algorithm with theoretical guarantee for a 0-1 quadratic programming problem arising in confidence ellipsoid maximization. Based on our approximation algorithm, we have proposed (ϵ, δ) -PAC algorithms **SAQM** that run in $O(\log K)$ time and provided an upper bound of the sample complexity, which is still worst-case optimal; the result indicates that our algorithm provided an exponential speedup over exhaustive search algorithm while keeping the statistical efficiency. We also designed two heuristic algorithms that empirically perform well: **SA-FOA** using first-order approximation and **CLUCEB-QM** based on lower-upper confidence bound algorithm. Finally, we have conducted experiments on synthetic and real-world datasets with more than 10^{10} super arms, demonstrating the superiority of our algorithms in terms of both the computation time and the sample complexity.

Chapter 4

Combinatorial Pure Exploration with Full-bandit Feedback under General Constraints

In the previous chapter, we focused on the setting where underlying combinatorial structure is a size- k subset. How can we tackle combinatorial pure exploration under general constraints with full-bandit feedback? How can we design adaptive algorithms with a problem-dependent optimal sample complexity? In this chapter, to answer these questions, we study the problem of combinatorial pure exploration with full-bandit feedback (CPE-BL), where underlying combinatorial structures include paths, matchings, and matroids. For CPE-BL, we design the first *polynomial-time adaptive* algorithm, whose sample complexity matches the lower bound (within a logarithmic factor) for a family of instances and has a mild dependence on the minimal gap.

4.1 Problem Statement

In CPE-BL, an agent is given d base arms numbered $1, 2, \dots, d$. Although we used $\mathcal{M} \subseteq 2^{[d]}$ to denote a set of super arms in the previous chapter, following the notation in linear bandits, we use $\mathcal{X} \subseteq \{0, 1\}^d$ to denote a set of super arms which satisfy a certain combinatorial structure such as size- k subsets, matroids, paths and matchings. Let m denote the maximum number of base arms that a super arm in \mathcal{X} contains, i.e. $m = \max_{x \in \mathcal{X}} \|x\|_1$ ($m \leq d$). There is an unknown environment vector $\theta \in \mathbb{R}^d$ with $\|\theta\|_2 \leq L$ for some known constant $L > 0$. At each time step t , an agent pulls a super arm x_t and receives a random reward (full-bandit feedback) $y_t = x_t^\top(\theta + \eta_t)$, where η_t is a zero-mean noise vector bounded in $[-1, 1]^d$ and it is independent among different time step t . Let $x^* = \operatorname{argmax}_{x \in \mathcal{X}} x^\top \theta$ denote the optimal super arm, and we assume that the optimal x^* is unique as previous pure exploration works [Chen et al., 2014, Lin et al., 2014, Fiez et al., 2019] do. Let Δ_i denote the gap of the expected rewards between x^* and the super arm with the i -th largest expected reward.

Given a confidence level $\delta \in (0, 1)$, the objective is to use as few samples as possible to identify the optimal super arm with probability at least $1 - \delta$. The learning performance is evaluated by the number of samples required by the agent, called the *sample complexity*.

For clarity, we recall that some notation. For a given family \mathcal{X} , we use $\Delta(\mathcal{X})$ to denote the set of probability distributions over \mathcal{X} . For distribution $\lambda \in \Delta(\mathcal{X})$, we define $\text{supp}(\lambda) = \{x : \lambda(x) > 0\}$, $M(\lambda) = \mathbb{E}_{z \sim \lambda}[zz^\top]$ and $\widetilde{M}(\lambda) = \sum_{x \in \text{supp}(\lambda)} xx^\top$.

4.2 Related Work and Chapter Contribution

For CPE-BL, ICB proposed in Chapter 3 is a polynomial-time but static algorithm, which has a heavy dependence on Δ_{\min} in the sample complexity and empirically requires a large number of samples for instances with small Δ_{\min} . Rejwan and Mansour [2020] develop a polynomial-time adaptive algorithm **CSAR** but it only works for the top- k case, which has a naive reduction to previous CPE-MB. For BAI-LB where the action space is often considered small, Tao et al. [2018] propose an adaptive algorithm **ALBA** with a light Δ_{\min} dependence. Fiez et al. [2019] present the first lower bound and a nearly optimal algorithm **RAGE**. Recently, Katz-Samuels et al. [2020] also design an improved nearly optimal algorithm **Peace**, which is built upon **RAGE**. Degenne et al. [2020] develop asymptotically optimal algorithms **LinGame** and **LinGame-C**. While the existing BAI-LB algorithms achieve satisfactory sample complexity, none of them can solve CPE-BL in polynomial time in d [Soare et al., 2014, Karnin, 2016, Xu et al., 2018, Tao et al., 2018, Fiez et al., 2019, Degenne et al., 2020, Katz-Samuels et al., 2020].

Table 4.1: Comparison between our results and existing results for CPE-BL and BAI-LB. “General” represents that the algorithm works for combinatorial structures including size- k subsets, paths, matchings, and matroids. $\tilde{O}(\cdot)$ only omits $\log \log$ factors. Main notation is defined in Section 4.1, and other specific notation are given in the footnote.

Algorithm	Sample complexity ¹	Case	Problem Type	Strategy	Time
PolyALBA in This Chapter	$\tilde{O}\left(\sum_{i=2}^{\lfloor \frac{d}{2} \rfloor} \frac{1}{\Delta_i^2} \log \frac{ \mathcal{X} }{\delta} + \frac{d^2 m_{\xi_{\max}}(\tilde{M}(\lambda))^{-1}}{\Delta_{d+1}^2} \log \frac{ \mathcal{X} }{\delta}\right)$	General	CPE-BL	Adaptive	Poly(d)
ICB in Chapter 3	$\tilde{O}\left(\frac{d \xi_{\max}(M(\lambda))^{-1} \rho(\lambda)}{\Delta_{\min}^2} \log \frac{d \xi_{\max}(M(\lambda))^{-1} \rho(\lambda)}{\Delta_{\min}^2 \delta}\right)$	General ²	CPE-BL	Static	Poly(d)
SAQM in Chapter 3	$\tilde{O}\left(\frac{d^{1/4} k \xi_{\max}(M(\lambda))^{-1} \rho(\lambda)}{\Delta_{\min}^2} \log \frac{d^{1/4} k \xi_{\max}(M(\lambda))^{-1} \rho(\lambda)}{\Delta_{\min}^2 \delta}\right)$	Top- k	CPE-BL	Static	Poly(d)
CSAR [Rejwan and Mansour, 2020]	$\tilde{O}\left(\sum_{i=2}^d \frac{1}{\Delta_i^2} \log \frac{d}{\delta}\right)$	Top- k	CPE-BL	Adaptive	Poly(d)
\mathcal{XY} -static [Soare et al., 2014]	$O\left(\frac{d}{\Delta_{\min}^2} \log \frac{ \mathcal{X} }{\delta \Delta_{\min}^2} + d^2\right)$	$\mathcal{X} \subseteq \mathbb{R}^d$	BAI-LB	Static	$\Omega(\mathcal{X})$
Explore-Verify [Karnin, 2016]	$O\left(\frac{d}{\Delta_{\min}^2} \log \frac{ \mathcal{X} }{\delta \Delta_{\min}} + d \log \delta^{-1}\right)$	$\mathcal{X} \subseteq \mathbb{R}^d$	BAI-LB	Static	$\Omega(\mathcal{X})$
LinGapE [Xu et al., 2018]	$\tilde{O}\left(d \sum_{x \in \mathcal{X}} H_x \log \frac{d \mathcal{X} }{\delta} \cdot \sum_{x \in \mathcal{X}} H_x\right)$	$\mathcal{X} \subseteq \mathbb{R}^d$	BAI-LB	Adaptive	$\Omega(\mathcal{X})$
\mathcal{Y} -ElimTil [Tao et al., 2018]	$\tilde{O}\left(\frac{d}{\Delta_{\min}^2} (\log \delta^{-1} + \log \mathcal{X})\right)$	$\mathcal{X} \subseteq \mathbb{R}^d$	BAI-LB	Adaptive	$\Omega(\mathcal{X})$
ALBA [Tao et al., 2018]	$\tilde{O}\left(\sum_{i=2}^d \frac{1}{\Delta_i^2} (\log \delta^{-1} + \log \mathcal{X})\right)$	$\mathcal{X} \subseteq \mathbb{R}^d$	BAI-LB	Adaptive	$\Omega(\mathcal{X})$
RAGE [Fiez et al., 2019]	$O\left(\sum_{t=1}^{\lfloor \log_2(4/\Delta_{\min}) \rfloor} 2(2^t)^2 \bar{\rho}(\mathcal{Y}(\mathcal{S}_t)) \log(t^2 \mathcal{X} ^2 / \delta)\right)$	$\mathcal{X} \subseteq \mathbb{R}^d$	BAI-LB	Adaptive	$\Omega(\mathcal{X})$
LinGame(-C) [Degenne et al., 2020]	$\limsup_{\delta \rightarrow 0} \frac{\mathbb{E}_{\theta}[\tau_{\delta}]}{\log(1/\delta)} \leq \min_{\lambda \in \Delta(\mathcal{X})} \max_{x \in \mathcal{X} \setminus \{x^*\}} \frac{2\ x^* - x\ _{M(\lambda)}^2}{((x^* - x)^{\top} \theta)^2}$	$\mathcal{X} \subseteq \mathbb{R}^d$	BAI-LB	Adaptive	$\Omega(\mathcal{X})$
Peace [Katz-Samuels et al., 2020]	$O\left(\left(\min_{\lambda \in \Delta(\mathcal{X})} \max_{x \in \mathcal{X} \setminus \{x^*\}} \frac{\ x^* - x\ _{M(\lambda)}^2}{((x^* - x)^{\top} \theta)^2} + \gamma^*\right) \log(1/\delta)\right)$	$\mathcal{X} \subseteq \mathbb{R}^d$	BAI-LB	Adaptive	$\Omega(\mathcal{X})$
Lower Bound [Fiez et al., 2019]	$\mathbb{E}_{\theta}[\tau_{\delta}] \geq \min_{\lambda \in \Delta(\mathcal{X})} \max_{x \in \mathcal{X} \setminus \{x^*\}} \frac{\ x^* - x\ _{M(\lambda)}^2}{((x^* - x)^{\top} \theta)^2} \log(1/2.4\delta)$	$\mathcal{X} \subseteq \mathbb{R}^d$	BAI-LB	-	-

Algorithm 4.1: Combinatorial lower-upper naive confidence bound (CLUNCB)

Input : Accuracy $\epsilon > 0$, confidence level $\delta \in (0, 1)$

1 **Initialization** For each $e \in [d]$, pull $x_e \in \mathcal{X}$ such that $e \in x_e$ once.
Initialize $A_{\mathbf{x}_t}^\omega$ and b_t ;

2 **while** $\tilde{\theta}_t^\top \tilde{x}_t - \tilde{\theta}_t^\top \hat{x}_t^* \leq \epsilon$ *is not true* **do**

3 $t \leftarrow t + 1$;

4 $\hat{x}_t^* \leftarrow \operatorname{argmax}_{x \in \mathcal{X}} \hat{\theta}_t^\top x$;

5 Set $\operatorname{rad}_t(e) = C_t \sqrt{(A_{\mathbf{x}_t}^\omega)^{-1}(e, e)}$ for all $e \in [d]$;

6 **for** $e = 1, \dots, d$ **do**

7 **if** $e \in \hat{x}_t^*$ **then** $\tilde{\theta}_t(e) \leftarrow \hat{\theta}_t(e) - \operatorname{rad}_t(e)$;

8 **else** $\tilde{\theta}_t(e) \leftarrow \hat{\theta}_t(e) + \operatorname{rad}_t(e)$;

9 $\tilde{x}_t \leftarrow \operatorname{argmax}_{x \in \mathcal{X}} \tilde{\theta}_t^\top x$;

10 $p_t \leftarrow \operatorname{argmax}_{e \in (\tilde{x}_t \setminus \hat{x}_t^*) \cup (\hat{x}_t^* \setminus \tilde{x}_t)} \operatorname{rad}_t(e)$;

11 Sample any $x_t \in \mathcal{X}$ such that $p_t \in x_t$;

12 Update $A_{\mathbf{x}_t}^\omega$, b_t and $\hat{\theta}_t$;

13 **Return** \hat{x}_t^*

In this chapter, we provide a general framework for solving CPE-BL. Our proposed algorithm simultaneously achieves the following properties: (a) polynomial-time complexity, (b) adaptive sampling, such that the sample complexity is not heavily dependent on Δ_{\min} ; (c) general combinatorial constraints, and (d) nearly optimal sample complexity for some family of instances. We empirically compare our algorithms with several state-of-the-art CPE-BL and BAI-LB algorithms. Our result demonstrates that (a) our algorithm runs much faster than all the others, some of which cannot even finish after days of running; (b) our algorithm is much more robust against varying Δ_{\min} than the existing nonadaptive algorithm. Comparison with the related results is given Table 4.1.

4.3 Naive Reduction of the Top- k Case and Analysis of a UCB-based Algorithm for CPE-BL

In this section, we briefly explain a naive reduction to the classic CPE-MB for CPE-BL in the top- k setting, and discuss that there is no simple reduction for general CPE-BL by showing an undesirable property of a UCB-based algorithm

¹Notation appearing in the table but not relevant in our problem setting are given below: $\rho(\lambda) = \max_{x \in \mathcal{X}} \|x\|_{M(\lambda)^{-1}}^2$. $\tilde{\Delta}_i = \theta_i - \theta_{k+1}$ if $i \leq k$ and $\theta_k - \theta_i$ otherwise. $H_x = \max_{x_i, x_j \in \mathcal{X}} \frac{\bar{\rho}_x(x_i, x_j)}{\max\{\tilde{\Delta}_i^2, \tilde{\Delta}_j^2\}}$ where $\tilde{\Delta} = (x^* - x_i)^\top \theta$ if $x_i \neq x^*$, $\operatorname{argmin}_{x \in \mathcal{X}} x^* - x$ otherwise, and $\bar{\rho}_x(x_i, x_j)$ is a term defined by the optimal solution to a convex optimization (see (11) in Xu et al. [2018]). $S_t = \{x \in \mathcal{X} : (x^* - x)^\top \theta \leq 4 \cdot 2^{-t}\}$. $\mathcal{Y}(S_t) = \{x - x' : \forall x, x' \in S_t, x \neq x'\}$. $\tilde{\rho}(\mathcal{Y}(S_t)) = \min_{\lambda \in \Delta(\mathcal{X})} \max_{v \in \mathcal{Y}(S_t)} \|v\|_{M(\lambda)^{-1}}$. $\gamma^* = \min_{\lambda \in \Delta(\mathcal{X})} \mathbb{E}_{\eta \sim N(0,1)} \left[\max_{x \in \mathcal{X} \setminus \{x^*\}} \frac{(x^* - x)^\top M(\lambda)^{-1/2} \eta}{(x^* - x)^\top \theta} \right]^2$.

²ICB runs in polynomial time for combinatorial constraints such that P_1 in (3.7) is solved in polynomial time.

with a regularized least-square estimator.

As discussed in the previous chapter, with only $O(k)$ more samples, the problem of top- k identification with full-bandit feedback can be solved by classic top- k algorithms in which base arms are always queried. Suppose that an algorithm has a sample complexity of $C_{\delta,\Delta}$ in classic setting, it yields a complexity of $\tilde{O}(k \cdot C_{\delta,\Delta})$ for the full-bandit setting, where \tilde{O} omits some log factors. This is due to the fact that the unbiased estimate can be obtained for the difference between the two base arms by comparing two k -base arm queries with one base arm difference. Formally, with any fixed base arm i_0 , one can get an unbiased estimate for the gap $\theta_j - \theta_{i_0}$ with $O(k)$ times larger variance by querying two super-arms $S \cup \{j\}$ and $S \cup \{i_0\}$ for $S \subseteq [d]$ such that $|S| = k - 1$ and $j, i_0 \notin S$. Therefore, when $C_{\delta,\Delta}$ has dependence of $\sum_{i \in [d]} \Delta_i^{-2}$, we also have a sample complexity which has dependence of $\sum_{i \in [d]} \Delta_i^{-2}$ for top- k case with full-bandit feedback. However, for more complex cases such as matroid, matroid intersection, and s - t path, we cannot use such a reduction due to its combinatorial constraint.

We show that with a simple modification using the regularized least-square estimator, CLUCB algorithm proposed in Chen et al. [2014] can work for CPE-BL for general constraints such as size- k subsets, matroid, matching, and s - t path, and it is a polynomial-time (ϵ, δ) -PAC algorithm. However, we prove that this naive adaption can be sub-optimal and the sample complexity depends on $\frac{1}{\Delta_{\min}^2}$ in the worst case.

4.3.1 Regularized Least Square Estimator

We call an algorithm *fully adaptive* if it changes the arm-selection strategy based on the past observation at all rounds. For such an adaptive algorithm, we cannot use the ordinary least-square estimator for $\theta \in \mathbb{R}^d$ as an unbiased estimator. Instead we will use the regularized least-square estimator. If the sequence of super-arm selections $\mathbf{x}_t = (x_1, \dots, x_t)$ is adaptively determined based on the past observations, the regularized least-square estimator is given by

$$\hat{\theta}_t = (A_{\mathbf{x}_t}^\omega)^{-1} b_{\mathbf{x}_t}, \quad (4.1)$$

where $A_{\mathbf{x}_t}^\omega$ and $b_{\mathbf{x}_t}$ is defined by

$$A_{\mathbf{x}_t}^\omega = \omega I + \sum_{i=1}^t x_i x_i^\top, \quad \text{and} \quad b_{\mathbf{x}_t} = \sum_{i=1}^t x_i r_i \in \mathbb{R}^n.$$

for regularization parameter $\omega > 0$ and the identity matrix I . If we set $\omega = 0$ and we are allowed to sample a base arm, i.e., unit vector at all rounds, it is easy to see that $A_{\mathbf{x}_t}^\omega(i, i) = T_i(t)$ for $i \in [d]$ and $A_{\mathbf{x}_t}^\omega(i, j) = 0$ for $i \neq j$, where $T_i(t)$ is the number of times that base arm i is sampled before round $t + 1$. Abbasi-Yadkori et al. [2011] showed the high probability bound for the regularized least-squares estimator $\hat{\theta}$ (see Proposition 3.2 in Chapter 3).

We also introduce the notion of the *width* for a decision set \mathcal{X} defined in Chen et al. [2014]; $\text{width}(\mathcal{X})$ prescribes the size of the thinnest exchange class. For example, if \mathcal{X} are independent sets of ground set $[d]$, $\text{width}(\mathcal{X}) \leq 2$.

4.3.2 Analysis of CLUCB for CPE-BL

In the setting where a base arm is always pulled at all rounds, the confidence radius is simply defined as $\text{rad}_t(e) = \sqrt{\frac{2 \log\left(\frac{4dt^3}{\delta}\right)}{T_e(t)}}$ for all $e \in [d]$. Since we are not allowed to pull each base arm in CPE-BL, we can not define such a radius as the above form. However, we have concentration inequalities for each unit vector of e , and thus we can construct the confidence radius in the full-bandit setting. From Proposition 3.2, we can construct the high probability confidence radius as follows.

Lemma 4.1. *Suppose that a reward from each base arm i for all $i \in [d]$ follows 1-sub-Gaussian distribution. For all $t > 0$ and all $i \in [d]$, the confidence radius $\text{rad}_t(i)$ is defined as*

$$\text{rad}_t(i) = C_t \sqrt{(A_{\mathbf{x}_t}^\omega)^{-1}(i, i)} \quad (\forall i \in [d]), \quad (4.2)$$

where C_t is given by

$$C_t \leq \kappa \sqrt{d \log \frac{1 + tm/\omega}{\delta}} + \omega^{\frac{1}{2}} L. \quad (4.3)$$

Let \mathbf{rad}_t be an d -dimensional vector with nonnegative entries. For \mathbf{rad}_t , define random event \mathcal{E}_t for all $t > 0$ as follows.

$$\mathcal{E}_t = \{\forall i \in [d], |\theta(i) - \hat{\theta}_t(i)| \leq \text{rad}_t(i)\} \quad (4.4)$$

Then we have

$$\Pr \left[\bigcap_{t=1}^{\infty} \mathcal{E}_t \right] \geq 1 - \delta. \quad (4.5)$$

The proof is omitted since it is straightforward from Proposition 3.2 and union bounds. Using the above confidence radius, we can design CLUCB-based algorithm for CPE-BL, which is detailed in Algorithm 4.1. We show that Algorithm 4.1 is (ϵ, δ) -PAC and its sample complexity bound is given in Corollary 4.1.

Corollary 4.1. *Let $\lambda_C \in \Delta(\mathcal{X})$ be a distribution in which $\lambda_C(x)$ represents the ratio that x is pulled by CLUCB. The total number of samples T is bounded as*

$$T = O \left(d\kappa^2 \tilde{H} \log \left(\frac{d\kappa^2 m \tilde{H} / \omega + \log \delta^{-1}}{\delta} \right) \right),$$

where \tilde{H} is defined as

$$\tilde{H} = \max_{e \in [d]} \left(M(\lambda_C)^{-1}(e, e) \min \left\{ \frac{9 \text{width}(\mathcal{X})^2}{\Delta_e^2}, \frac{4m^2}{\epsilon d^2} \right\} \right).$$

Algorithm 4.2: ALBA(S, δ) [Tao et al., 2018]

Input : Action set S and confidence δ .

```
1 Initialize  $S_1 \leftarrow S$ ;  
2 for  $q \leftarrow 1, \dots, \lfloor \log_2 d \rfloor$  do  
3    $\delta_q \leftarrow \frac{6}{\pi^2} \frac{\delta}{(q+1)^2}$ ;  
4    $S_{q+1} \leftarrow \text{ElimTil}_{\lfloor \frac{d}{2^q} \rfloor}(S_q, \delta_q)$ ;  
5    $q \leftarrow q + 1$ ;
```

Output: $x \in S_{q+1}$

As can be seen, the sample complexity depends on Δ_{\min}^{-2} in the worst case. Also, since CLUNCB is fully adaptive, we cannot completely control λ_C beforehand and thus $M(\lambda_C)^{-1}(e, e) \leq \lambda_{\max}(M(\lambda_C)^{-1})$ can be large. Therefore, we may need different approaches which does not rely on the confidence radius given by Lemma 4.1 to design adaptive algorithms.

4.4 Adaptive Algorithm for CPE-BL

In this section, we propose the first polynomial-time adaptive algorithm, namely PolyALBA, for CPE-BL, and show that its sample complexity matches the lower bound (within a logarithmic factor) for a family of instances.

ALBA algorithm [Tao et al., 2018]. Before stating the main algorithm, we introduce the Adaptive Linear Best Arm (ALBA) algorithm for BAI-LB [Tao et al., 2018] (see Algorithm 4.2 for its description), which is the key subroutine of the proposed method PolyALBA. First, we describe the randomized least-square estimator defined by Tao et al. [2018]. Let y_1, \dots, y_n be n *i.i.d.* samples following a given distribution $\lambda \in \Delta(\mathcal{X})$, and let the corresponding rewards be r_1, \dots, r_n respectively. Let $b = \sum_{i=1}^n r_i y_i$. Then, the randomized estimator $\hat{\theta}$ is given by $\hat{\theta} = A^{-1}b$, where $A = nM(\lambda) \in \mathbb{R}^{d \times d}$ (recall that $M(\lambda) = \mathbb{E}_{z \sim \lambda}[zz^\top]$). The procedure for computing the estimate for θ is described in **VectorEst** (Algorithm 4.5). ALBA is an elimination-based algorithm, where in round q it identifies the top $d/2^q$ arms and discards the remaining arms by means of **ElimTil_p** (Algorithm 4.3). Note that ALBA(S, δ) runs in time polynomial to $|S|$. Since in CPE-BL, $|\mathcal{X}|$ is exponential to the instance size, it is infeasible to run ALBA with $S = \mathcal{X}$. Our main contribution is the nontrivial construction of a polynomial sized S_1 to run ALBA with.

For completeness, we provide the algorithm for computing the optimal distribution $\lambda_{\mathcal{X}_\sigma}^* = \min_{\lambda \in \Delta(\mathcal{X}_\sigma)} \max_{x \in \mathcal{X}_\sigma} x^\top M(\lambda)^{-1}x$ for a given set of actions $\mathcal{X}_\sigma \subseteq \mathcal{X}$, which is used in both ALBA [Tao et al., 2018] and our method PolyALBA. Algorithm 4.4 details the entropic mirror descent algorithm proposed in Tao et al. [2018].

Main algorithm. Now we present the proposed algorithm PolyALBA (see Algorithm 4.6 for its description), in which ALBA is invoked with $S = S_1$ with $|S_1| = d$. Set S_1 is constructed by a novel preparation procedure in the first epoch ($q = 0$). In this preparation epoch, we first compute a fixed distribution

Algorithm 4.3: ElimTil_p(S, δ)

Input : A parameter p , arms set S and confidence level δ .

- 1 Compute $\lambda_S^* \leftarrow \min_{\lambda \in \Delta(S)} \max_{x \in S} x^\top M(\lambda)^{-1} x$ by Algorithm 4.4;
- 2 Initialize $S_1 \leftarrow S$, $r \leftarrow 1$;
- 3 **while** $|S_r| > p$ **do**
- 4 Set $\varepsilon_r \leftarrow 1/2^r$, $\delta_r \leftarrow 6/\pi^2 \cdot \delta/r^2$;
- 5 $\hat{\theta}_r \leftarrow \text{VectorEst}(\lambda_S^*, c_0 \frac{2+(6+\varepsilon_r/2)d}{(\varepsilon/2)^2} \ln \frac{5|S|}{\delta_r})$;
- 6 $x_r \leftarrow \operatorname{argmax}_{x \in S_r} x^\top \hat{\theta}_r$;
- 7 $S_{r+1} \leftarrow S_r \setminus \{x \in S_r \mid x^\top \hat{\theta}_r < x_r^\top \hat{\theta}_r - \varepsilon_r\}$;
- 8 $r \leftarrow r + 1$;

Output: S_r

Algorithm 4.4: The entropic mirror descent for computing $\lambda_{\mathcal{X}_\sigma}^*$

Input : d -set of base arms $[d]$, a set of super arms $\mathcal{X}_\sigma \subseteq \mathcal{X}$, Lipschitz constant L_f of function $\log \det M(\lambda)$ and tolerance ϵ

- 1 Initialize $t \leftarrow 1$ and $\lambda^{(1)} \leftarrow (1/|\mathcal{X}_\sigma|, \dots, 1/|\mathcal{X}_\sigma|)$;
- 2 **while** $|\max_{x \in \mathcal{X}_\sigma} x^\top M(\lambda^{(t)})^{-1} x| - d \geq \epsilon$ **do**
- 3 $a_t \leftarrow \frac{\sqrt{2 \ln |\mathcal{X}_\sigma|}}{L_f \sqrt{t}}$;
- 4 Compute gradient $G_i^{(t)} \leftarrow \operatorname{Tr}(M(\lambda^{(t)})^{-1} (x_i x_i^\top))$;
- 5 Update $\lambda_i^{(t+1)} \leftarrow \frac{\lambda_i^{(t)} \exp(a_t G_i^{(t)})}{\sum_{i=1}^{|\mathcal{X}_\sigma|} \lambda_i^{(t)} \exp(a_t G_i^{(t)})}$;
- 6 $t \leftarrow t + 1$;
- 7 $\lambda_{\mathcal{X}_\sigma}^* \leftarrow \lambda^{(t)}$;

Output: $\lambda_{\mathcal{X}_\sigma}^*$

$\lambda \in \Delta(\mathcal{X})$ that has a polynomial-size support and a key parameter α (line 1). Then, based on λ we apply static estimation to estimate θ , until we see a big enough gap between the empirically best and the $(d+1)$ -th best actions (lines 4–13). The empirical top- d actions, excluding those that also have big gaps to the best one, form the set S_1 (lines 10–11), which is used to call **ALBA** to obtain the final result \hat{x}^* .

Note that, computing the empirical best $d+1$ super arms can be done in polynomial time by using *Lawler's k -best procedure* [Lawler, 1972]. This procedure only requires the existence of the efficient maximization oracle, which is satisfied in many combinatorial problems such as maximum matching, shortest paths and minimum spanning tree.

We remark that the computational efficiency of **PolyALBA** is not merely owing to the Lawler's k -best procedure. In fact, even if previous BAI-LB algorithms apply the same procedure, they cannot run in polynomial time since they explicitly maintain exponential-sized action set and sample on distributions with exponential supports. These render heavy computation and memory in every round of previous algorithms. In contrast, we avoid the naive enumeration and sampling on the combinatorial space directly, and instead find empirical top- d actions as representatives through a novel polynomial-time computation procedure.

Algorithm 4.5: VectorEst(λ, n)

Input : distribution λ and the number of samples n

- 1 Let y_1, \dots, y_n be the n samples acquired from $\text{supp}(\lambda)$ according to the distribution λ ;
- 2 Pull arms y_1, \dots, y_n ;
- 3 Observe the rewards r_1, \dots, r_n ;
- 4 $A \leftarrow n \cdot \sum_{x \in \text{supp}(\lambda)} \lambda(x) x x^\top$;
- 5 $b \leftarrow \sum_{i=1}^n r_i y_i$;

Output: The estimate $\hat{\theta} \leftarrow A^{-1}b$

Algorithm 4.6: PolyALBA

Input : confidence level δ , a set of base arms $[d]$, and

$$c_0 = \max\{4L^2, 3\}.$$

- 1 Set $q \leftarrow 0$ and $\delta_q \leftarrow \frac{6}{\pi^2} \frac{\delta}{(q+1)^2}$;
- 2 Compute a distribution $\lambda \leftarrow \lambda_{\mathcal{X}_\sigma}^*$ and parameter
 $\alpha \leftarrow \sqrt{md/\xi_{\min}(\widetilde{M}(\lambda_{\mathcal{X}_\sigma}^*))}$ by Algorithm 4.7;
- 3 $r \leftarrow 1$;
- 4 **while** true **do**
 - 5 Set $\varepsilon_r \leftarrow \frac{1}{2^r}$ and $\delta_r \leftarrow \frac{6}{\pi^2} \frac{\delta_q}{r^2}$;
 - 6 $\ell(\varepsilon) \leftarrow \frac{2m+2\alpha\sqrt{md}+4\alpha^2d+\alpha\varepsilon d}{\varepsilon^2}$;
 - 7 $\hat{\theta}_r \leftarrow \text{VectorEst}(\lambda, c_0 \ell(\frac{\varepsilon_r}{2}) \ln(\frac{5|\mathcal{X}|}{\delta_r}))$;
 - 8 Select $d+1$ actions $\hat{x}_1, \dots, \hat{x}_d, \hat{x}_{d+1}$ with the highest $d+1$ empirical means $x^\top \hat{\theta}_r$ in all $x \in \mathcal{X}$;
 - 9 **if** $\hat{x}_1^\top \hat{\theta}_r - \hat{x}_{d+1}^\top \hat{\theta}_r > \varepsilon_r$ **then**
 - 10 $B_1 \leftarrow \{\hat{x}_1, \dots, \hat{x}_d\}$;
 - 11 $S_1 \leftarrow B_1 \setminus \{x \in B_1 \mid \hat{x}_1^\top \hat{\theta}_r - x^\top \hat{\theta}_r > \varepsilon_r\}$;
 - 12 **break**;
 - 13 $r \leftarrow r + 1$;
- 14 $\hat{x}^* \leftarrow \text{output by ALBA}(S_1, \delta_1)$

Output: \hat{x}^*

Algorithm 4.7: Computing a distribution λ

Input : d -base arms

- 1 Choose any d super arms $\mathcal{X}_\sigma = \{b_1, \dots, b_d\}$ from \mathcal{X} , such that $\text{rank}(X) = d$ where $X = (b_1, \dots, b_d)$;
- 2 $\lambda_{\mathcal{X}_\sigma}^* \leftarrow \arg\min_{\lambda \in \Delta(\mathcal{X}_\sigma)} \max_{x \in \mathcal{X}_\sigma} x^\top M(\lambda)^{-1} x$;
- 3 $\alpha \leftarrow \sqrt{md/\xi_{\min}(\widetilde{M}(\lambda_{\mathcal{X}_\sigma}^*))}$;

Output: $\lambda_{\mathcal{X}_\sigma}^*$ and α

4.5 Theoretical Analysis of PolyALBA

Now we provide the sample complexity bound of PolyALBA and show that our sample complexity matches the information theoretical lower bound up to loga-

rithmic factors.

4.5.1 Analysis of the Statistical and Computational Efficiency

Theorem 4.1. *With probability at least $1 - \delta$, the PolyALBA algorithm (Algorithm 4.6) returns the best super arm x^* with sample complexity*

$$O\left(\sum_{i=2}^{\lfloor \frac{d}{2} \rfloor} \frac{c_0}{\Delta_i^2} (\ln \delta^{-1} + \ln |\mathcal{X}| + \ln \ln \Delta_i^{-1}) + \frac{c_0 d (\alpha \sqrt{m} + \alpha^2)}{\Delta_{d+1}^2} (\ln \delta^{-1} + \ln |\mathcal{X}| + \ln \ln \Delta_{d+1}^{-1})\right),$$

where $\alpha = \sqrt{md/\xi_{\min}(\widetilde{M}(\lambda_{\mathcal{X}_\sigma}^*))}$.

The first term in Theorem 4.1 is for the remaining epochs required by subroutine ALBA and the second term is for the preparation procedure. As shown in Theorem 4.1, our sample complexity bound has lighter dependence on Δ_{\min} , compared with the existing results. Now we explain the key role for achieving the polynomial-time complexity of PolyALBA in the first epoch played by the distribution $\lambda_{\mathcal{X}_\sigma}^*$ and parameter α . Notice that even if we employ a uniform distribution on a polynomial-size support $\mathcal{X}_\sigma \subseteq \mathcal{X}$, i.e., $\lambda_{\mathcal{X}_\sigma} = (1/|\mathcal{X}_\sigma|)_{x \in \mathcal{X}_\sigma}$, computing the maximal confidence bound $\max_{x \in \mathcal{X}} \|x\|_{M(\lambda_{\mathcal{X}_\sigma})^{-1}}$ is NP-hard, while many (UCB-based) algorithms in LB ignore this issue and simply use a brute force method. In contrast, PolyALBA utilizes G-optimal design [Pukelsheim, 2006] and runs in polynomial time while guaranteeing the optimality. In the following lemma, we show that $\alpha\sqrt{d}$ gives the upper bound on the maximal ellipsoidal norm associated to $M(\lambda_{\mathcal{X}_\sigma}^*)^{-1}$.

Lemma 4.2. *For $\lambda_{\mathcal{X}_\sigma}^*$ and α obtained by Algorithm 4.7, it holds that*

$$\max_{x \in \mathcal{X}} \|x\|_{M(\lambda_{\mathcal{X}_\sigma}^*)^{-1}} \leq \alpha\sqrt{d},$$

where $\alpha = \sqrt{md/\xi_{\min}(\widetilde{M}(\lambda_{\mathcal{X}_\sigma}^*))}$.

In order to explain that this lemma gives approximation guarantee of Algorithm 4.7 for $\min_{\lambda \in \Delta(\mathcal{X})} \max_{x \in \mathcal{X}} \|x\|_{M(\lambda)^{-1}}$, we introduce the following equivalence theorem in Kiefer and Wolfowitz [1960] adopted to our setting of CPE-BL.

Proposition 4.1 (Kiefer and Wolfowitz [1960]). *Define $M(\lambda) = \mathbb{E}_{z \sim \lambda}[zz^\top]$ for any distribution λ supported on $\mathcal{X} \subseteq \mathbb{R}^d$. We consider two extremum problems.*

The first is to choose λ so that

$$(1) \lambda \text{ maximizes } \det M(\lambda) \quad (D\text{-optimal design})$$

The second one is to choose λ so that

$$(2) \lambda \text{ minimizes } \max_{x \in \mathcal{X}} x^\top M(\lambda)^{-1} x \quad (G\text{-optimal design})$$

We note that $\mathbb{E}_{x \sim \lambda}[x^\top M(\lambda)^{-1} x]$ is d , hence, $\max_{x \in \mathcal{X}} x^\top M(\lambda)^{-1} x \geq d$, and thus a sufficient condition for λ to satisfy (2) is

$$(3) \max_{x \in \mathcal{X}} x^\top M(\lambda)^{-1} x = d.$$

Statements (1), (2) and (3) are equivalent.

From the equivalence theorem for optimal experimental designs in Proposition 4.1, it holds that $\min_{\lambda \in \Delta(\mathcal{X})} \max_{x \in \mathcal{X}} \|x\|_{M(\lambda)^{-1}} = \sqrt{d}$. From this fact and Lemma 4.2, we can see that $\lambda_{\mathcal{X}_\sigma}^*$ obtained by Algorithm 4.7 is α (≥ 1)-approximate solution to $\min_{\lambda \in \Delta(\mathcal{X})} \max_{x \in \mathcal{X}} \|x\|_{M(\lambda)^{-1}}$ where \mathcal{X} can be defined by general combinatorial constraints. Note that α can be easily obtained by computing $\xi_{\min}(\widehat{M}(\lambda_{\mathcal{X}_\sigma}^*))$ (recall that $\widehat{M}(\lambda) = \sum_{x \in \text{supp}(\lambda)} x x^\top$). Therefore, by employing $\lambda_{\mathcal{X}_\sigma}^*$ and a prior knowledge of its approximation ratio α , we can guarantee that the preparation sampling scheme identifies a set S_1 containing the optimal super arm x^* with high probability. In the remaining epochs, PolyALBA can successfully focus on sampling near-optimal super arms by ALBA owing to the optimality of S_1 . Note that $\alpha = 1$ if we compute $\min_{\lambda \in \Delta(\mathcal{X})} \max_{x \in \mathcal{X}} \|x\|_{M(\lambda)^{-1}}$ exactly. If we approximately solve it, α is independent on the arm-selection ratio but it can depend on the support of λ . For further discussion on improving α will be provided in Section 4.6.

4.5.2 Discussion on the Optimality

For BAI-LB, Fiez et al. [2019] propose the first sample complexity lower bound

$$\mathbb{E}_\theta[\tau_\delta] \geq \min_{\lambda \in \Delta(\mathcal{X})} \max_{x \in \mathcal{X} \setminus \{x^*\}} \frac{\|x^* - x\|_{M(\lambda)^{-1}}^2}{((x^* - x)^\top \theta)^2} \log(1/2.4\delta),$$

and a nearly optimal algorithm RAGE with sample complexity

$$O \left(\sum_{t=1}^{\lfloor \log_2(4/\Delta_{\min}) \rfloor} 2(2^t)^2 \tilde{\rho}(\mathcal{Y}(S_t)) \log(t^2 |\mathcal{X}|^2 / \delta) \right),$$

where $S_t = \{x \in \mathcal{X} : (x^* - x)^\top \theta \leq 4 \cdot 2^{-t}\}$, $\mathcal{Y}(S_t) = \{x - x' : \forall x, x' \in S_t, x \neq x'\}$ and $\tilde{\rho}(\mathcal{Y}(S_t)) = \min_{\lambda \in \Delta(\mathcal{X})} \max_{v \in \mathcal{Y}(S_t)} \|v\|_{M(\lambda)^{-1}}$. Fiez et al. [2019] prove that this upper bound for RAGE matches the lower bound with a logarithmic factor.

Below we show that for a family of instances, the sample complexity of PolyALBA matches that of RAGE and also achieves near-optimality. Consider a family of instances such that $\Delta_{\lfloor d/2^{(t-2)} \rfloor + 1} = 4 \cdot 2^{-t}$, $t = 2, 3, \dots, \log_2(\frac{4}{\Delta_{\min}})$, which can be easily found in practical sub-problems. For example, in the *Multi-Bandit* case [Gabillon et al., 2011a], we are given base arms numbered by $1, \dots, 10$, and a feasible super arm consists of two base arms respectively from $\{1, \dots, 5\}$ and

$\{6, \dots, 7\}$, where $d = 10$ and $|\mathcal{X}| = 25$. Suppose that

$$\theta = (2.5, 2, 1.5, 1, 0.5, 0.625, 0.5, 0.375, 0.25, 0.125)^\top.$$

We use x_i and A_i to denote the indicator vector and the set of base arm indices for the i -th best super arm, respectively. Then, $A_1 = \{1, 6\}$, $x_1^\top \theta = 3.125$, $A_2 = \{1, 7\}$, $x_2^\top \theta = 3$, $A_3 = \{1, 8\}$, $x_3^\top \theta = 2.875$, $A_6 = \{2, 6\}$, $x_6^\top \theta = 2.625$, $A_{11} = \{3, 6\}$, $x_{11}^\top \theta = 2.125$, which belongs to the considered family of instances.

In such family of instances, our **PolyALBA** performs similar elimination procedure as **RAGE**. Specifically, since the sample complexity of **PolyALBA** guarantees that for epoch q , any action x with $\Delta_x \geq \Delta_{\lfloor d/2^q \rfloor + 1}$ will be discarded, we have that $\forall x' \in S_q, \Delta_{x'} \leq \Delta_{\lfloor d/2^{q-1} \rfloor + 1} = 2 \cdot 2^{-q}$, $q = 1, 2, \dots, \log_2(\frac{4}{\Delta_{\min}})$, which is the same to the gap constraint for S_t in **RAGE**. Then, in epoch $q = 2, \dots, \log_2(\frac{4}{\Delta_{\min}})$, **PolyALBA** essentially performs the same procedure as **RAGE**, while in epoch $q = 1$, **PolyALBA** requires a larger number of samples (the $\xi_{\max}(\tilde{M}^{-1}(\lambda))$ -related term) than **RAGE**. Formally, in such instances **PolyALBA** has sample complexity

$$O\left(\sum_{t=2}^{\lfloor \log_2(4/\Delta_{\min}) \rfloor} 2(2^t)^2 \tilde{\rho}(\mathcal{Y}(S_t)) \log(t^2 |\mathcal{X}|^2 / \delta) + md \xi_{\max}(\tilde{M}^{-1}(\lambda)) \tilde{\rho}(\mathcal{Y}(S_1)) \log(|\mathcal{X}|^2 / \delta)\right).$$

For sufficiently small Δ_{\min} (increase d in the above Multi-Bandit case to obtain small Δ_{\min}), the additional term related to $\xi_{\max}(\tilde{M}^{-1}(\lambda))$ is absorbed and the result is the same to that of **RAGE**, which matches the lower bound within a logarithmic factor. The sample complexity of **PolyALBA** shows superiority over other heavily Δ_{\min} -dependent algorithms [Soare et al., 2014, Karnin, 2016, Kuroki et al., 2020], and the additional term related to $\xi_{\max}(\tilde{M}^{-1}(\lambda))$ can be viewed as a cost for achieving computational efficiency.

We also remark that **PolyALBA** is close to the lower bound in the worst-case instances and our sample complexity has an interesting relation with that of G-allocation strategy for BAI-LB [Soare et al., 2014]. Soare et al. [2014] proved the following lemma to bound the *oracle complexity* H_{LB} defined as follows, which also appears in the information theoretic lower bound [Fiez et al., 2019].

$$H_{\text{LB}} = \min_{\lambda \in \Delta(\mathcal{X})} \max_{x \in \mathcal{X} \setminus \{x^*\}} \frac{\|x^* - x\|_{M(\lambda)^{-1}}^2}{((x^* - x)^\top \theta)^2}.$$

Lemma 4.3 (Lemma 2, Soare et al. [2014]). *Given an arm set $\mathcal{X} \subseteq \mathbb{R}^d$ and parameter θ , the complexity H_{LB} is such that*

$$\max_{x \in \mathcal{X} \setminus \{x^*\}} \frac{\|x^* - x\|_2^2}{L_x \Delta_{\min}^2} \leq H_{\text{LB}} \leq \frac{4d}{\Delta_{\min}^2},$$

where L_x is the upper bound of the ℓ_2 -norm of any $x \in \mathcal{X}$. Furthermore, if \mathcal{X} is the canonical basis, the problem reduces to a MAB and $\sum_{i=1}^{|\mathcal{X}|} 1/\Delta_i \leq H_{\text{LB}} \leq 2 \sum_{i=1}^{|\mathcal{X}|} 1/\Delta_i$.

Algorithm 4.8: Approximation algorithm for G-optimal design by the ellipsoid method

- Input** : d -set of base arms $[d]$, $n \in \mathbb{Z}_+$, $\tilde{n} \in \mathbb{Z}_+$.
- 1 **for** $i = 1, \dots, n$ **do**
 - 2 Choose $\mathcal{X}_{\sigma_i} \leftarrow$ any \tilde{n} -super arms;
 - 3 Compute $\lambda_i \leftarrow \lambda_{\mathcal{X}_{\sigma_i}}^*$ by Algorithm 4.4 for \mathcal{X}_{σ_i} ;
 - 4 Compute $w_{\lambda_i} \in \mathbb{R}_+^d$ by setting $w_{\lambda_i} = (\sum_{j \in [d]} |M(\lambda_i)_{e,j}^{-1/2}|)_{e \in [d]} \in \mathbb{R}_+^d$;
 - 5 Perform the ellipsoid method to solve $\text{LP}_{\text{primal}}$:

$$\text{LP}_{\text{primal}} : \min. \nu \tag{4.6}$$

$$\begin{aligned} \text{s.t. } \quad & \nu \geq \sum_{i \in [n]} h_i \sum_{e \in [d]} w_{\lambda_i, e} x_e, \quad (\forall x \in \mathcal{X}) \\ & h \in \Delta([n]). \end{aligned}$$

- 6 $h^* \leftarrow$ optimal solution to $\text{LP}_{\text{primal}}$;
 - 7 $\nu^* \leftarrow$ optimal value of $\text{LP}_{\text{primal}}$
 - 8 Sample $i^* \in [n]$ from $h^* \in \Delta([n])$;
 - 9 $\alpha \leftarrow \min \left\{ \frac{\nu^*}{\sqrt{d}}, \min_{i \in [n]} \sqrt{\frac{md}{\xi_{\min}(\tilde{M}(\lambda_i))}} \right\}$;
- Output:** λ_{i^*} and α
-

Soare et al. [2014] designed the G-allocation strategy and its sample complexity is $O\left(\frac{4d}{\Delta_{\min}^2} \log \frac{|\mathcal{X}|}{\delta}\right)$, which shows that G-allocation strategy is optimal within logarithmic terms for instances where the worst-case value of H_{LB} is given (See Theorem 1 in their paper). For instances with $\Delta_i = \Delta_{\min}, \forall i > 1$, PolyALBA has the sample complexity of $\tilde{O}\left(\frac{md^2 \xi_{\max}(\tilde{M}^{-1}(\lambda))}{\Delta_{\min}^2} \log \frac{|\mathcal{X}|}{\delta}\right)$, which matches that of G-allocation strategy up to a factor of $md \xi_{\max}(\tilde{M}^{-1}(\lambda))$ (when we ignore the log log terms). PolyALBA also employs G-optimal design for its static phase but we restrict its support in order to make it running in polynomial time, which causes the additional term related to $\xi_{\max}(\tilde{M}^{-1}(\lambda))$. Note that the lower bound [Fiez et al., 2019] does not consider time complexity, and the gap in the additional term may be a price paid for achieving computational efficiency.

To sum up, to our best knowledge, PolyALBA is the first polynomial-time adaptive algorithm that works for CPE-BL with general combinatorial structures and achieves nearly optimal sample complexity for a family of problem instances.

4.6 Discussion on Improving α

In this section, we further discuss the approximation algorithm for computing $\lambda \in \Delta(\mathcal{X})$ and provide Algorithm 4.8, which can be one alternative of Algorithm 4.7 in PolyALBA. Lemma 4.2 indicates that choosing $\text{supp}(\lambda)$ that maximizes $\xi_{\min}(\tilde{M}(\lambda))$ gives the better bound. Such a design is so called the *E-optimal design*, i.e., the goal is to minimize the maximum eigenvalue of the error covariance [Pukelsheim, 2006]. If we are allowed to pull unit vectors, we have $\xi_{\min}(\tilde{M}(\lambda)) \geq 1$. For general cases, more sophisticated algorithm by the ellip-

soid method [Grötschel et al., 1981] is considered in this section as one alternative of Algorithm 4.7. This approach provides the better choice of λ (or equivalently α) than Algorithm 4.7 in terms of the expectation value.

Let $\Delta(\mathcal{X})_{\text{poly}}$ be the subset of probability distributions over \mathcal{X} with polynomial sized support. Our task is to find an approximate solution $\tilde{\lambda} \in \Delta(\mathcal{X})_{\text{poly}}$ to the following minmax optimization:

$$\min_{\lambda \in \Delta(\mathcal{X})} \max_{x \in \mathcal{X}} \|x\|_{M(\lambda)^{-1}}.$$

We denote the exact G-optimal design by $\lambda^* = \operatorname{argmin}_{\lambda \in \Delta(\mathcal{X})} \max_{x \in \mathcal{X}} \|x\|_{M(\lambda)^{-1}}$. The above minmax optimization is computationally intractable in combinatorial settings, while many existing methods in linear bandits involved the brute force to solve G-optimal design problem [Fiez et al., 2019, Soare et al., 2014, Tao et al., 2018]. To avoid a intractable brute force, we address a relaxation problem for the minmax optimization, i.e., a randomized mixed strategy for the robust combinatorial optimization. However, since the ellipsoidal norm $\|x\|_{M(\lambda)^{-1}}$ has the quadratic form, such a relaxation problem is still hard to compute. To overcome this challenge, we consider a simpler norm instead of the ellipsoidal norm. In higher level, this idea is similar to that of Dani et al. [2008]; they use skewed octahedron called ConfidenceBall₁ as its confidence region rather than the ellipsoid. The radius of ConfidenceBall₁ has been set large enough such that it contains the confidence ellipsoid as an inscribed subset. Whereas they use 1-norm $\|M(\lambda)^{1/2}x\|_1$ to define ConfidenceBall₁, however, $\max_{x \in \mathcal{X}} \|M(\lambda)^{1/2}x\|_1$ is still intractable in the combinatorial action space. To avoid the computational hardness, we introduce a linear function $g_\lambda : \{0, 1\}^d \rightarrow \mathbb{R}_+$ in order to utilize the underlying combinatorial structure. We define $w_\lambda = (\sum_{j \in [d]} |M(\lambda)^{-1/2}|_{i,j})_{i \in [d]} \in \mathbb{R}_+^d$ for $\lambda \in \Delta(\mathcal{X})$. For $\lambda \in \Delta(\mathcal{X})$, a linear function $g_\lambda : \{0, 1\}^d \rightarrow \mathbb{R}_+$ is represented as $g_\lambda(x) = \sum_{e \in [d]} w_{\lambda,e} x_e$. We shall assume that the Ellipsoid method computes the optimal solution for linear programmings by enough iterations [Grötschel et al., 1981].

We show that the optimal value ν^* in Algorithm 4.8 gives an upper bound of the confidence ellipsoidal norm.

Lemma 4.4. *Let \mathcal{X} be a family of super arms satisfying given constraints such as the matroid, matroid intersection, and s-t path. Let $\lambda^{h^*} \in \Delta(\mathcal{X})_{\text{poly}}$ be an output by Algorithm 4.8. Let h^* be an optimal solution and ν^* be the optimal value for $\text{LP}_{\text{primal}}$. Then, λ^{h^*} satisfies*

$$\max_{x \in \mathcal{X}} \mathbb{E}[\|x\|_{M(\lambda^{h^*})^{-1}}] \leq \nu^*.$$

By using the above property, we have the expected sample complexity given in the following corollary.

Corollary 4.2. *Let ν^* be the optimal value for $\text{LP}_{\text{primal}}$ obtained in Algorithm 4.8. With probability at least $1 - \delta$, the PolyALBA algorithm (Algorithm*

4.6) will return the best super arm x^* with the expected sample complexity

$$O\left(\frac{c_0 d(\alpha\sqrt{m} + \alpha^2)}{\Delta_{d+1}^2} (\ln \delta^{-1} + \ln |\mathcal{X}| + \ln \ln \Delta_{d+1}^{-1}) + \sum_{i=2}^{\lfloor \frac{d}{2} \rfloor} \frac{c_0}{\Delta_i^2} (\ln \delta^{-1} + \ln |\mathcal{X}| + \ln \ln \Delta_i^{-1})\right),$$

where $\alpha = \min \left\{ \frac{\nu^*}{\sqrt{d}}, \min_{i \in [n]} \sqrt{\frac{md}{\xi_{\min}(\bar{M}(\lambda_i))}} \right\}$.

Note that Algorithm 4.8 runs in polynomial time as long as $g_\lambda(x)$ is a linear function and not necessarily $g_\lambda(x) = \sum_{e \in [d]} w_{\lambda,e} x_e$ defined in this section.

LP-based algorithm for combinatorial robust optimization. We briefly explain polynomial-time solvability of $\text{LP}_{\text{primal}}$ in Algorithm 4.8 by the Ellipsoid method. Given a family \mathcal{X} satisfying a combinatorial constraint and n -set function $g_{\lambda_1}, \dots, g_{\lambda_n} : 2^{[d]} \rightarrow \mathbb{R}_+$, we describe how to solve the following combinatorial robust optimization:

$$\min_{h \in \Delta([n])} \max_{x \in \mathcal{X}} \sum_{i \in [n]} h_i g_{\lambda_i}(x).$$

By von Neumann's minimax theorem, it holds that

$$\min_{h \in \Delta([n])} \max_{x \in \mathcal{X}} \sum_{i \in [n]} h_i g_{\lambda_i}(x) = \max_{p \in \Delta(\mathcal{X})} \min_{i \in [n]} \sum_{x \in \mathcal{X}} p_x g_{\lambda_i}(x).$$

A *polytope* of \mathcal{X} is defined as $P(\mathcal{X}) = \text{conv}\{x : x \in \mathcal{X}\}$. For a vector $x \in \mathcal{X}$ we define S^x to be the corresponding subset form, i.e. $S^x = \{i \in [n] : x_i = 1\}$. The key observation is that for any distribution $p \in \Delta(\mathcal{X})$, we can obtain a point $y \in P(\mathcal{X})$ by $y = p^\top \cdot x$, and thus for every $e \in [d]$, $y_e = \sum_{x \in \mathcal{X} : e \in S^x} p_x$. This means that y_e is the marginal probability of seeing an included dimension (a.k.a. a base arm e) when selecting vector x (a.k.a. super arm S^x) according to distribution p .

Then, the optimal value of the above problems is equal to the value of the following LP:

$$\begin{aligned} \text{LP}_{\text{dual}} : \max. \quad & s \\ \text{s.t.} \quad & s \leq \sum_{e \in [d]} w_{\lambda_i, e} y_e, \quad (\forall i \in [n]) \\ & y \in P(\mathcal{X}). \end{aligned} \tag{4.7}$$

If \mathcal{X} is a matroid, matroid intersection, or the set of s - t paths, there exists an efficient *separation oracle* for $P(\mathcal{X})$. The separation problem for these constraints can be solved in polynomial time as long as $g_{\lambda_1}, \dots, g_{\lambda_n}$ are linear functions. Therefore, due to the theorem of Grötschel et al. [1981], we can solve the LP in polynomial time in d and n . Note that the Ellipsoid method can find an

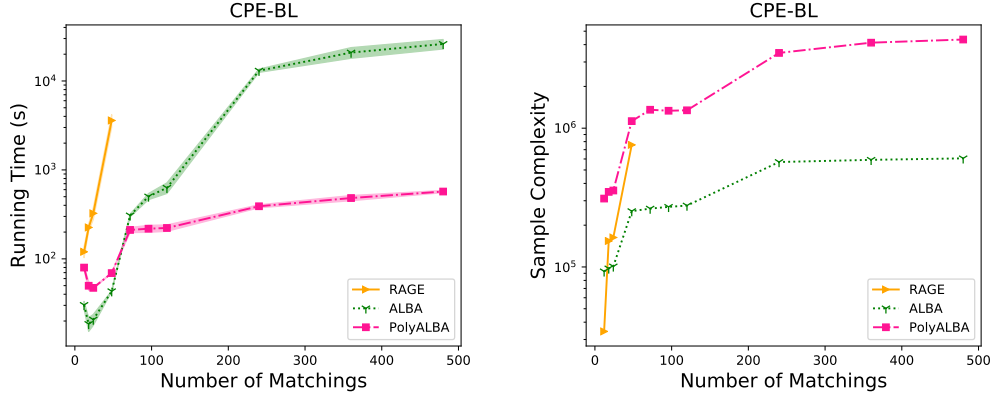


Figure 4.1: Experimental results of running time and sample complexity for CPE-BL.

optimal solution to the dual problem of LP_{dual} , i.e., LP_{primal} in (4.6). Therefore, we can obtain $h^* = \operatorname{argmin}_{h \in \Delta([n])} \max_{x \in \mathcal{X}} \sum_{i \in [n]} h_i g_{\lambda_i}(x)$. For the knapsack constraint and the $r(> 2)$ -matroid intersection constraint, the corresponding separation problems are NP-hard. Kawase and Sumita [2019] proposed approximation schemes by solving a separation problem for a relaxation of the polytope, which gives PTAS for the knapsack constraint and $2/(er)$ -approximate solution for r -matroid intersection constraint.

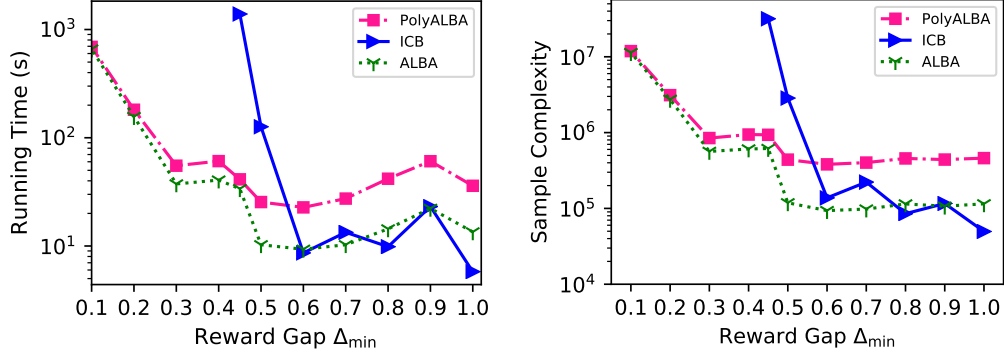
4.7 Experiments

We conduct experiments for CPE-BL on the matching instances, and compare our algorithms with the state-of-the-arts in terms of both running time and sample complexity. We also provide empirical results for the top- k case with discussion on Δ_{\min} -dependence. We conduct all the experiments on Intel Xeon E5-2640 v3 CPU at 2.60GHz with 132GB RAM.

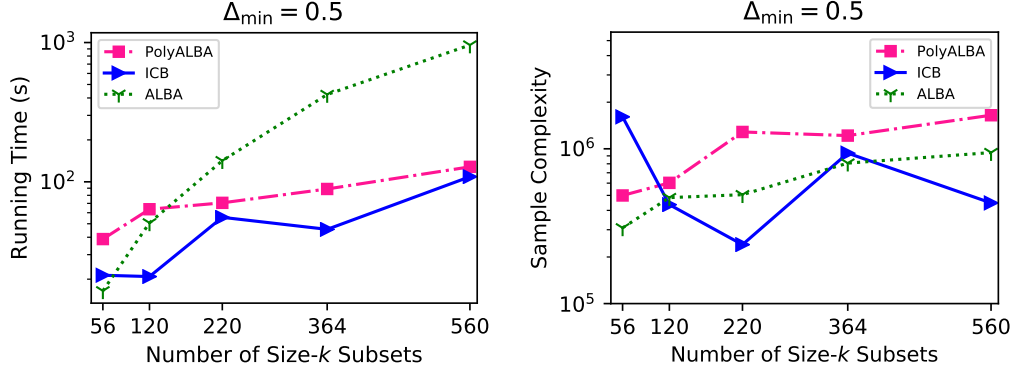
4.7.1 Experiments on the Matching Instances

Experimental settings. We compare our PolyALBA with BAI-LB algorithms, ALBA and RAGE, in terms of running time and sample complexity. We set action space \mathcal{X} as a set of matchings in 3-by-3, 4-by-4 and 5-by-5 complete bipartite graphs. The dimension d , i.e. the number of edges, is set from 9 to 25. The number of matchings $|\mathcal{X}|$ are set from 12 to 480. $\theta_1, \dots, \theta_d$ is set as a geometric sequence in $[0, 1]$. We simulate the random feedback for action x by a Gaussian distribution with mean of $x^\top \theta$ and unit variance. For each algorithm, we perform 20 independent runs and present the average running time and sample complexity with 95% confidence intervals across runs.

Results. In the experiments, RAGE [Fiez et al., 2019] reports memory errors when $|\mathcal{X}| > 48$ due to its heavy memory burden, and thus we only obtain its results on small- $|\mathcal{X}|$ instances. As shown in Figure 4.1(a) with a logarithmic y-axis, our PolyALBA runs about two orders of magnitude faster than ALBA and RAGE, and the running time of PolyALBA increases more slowly than the others



(a) The number of samples and running time with $(d, k) = (8, 3)$. We vary the minimum gap Δ_{\min} from 0.1 to 1.0. Each point is an average over 10 realizations.



(b) The number of samples and running time with $k = 3$, $d = 8, 10, 12, 14, 16$, and $\Delta_{\min} = 0.5$. Each point is an average over 10 realizations.

Figure 4.2: Experimental results of running time and sample complexity for full-bandit top- k instances.

as $|\mathcal{K}|$ increases. For sample complexity results, due to the extra preparation epoch, PolyALBA has a worse sample complexity. From the results, we see that ALBA and RAGE are too slow to run and PolyALBA is the only feasible algorithm, since in practice one has to keep the computation time low first to make an algorithm useful.

4.7.2 Experiments on the Top- k Instances

The main purpose of the experiments in this subsection is to see the dependence of the performance on the minimum gap Δ_{\min} . We empirically demonstrate that PolyALBA is robust across different settings with varying Δ_{\min} and runs much faster than existing BAI-LB algorithms as reported in Figure 4.2.

Experimental settings. As a polynomial-time baseline algorithm, we implement ICB presented in Chapter 3, whose sample complexity heavily depends on Δ_{\min} . We set the expected rewards for the top- k base arms uniformly at random from $[0.5, 1.0]$. Let $\theta_{\min-k}$ be the minimum expected reward in the top- k base arms. We set the expected reward of the top $(k + 1)$ -th base arm to $\theta_{\min-k} - \Delta_{\min}$ for the predetermined parameter $\Delta_{\min} \in [0.1, 1.0]$. Then, we generate the expected rewards of the rest of base arms by uniform samples from

$[-1.0, \theta_{\min-k} - \Delta_{\min}]$ so that expected rewards of the best super-arm is larger than those of the rest of super arms by at least Δ_{\min} . We set the additive noise distribution $\mathcal{N}(0, 1)$. In all instances we set $\delta = 0.05$. In order to perform the exponential-time algorithms **ALBA** in reasonable time, we run the experiments with $d = 8$, $k = 3$, and thus $|\mathcal{X}| = 56$. Note that since **RAGE** is already prohibitive in these instances due to its memory error, we exclude it here. The result is shown in Figure 4.2(a). In the second experiment, we evaluate **PolyALBA**, **ALBA**, and **ICB** on the synthetic instances with varying $|\mathcal{X}|$ and fixed Δ_{\min} . To perform **ICB** with a reasonable sample complexity, we set $\Delta_{\min} = 0.5$ in all instances. We vary the number of base arms $d = [8, 10, 12, 14, 16]$ while $k = 3$ is fixed. Thus, $|\mathcal{X}| \in [56, 120, 220, 364, 560]$ in this experiment. The result is shown in Figure 4.2(b).

Results. As can be seen in Figure 4.2(a), **PolyALBA** performs well in all instances, while a polynomial-time baseline **ICB** is sensitive to the value of Δ_{\min} , which matches our theoretical analysis. Indeed, **ICB** cannot stop even after several days because it requires at least more than 10^9 samples when $\Delta_{\min} = 0.4$. On the other hand, **PolyALBA** is still competitive with the exponential-time baseline **ALBA**. Figure 4.2(b) demonstrates that **ALBA** is too slow in larger sized instances; we report that **ALBA** cannot work in the large instance with $d = 18$ and $|\mathcal{X}| = 816$ due to its memory issue in our environment. From the results, **PolyALBA** is the only practical algorithm that is efficient in terms of both time complexity and sample complexity. We validate that the sample complexity of **PolyALBA** is robust against the minimum gap, which well suits the real-world applications.

4.8 Proofs of Theoretical Results

4.8.1 Proof of Corollary 4.1

Proof. First, we state the following two lemmas in Chen et al. [2014]; Lemma 4.5 (Lemma 12 in Chen et al. [2014]) shows that if the confidence radius is valid, then **CLUCB** always outputs ϵ -optimal set, and Lemma 4.6 (Lemma 13 in Chen et al. [2014]) implies that if the confidence radius of an arm is sufficiently small, then the arm will not be chosen as p_t . Note that we have the lemmas since Lemmas 3, 5, 7, and 10 in Chen et al. [2014] also hold for our setting where \mathbf{rad}_t is given by (4.2) and the empirical mean is replaced with the least square estimator $\hat{\theta}_t$ in (4.1).

Lemma 4.5 (Lemma 12 in Chen et al. [2014]). *If **CLUCB** stops on round t and suppose that \mathcal{E}_t occurs. Then, we have $\theta^\top x^* - \theta^\top x_{\text{Out}} \leq \epsilon$.*

Lemma 4.6 (Lemma 13 in Chen et al. [2014]). *Given any t and suppose that event \mathcal{E}_t occurs. For any $e \in [d]$, if $\text{rad}_t(e) < \max \left\{ \frac{\Delta_\epsilon}{3\text{width}(\mathcal{X})}, \frac{\epsilon}{2m} \right\}$, then $p_t \neq e$.*

The event $\bigcap_{t=1}^{\infty} \mathcal{E}_t$ occurs with probability at least $1 - \delta$ from Lemma 4.1. From Lemma 4.5, under the event $\bigcap_{t=1}^{\infty} \mathcal{E}_t$, **CLUNCB** returns an ϵ -optimal set.

Therefore, in the rest of part, we shall assume this event holds.

Fix any arm $e \in [d]$ and let τ_e be the last round which arm e is chosen as p_t . From (4.3) and for a small $\omega \leq \frac{\kappa^2}{L^2} \log \delta^{-1}$, we have

$$\begin{aligned} C_{\tau_e} &\leq \kappa \sqrt{n \log \frac{1 + \tau_e m / \omega}{\delta}} + \omega^{\frac{1}{2}} L \\ &\leq 2\kappa \sqrt{d \log \left(\frac{1 + \tau_e m / \omega}{\delta} \right)}. \end{aligned}$$

By Lemma 4.6, we have $\text{rad}_{\tau_e}(e) \geq \max \left\{ \frac{\Delta_e}{3\text{width}(\mathcal{X})}, \frac{\epsilon}{2m} \right\}$. We define $\Lambda_{\mathbf{x}_{\tau_e}} = \frac{A_{\mathbf{x}_t}^\lambda}{\tau_e}$. Then, we have

$$\begin{aligned} \max \left\{ \frac{\Delta_e^2}{9\text{width}(\mathcal{X})^2}, \frac{\epsilon^2}{4m^2} \right\} &\leq \text{rad}_t^2(e) = C_{\tau_e}^2 (A_{\mathbf{x}_{\tau_e}}^\lambda)^{-1}(e, e) \\ &\leq 4\kappa^2 d \log \left(\frac{1 + \tau_e m / \omega}{\delta} \right) (A_{\mathbf{x}_{\tau_e}}^\lambda)^{-1}(e, e) \\ &= 4\kappa^2 d \log \left(\frac{1 + \tau_e m / \omega}{\delta} \right) \frac{(\Lambda_{\mathbf{x}_{\tau_e}}^\iota)^{-1}(e, e)}{\tau_e}. \end{aligned}$$

That is, we obtain

$$\tau_e \leq H_e \log \left(\frac{1 + \tau_e m / \omega}{\delta} \right),$$

where we define $H_e = 4\kappa^2 d (\Lambda_{\mathbf{x}_{\tau_e}}^\iota)^{-1}(e, e) \min \left\{ \frac{9\text{width}(\mathcal{X})^2}{\Delta_e^2}, \frac{4m^2}{\epsilon^2} \right\}$. Let $\tau' (\leq \tau_e)$ satisfying

$$\tau_e = H_e \left(\log \left(1 + \frac{\tau' m}{\omega} \right) + \log \frac{1}{\delta} \right). \quad (4.1)$$

Then, we have

$$\begin{aligned} \tau' \leq \tau_e &= H_e \left(\log \left(1 + \frac{\tau' m}{\omega} \right) + \log \frac{1}{\delta} \right) \\ &\leq H_e \left(\sqrt{\frac{\tau' m}{\omega}} + \log \frac{1}{\delta} \right). \end{aligned} \quad (4.2)$$

Solving (4.2) for $\sqrt{\tau'}$, we obtain

$$\begin{aligned} \sqrt{\tau'} &\leq \frac{1}{2} \left(H_e \sqrt{\frac{m}{\omega}} + \sqrt{H_e^2 \frac{m}{\omega} + 4H_e \log \frac{1}{\delta}} \right) \\ &\leq 2\sqrt{H_e^2 \frac{m}{4\omega}} + H_e \log \frac{1}{\delta}. \end{aligned}$$

That is, we see that $\tau' = O\left(H_e^2 \frac{m}{\omega} + H_e \log \frac{1}{\delta}\right)$, which shows that

$$\log\left(\frac{1 + \tau' m}{\omega}\right) = O\left(\log\left(\frac{m H_e}{\omega} + \log \frac{1}{\delta}\right)\right). \quad (4.3)$$

Combining (4.3) into (4.1), we obtain

$$\tau_e = O\left(H_e \log\left(\frac{m H_e / \omega + \log \delta^{-1}}{\delta}\right)\right).$$

The number of samples used by CLUNCB is $T = \max_{e \in [d]} \tau_e$.

Recall that $\lambda_C \in \Delta(\mathcal{X})$ is a distribution in which $\lambda_C(x)$ represents the ratio that x was pulled by CLUNCB. Suppose that T is sufficiently large such that $\Lambda_{\mathbf{x}_T} = \frac{A_{\mathbf{x}_T}^\lambda}{T} \approx M(\lambda_C)$. Define

$$\tilde{H} = \max_{e \in [d]} \left((M(\lambda_C)^{-1}(e, e) \min \left\{ \frac{9 \text{width}(\mathcal{X})^2}{\Delta_e^2}, \frac{4m^2}{\epsilon d^2} \right\} \right)$$

Then, we have

$$T = \max_{e \in [d]} \tau_e = O\left(d \kappa^2 \tilde{H} \log\left(\frac{d \kappa^2 m \tilde{H} / \omega + \log \delta^{-1}}{\delta}\right)\right).$$

□

4.8.2 Proof of Lemma 4.2

Proof. Recall that in Algorithm 4.7, we choose d super arms $\mathcal{X}_\sigma = \{x_1, \dots, x_d\}$ from \mathcal{X} , such that $\text{rank}(X) = d$ where $X = (x_1, \dots, x_d)$. Then, for any super arm $z \in \mathcal{X}$, z can be written as a linear combination of x_1, x_2, \dots, x_d , i.e.,

$$z = Xw,$$

where $w \in \mathbb{R}^d$ is the vector of coefficients. Let $\xi_{\min}(A)$ denote the smallest eigenvalue of matrix A . Then, we have

$$\begin{aligned} \sum_{k=1}^d |w_k| &\leq \sqrt{d \left(\sum_{k=1}^d w_k^2 \right)} \\ &= \sqrt{dw^\top w} \\ &\leq \sqrt{dw^\top X^\top X w} \cdot \max_{w' \in \mathbb{R}^d} \sqrt{\frac{w'^\top w'}{w'^\top X^\top X w'}} \\ &\leq \sqrt{dw^\top X^\top X w} \cdot \sqrt{\frac{1}{\xi_{\min}(X^\top X)}} \end{aligned}$$

$$\begin{aligned}
&= \sqrt{\frac{d}{\xi_{\min}(X^\top X)}} \|z\|_2 \\
&\leq \sqrt{\frac{md}{\xi_{\min}(X^\top X)}} \\
&= \sqrt{\frac{md}{\xi_{\min}(XX^\top)}} \\
&= \alpha.
\end{aligned}$$

Note that in Algorithm 4.7, we compute

$$\lambda_{\mathcal{X}_\sigma}^* = \operatorname{argmin}_{\lambda \in \Delta(\mathcal{X}_\sigma)} \max_{x \in \mathcal{X}_\sigma} x^\top M(\lambda)^{-1} x$$

by the entropic mirror descent. $\lambda_{\mathcal{X}_\sigma}^*$ is the solution to Proposition 4.1 and satisfies $\max_{x \in \mathcal{X}_\sigma} x^\top M(\lambda_{\mathcal{X}_\sigma}^*)^{-1} x = d$. Thus, $\max_{x \in \mathcal{X}_\sigma} \|x\|_{M(\lambda_{\mathcal{X}_\sigma}^*)^{-1}} = \sqrt{d}$. Thus, for any $z \in \mathcal{X}$ we have

$$\begin{aligned}
\|z\|_{M(\lambda_{\mathcal{X}_\sigma}^*)^{-1}} &= \|w_1 x_1 + \cdots + w_d x_d\|_{M(\lambda_{\mathcal{X}_\sigma}^*)^{-1}} \\
&\leq |w_1| \cdot \|x_1\|_{M(\lambda_{\mathcal{X}_\sigma}^*)^{-1}} + \cdots + |w_d| \cdot \|x_d\|_{M(\lambda_{\mathcal{X}_\sigma}^*)^{-1}} \\
&\leq |w_1| \cdot \sqrt{d} + \cdots + |w_d| \cdot \sqrt{d} \\
&= (|w_1| + \cdots + |w_d|) \sqrt{d} \\
&\leq \alpha \sqrt{d}.
\end{aligned}$$

□

4.8.3 Technical lemmas for Theorem 4.1

We show two technical lemmas to prove Theorem 4.1.

Lemma 4.7. *When $n \geq c_0 \ell(\varepsilon) \ln(\frac{5|\mathcal{X}|}{\delta_r})$ where $\varepsilon \leq 3$, we have*

$$\Pr[|x^\top \theta - x^\top \hat{\theta}| \leq \varepsilon, \forall x \in \mathcal{X}] \geq 1 - \delta.$$

Proof. We introduce the high probability bound for the estimator $\hat{\theta}$ as follows.

Proposition 4.2 (Lemma 10 Tao et al. [2018]). *Let $c_0 = \max\{4L^2, 3\}$. Let $n \geq \ell \ln(5/8)$ where $\ell \geq d$. For any fixed $x \in \mathcal{X}$, with probability at least $1 - \delta$, we have*

$$|x^\top (\theta - \hat{\theta})| \leq \sqrt{\frac{2\|x\|_2^2 + 2\sqrt{d}\|x\|_2\|x\|_{M(\lambda)^{-1}} + (4 + 2\sqrt{d/\ell})\|x\|_{M(\lambda)^{-1}}^2}{\ell}}.$$

Since $\|x\|_2 \leq \sqrt{m}$ and $\|x\|_{M(\lambda)^{-1}} \leq \alpha\sqrt{d}$ from Lemma 4.2, applying Propo-

sition 4.2 for every super arm in \mathcal{X} and via a union bound, we have that when $n \geq c_0 \ell \ln(\frac{5|\mathcal{X}|}{\delta_r})$ where $\ell \geq d$,

$$\Pr \left[|x^\top \theta - x^\top \hat{\theta}| \leq \sqrt{\frac{2m + 2\alpha\sqrt{md} + (4 + 2\sqrt{d/\ell})\alpha^2 d}{\ell}}, \forall x \in \mathcal{X} \right] \geq 1 - \delta.$$

Setting ℓ as $\ell(\varepsilon) := \frac{2m + 2\alpha\sqrt{md} + 4\alpha^2 d + \alpha\varepsilon d}{\varepsilon^2}$, we have that with probability at least $1 - \delta$, $\forall x \in \mathcal{X}$,

$$\begin{aligned} |x^\top \theta - x^\top \hat{\theta}| &\leq \sqrt{\frac{2m + 2\alpha\sqrt{md} + (4 + 2\sqrt{d/\ell})\alpha^2 d}{\ell}} \\ &= \left(\frac{2m + 2\alpha\sqrt{md} + 4\alpha^2 d}{2m + 2\alpha\sqrt{md} + 4\alpha^2 d + \alpha\varepsilon d} \varepsilon^2 \right. \\ &\quad \left. + \frac{2\alpha^2 d^{\frac{3}{2}}}{(2m + 2\alpha\sqrt{md} + 4\alpha^2 d + \alpha\varepsilon d)^{\frac{3}{2}}} \varepsilon^3 \right)^{\frac{1}{2}} \\ &\leq \varepsilon \left(1 - \frac{\alpha\varepsilon d}{2m + 2\alpha\sqrt{md} + 4\alpha^2 d + \alpha\varepsilon d} \right. \\ &\quad \left. + \frac{2\alpha\sqrt{d}}{\sqrt{2m + 2\alpha\sqrt{md} + 4\alpha^2 d + \alpha\varepsilon d}} \right. \\ &\quad \left. \frac{\alpha\varepsilon d}{(2m + 2\alpha\sqrt{md} + 4\alpha^2 d + \alpha\varepsilon d)} \right)^{\frac{1}{2}} \\ &\leq \varepsilon, \end{aligned}$$

which completes the proof. \square

Next, we show the sample complexity bound for the epoch $q = 0$.

Lemma 4.8. *With probability at least $1 - \delta_0$, the first epoch $q = 0$ in Algorithm 4.6 satisfies the following properties: (i) epoch $q = 0$ ends with $x^* \in S_1$; and (ii) the sample complexity is bounded by*

$$O \left(\frac{c_0(\alpha\sqrt{md} + \alpha^2 d)}{\Delta_{d+1}^2} (\ln \delta^{-1} + \ln |\mathcal{X}| + \ln \ln \Delta_{d+1}^{-1}) \right).$$

Proof. For round r of the first epoch, define event $\mathcal{F}_r := \{|x^\top \theta - x^\top \hat{\theta}_r| \leq \frac{\varepsilon}{2}, \forall x \in \mathcal{X}\}$. Applying Lemma 4.7, we have $\Pr[\mathcal{F}_r] \leq 1 - \delta_r$. Define event $\mathcal{F} := \bigcap_{r=1}^{\infty} \mathcal{F}_r$. By a union bound, we have $\Pr[\mathcal{F}] \geq 1 - \sum_{r=1}^{\infty} \delta_r = 1 - \sum_{r=1}^{\infty} \frac{6}{\pi^2} \frac{\delta_q}{r^2} \geq 1 - \delta_q$. We condition the remaining proof on event \mathcal{F} .

(i) First, we show that the first epoch $q = 0$ will end with $x^* \in S_1$. Let x_1, x_2, \dots denote the super arms ranked by $x^\top \theta$ for $x \in \mathcal{X}$ (i.e., $x_1^\top \theta \geq x_2^\top \theta, \dots$), and we use x^* and x_1 interchangeably.

For any round $r \geq 1$, $x_1^\top \hat{\theta}_r \geq x_1^\top \theta - \frac{\varepsilon_r}{2} \geq \hat{x}_1^\top \theta - \frac{\varepsilon_r}{2} \geq \hat{x}_1^\top \hat{\theta}_r - \varepsilon_r$. Rearranging the terms, we have $\hat{x}_1^\top \hat{\theta}_r - x_1^\top \hat{\theta}_r \leq \varepsilon_r$, which implies that x_1 will never be discarded. Thus, when the first epoch $q = 0$ ends, $x_1 \in S_1$.

Let r^* be the smallest round such that $\varepsilon_{r^*} < \frac{\Delta_{d+1}}{2}$. In round r^* , for x_i s.t. $i \geq d+1$, $x_1^\top \hat{\theta}_{r^*} - x_i^\top \hat{\theta}_{r^*} \geq (x_1^\top \theta - \frac{\varepsilon_{r^*}}{2}) - (x_i^\top \theta + \frac{\varepsilon_{r^*}}{2}) \geq \Delta_i - \varepsilon_{r^*} \geq \Delta_{d+1} - \varepsilon_{r^*} > \varepsilon_{r^*}$. Then, the first epoch $q = 0$ will end.

(ii) Since the first epoch $q = 0$ will end in (or before) round r^* , which is the smallest round such that $\varepsilon_{r^*} < \frac{\Delta_{d+1}}{2}$, then the sample complexity of the first epoch $q = 0$ is bounded by

$$\begin{aligned} & O\left(\frac{c_0(\alpha\sqrt{md} + \alpha^2 d)}{\varepsilon_{r^*}^2} \ln\left(\frac{|\mathcal{X}|}{\delta_{r^*}}\right)\right) \\ &= O\left(\frac{c_0(\alpha\sqrt{md} + \alpha^2 d)}{\Delta_{d+1}^2} (\ln \delta^{-1} + \ln |\mathcal{X}| + \ln \ln \Delta_{d+1}^{-1})\right). \end{aligned}$$

□

4.8.4 Proof of Theorem 4.1

Proof. Define $q^* = \lfloor \log_2 d \rfloor$. For epoch $q = 0$, applying Lemma 4.8, the sample complexity is bounded by

$$O\left(\frac{c_0(\alpha\sqrt{md} + \alpha^2 d)}{\Delta_{d+1}^2} (\ln \delta^{-1} + \ln |\mathcal{X}| + \ln \ln \Delta_{d+1}^{-1})\right).$$

For epoch $q \geq 1$, the PolyALBA algorithm directly calls subroutine ALBA. Applying Lemma 17 in Tao et al. [2018], we can bound the sample complexity for epoch $q \geq 1$ by

$$\begin{aligned} & \sum_{q=1}^{q^*} O\left(\frac{c_0 \lfloor \frac{d}{2^{q-1}} \rfloor}{\Delta_{\lfloor \frac{d}{2^q} \rfloor + 1}^2} \left(\ln \delta^{-1} + \ln |\mathcal{X}| + \ln \ln \Delta_{\lfloor \frac{d}{2^q} \rfloor + 1}^{-1}\right)\right) \\ &= \sum_{q=1}^{q^*} O\left(\frac{c_0 \lfloor \frac{d}{2^q} \rfloor}{\Delta_{\lfloor \frac{d}{2^q} \rfloor + 1}^2} \left(\ln \delta^{-1} + \ln |\mathcal{X}| + \ln \ln \Delta_{\lfloor \frac{d}{2^q} \rfloor + 1}^{-1}\right)\right) \\ &= \sum_{q=1}^{q^*} O\left(\frac{c_0(\lfloor \frac{d}{2^q} \rfloor - \lfloor \frac{d}{2^{q+1}} \rfloor)}{\Delta_{\lfloor \frac{d}{2^q} \rfloor + 1}^2} \right. \\ & \quad \left. \left(\ln \delta^{-1} + \ln |\mathcal{X}| + \ln \ln \Delta_{\lfloor \frac{d}{2^q} \rfloor + 1}^{-1}\right)\right) \\ &= \sum_{q=1}^{q^*-1} \sum_{\lfloor \frac{d}{2^{q+1}} \rfloor + 1}^{\lfloor \frac{d}{2^q} \rfloor} O\left(\frac{c_0}{\Delta_i^2} (\ln \delta^{-1} + \ln |\mathcal{X}| + \ln \ln \Delta_i^{-1})\right) \end{aligned}$$

$$= O \left(\sum_{i=2}^{\lfloor \frac{d}{2} \rfloor} \frac{c_0}{\Delta_i^2} (\ln \delta^{-1} + \ln |\mathcal{X}| + \ln \ln \Delta_i^{-1}) \right)$$

Summing the sample complexity for epoch $q = 0$ and $q \geq 1$, we obtain the theorem. \square

4.8.5 Proof of Lemma 4.4

Proof. By the definition of $w_\lambda = (\sum_{j \in [d]} |M(\lambda)_{i,j}^{-1/2}|)_{i \in [d]}$ and definitions of the quadratic norm and 1-norm, we have

$$\begin{aligned} \|x\|_{M(\lambda)^{-1}} &= \|M(\lambda)^{-1/2}x\|_2 \\ &\leq \|M(\lambda)^{-1/2}x\|_1 \\ &\leq \sum_{e \in [d]} w_{\lambda,e} x_e \\ &= g_\lambda(x) \quad (\forall x \in \{0, 1\}^d). \end{aligned} \tag{4.4}$$

Let $y^* = \operatorname{argmax}_{x \in \mathcal{X}} \|x\|_{M(\lambda)^{-1}}$. From Eq. (4.4), it holds that

$$\max_{x \in \mathcal{X}} \|x\|_{M(\lambda)^{-1}} = \|y^*\|_{M(\lambda)^{-1}} \leq g_\lambda(y^*). \tag{4.5}$$

Recall that $\lambda^{h^*} = \lambda_{i^*}$ where i^* was sampled from $h^* \in \Delta([n])$ in Algorithm 4.8; we have $\mathbb{E}[w_{\lambda^{h^*},e}] = \sum_{i \in [n]} h_i^* w_{\lambda_i,e}$ for all $e \in [d]$. Thus, for any $x \in \mathcal{X}$, we see that

$$\mathbb{E}[g_{\lambda^{h^*}}(x)] = \sum_{i \in [n]} h_i^* g_{\lambda_i}(x).$$

From the above, we have that

$$\begin{aligned} \max_{x \in \mathcal{X}} \mathbb{E}[\|x\|_{M(\lambda^{h^*})^{-1}}] &\leq \max_{x \in \mathcal{X}} \mathbb{E}[g_{\lambda^{h^*}}(x)] \\ &= \max_{x \in \mathcal{X}} \sum_{i \in [n]} h_i^* g_{\lambda_i}(x). \end{aligned} \tag{4.6}$$

Thus, we obtain

$$\begin{aligned} \max_{x \in \mathcal{X}} \mathbb{E}[\|x\|_{M(\lambda^{h^*})^{-1}}] &\leq \max_{x \in \mathcal{X}} \sum_{i \in [n]} h_i^* g_{\lambda_i}(x) \\ &= \min_{h \in \Delta([n])} \max_{x \in \mathcal{X}} \sum_{i \in [n]} h_i g_{\lambda_i}(x) \\ &= \nu^* \end{aligned}$$

where the first inequality follows by Eq. (4.6) and the equations follow from the

fact that h^* is an optimal solution for $\text{LP}_{\text{primal}}$. □

4.9 Conclusion

In this chapter, we have studied the combinatorial pure exploration with full-bandit feedback (CPE-BL), where underlying combinatorial structures include size- k subsets, paths, matchings, and matroids. For CPE-BL, we have designed the first adaptive algorithm, **PolyALBA**, that runs in polynomial time. In **PolyALBA** we have two phases: the first phase is a static procedure for finding a polynomial-sized set of super arms which contains the optimal super arm with high probability and discards other super arms. The second phase focuses on sampling super arms among the remaining set by adaptive elimination procedures. We have shown that the sample complexity of **PolyALBA** achieves the worst-case optimality for a family of instances. We empirically perform our algorithms with several some of state-of-the-art CPE-BL and BAI-LB algorithms. Our result have demonstrated that (a) our algorithm runs much faster than all the others, some of which cannot even finish after days of running; (b) our algorithm is much more robust on settings where Δ_{\min} is varying than the static algorithm proposed in Chapter 3.

Chapter 5

Online Dense Subgraph Discovery via Linear Feedback

In the previous chapters, the reward function was simply assumed to be linear. However, in many combinatorial optimization problems, the objective (reward) function can be nonlinear and our proposed methods in previous chapters cannot be applied to such nonlinear settings. When the reward function is no longer linear in combinatorial pure exploration problems, how can we utilize the property of graph structures to resolve non-linearity? In this chapter, we focus on the specific graph optimization problem, called the densest subgraph problem whose objective (reward) function is defined as the degree density in the bandit setup. In this problem, an agent queries edge subsets rather than only single edges and observes a noisy sum of edge weights in a queried subset, and the goal is to identify the densest subgraph in a graph. For this problem, we propose a polynomial-time algorithm that obtains a nearly-optimal solution with high probability. Moreover, to deal with large-sized graphs, we design a more scalable algorithm with a theoretical guarantee. Computational experiments using real-world graphs demonstrate the effectiveness of our algorithms.

5.1 Introduction

We first introduce dense subgraph discovery and the densest subgraph problem. Then, we summarize our contribution, and provide a review of existing methods.

5.1.1 Dense Subgraph Discovery

Dense subgraph discovery (DSD) aims to find a dense component in weighted graphs. This is a fundamental graph-mining task with a variety of applications and thus has received much attention recently. Applications include detection of communities or span link farms in Web graphs [Dourisboure et al., 2007, Gibson et al., 2005], molecular complexes extraction in protein-protein interaction networks [Bader and Hogue, 2003], extracting experts in crowdsourcing systems [Kawase et al., 2019], and real-time story identification in micro-blogging streams [Angel et al., 2012].

Among a lot of optimization problems arising in dense subgraph discovery, the most popular one would be the *densest subgraph problem*. In this problem,

given an edge-weighted undirected graph, we are asked to find a subset of vertices that maximizes the so-called *degree density* (or simply *density*), which is defined as half the average degree of the subgraph induced by the subset. Unlike most optimization problems for dense subgraph discovery, the densest subgraph problem can be solved exactly in polynomial time using some exact algorithms, e.g., Charikar’s linear-programming-based (LP-based) algorithm [Charikar, 2000] and Goldberg’s flow-based algorithm [Goldberg, 1984]. Moreover, there is a simple greedy algorithm called the *greedy peeling*, which obtains a well-approximate solution in almost linear time [Charikar, 2000]. Owing to the solvability and the usefulness of solutions, the densest subgraph problem has actively been studied in data mining, machine learning, and optimization communities [Ghaffari et al., 2019, Gionis and Tsourakakis, 2015, Miller et al., 2010, Papailiopoulos et al., 2014].

Although the densest subgraph problem requires a full input of the graph data, in many real-world applications, the edge weights need to be estimated from *uncertain* measurements. For example, consider protein–protein interaction networks, where vertices correspond to proteins in a cell and edges (resp. edge weights) represent the interactions (resp. the strength of interactions) among the proteins. In the generation process of such networks, the edge weights are estimated through biological experiments using measuring instruments with some noises [Nepusz et al., 2012]. As another example, consider social networks, where vertices correspond to users of some social networking service and edge weights represent the strength of communications (e.g., the number of messages exchanged) among them. In practice, we often need to estimate the edge weights by observing anonymized communications between users [Adar and Ré, 2007]. Therefore we need to develop another framework to resolve such uncertainties.

5.1.2 Chapter Contribution

In this chapter, we introduce a novel learning problem for dense subgraph discovery, which we call *densest subgraph bandits* (*DS bandits*), by incorporating the concepts of stochastic combinatorial bandits [Chen et al., 2013, 2014] into the densest subgraph problem. In DS bandits, an agent is given an undirected graph, whose edge-weights are associated with unknown probability distributions. During the exploration period, the agent chooses a subset of edges (rather than only single edge) to sample, and observes the sum of noisy edge weights in a queried subset. We investigate DS bandits with the objective of best arm identification, that is, the agent must report one subgraph that she believes to be optimal after the exploration period.

The contribution of this chapter is three-fold and can be summarized as follows.

- 1) We address a problem for dense subgraph discovery with no access to a sampling oracle for single edges (Problem 5.1) in the fixed confidence setting. For this problem, we present a general learning algorithm **DS-Lin** (Algorithm 5.2) based on the technique of linear bandits [Auer, 2003]. We provide an upper bound of the number of samples that **DS-Lin** requires to identify an ϵ -optimal solution with probability at least $1 - \delta$ for $\epsilon > 0$ and $\delta \in (0, 1)$ (Theorem 5.1). Our key idea is to utilize an approximation algorithm (Algorithm 5.1) to compute the maximal confidence bound, thereby guaranteeing that the output by **DS-Lin**

is an ϵ -optimal solution and the running time is polynomial in the size of a given graph.

2) To deal with large-sized graphs, we further investigate another problem with access to sampling oracle for any subset of edges (Problem 5.2) with a given fixed budget T . For this problem, we design a scalable and parameter-free algorithm DS-SR (Algorithm 5.3) that runs in $O(n^2T)$, while DS-Lin needs $O(m^2)$ time for updating the estimate, where m is the number of edges. Our key idea is to combine the *successive reject* strategy [Audibert et al., 2010] for the multi-armed bandits and the *greedy peeling* algorithm [Charikar, 2000] for the densest subgraph problem. We prove an upper bound on the probability that DS-SR outputs a solution whose degree density is less than $\frac{1}{2}\text{OPT} - \epsilon$, where OPT is the optimal value (Theorem 5.2).

3) In a series of experimental assessments, we thoroughly evaluate the performance of our proposed algorithms using well-known real-world graphs. We confirm that DS-Lin obtains a nearly-optimal solution even if the minimum size of queryable subsets is larger than the size of an optimal subset, which is consistent with the theoretical analysis. Moreover, we demonstrate that DS-SR finds nearly-optimal solutions even for large-sized instances, while significantly reducing the number of samples for single edges required by a state-of-the-art algorithm.

5.1.3 Related Work

Our learning problem can be seen as a novel variant of combinatorial pure exploration (CPE) problems [Chen et al., 2014, 2016a, 2017]. In the literature, most existing work on CPE has considered the case where the agent obtains feedback from each arm in a pulled subset of arms, i.e., the *semi-bandit* setting, or each individual arm can be queried (e.g. [Chen et al., 2014, 2017, Bubeck et al., 2013, Gabillon et al., 2012, Huang et al., 2018]). Thus, the above studies cannot deal with the aggregated reward from a subset of arms. On the other hand, existing work on the *full-bandit* setting [Rejwan and Mansour, 2020] and all algorithms proposed in Chapter 3 and 4 have assumed that the objective function is linear and the size of subsets to query is exactly k at any round, while our reward function (i.e., the degree density) is not linear and the size of subsets to query is not fixed in advance. If we fix the size of subsets to query to k in DS bandits, the corresponding offline problem (called the densest k -subgraph problem) becomes NP-hard and the best known approximation ratio is just $\Omega(1/n^{1/4+\epsilon})$ for any $\epsilon > 0$ [Bhaskara et al., 2010], where n is the number of vertices.

In the literature of dense subgraph discovery, in order to handle the uncertainty of edge weights, Miyauchi and Takeda [2018] introduced a robust optimization variant of the densest subgraph problem. In their method, all edges are repeatedly queried by a sampling oracle that returns an individual edge weight. However, such a sampling procedure for individual edges is often quite costly or sometimes impossible due to privacy issues or system constraints. On the other hand, it is often affordable to observe aggregated information of a subset of edges, and our model can utilize such weak information which is easier to obtain.

There are other problem variations under uncertain scenarios. Zou [2013] studied the densest subgraph problem on *uncertain graphs*. Uncertain graphs are

a generalization of graphs, which can model the uncertainty of the existence of edges (rather than the uncertainty of edge weights). More formally, an uncertain graph consists of an unweighted graph $G = (V, E)$ and a function $p : E \rightarrow [0, 1]$, where $e \in E$ is present with probability $p(e)$ whereas $e \in E$ is absent with probability $1 - p(e)$. In the problem introduced by Zou [2013], given an uncertain graph G , we are asked to find $S \subseteq V$ that maximizes the expected value of the density. Zou [2013] demonstrated that this problem can be reduced to the original densest subgraph problem, and designed polynomial-time exact algorithm using the reduction. Very recently, Tsourakakis et al. [2020] introduced a novel optimization model, which they refer to as the *risk-averse DSD*. In this model, given an uncertain graph, we are asked to find $S \subseteq V$ that has a large expected value of the density, at the same time, has a small risk. The risk of $S \subseteq V$ is measured by the probability that S is not dense on a given uncertain graph. They showed that the risk-averse DSD can be reduced to NEG-DSD, and designed an efficient approximation algorithm based on the reduction.

In addition to the above uncertain variants, the densest subgraph problem has many other interesting variations. In particular, the size-restricted variants have been actively studied [Andersen and Chellapilla, 2009, Bhaskara et al., 2010, Feige et al., 2001, Khuller and Saha, 2009]. For example, in the densest k -subgraph problem [Feige et al., 2001], given an edge-weighted graph G and a positive integer k , we are asked to find $S \subseteq V$ that maximizes the density $f_w(S)$ (or equivalently $w(S)$) subject to the size constraint $|S| = k$. It is known that such a size restriction makes the problem much harder; in fact, the densest k -subgraph problem is NP-hard and the best known approximation ratio is $\Omega(1/n^{1/4+\epsilon})$ for any $\epsilon > 0$ [Bhaskara et al., 2010]. The densest subgraph problem has also been extended to more general graph structures such as hypergraphs [Hu et al., 2017, Miyauchi et al., 2015] and multilayer networks [Galimberti et al., 2017]. Moreover, to cope with the dynamics of real-world graphs and to model the limited computation resources in reality, some literature has considered dynamic settings [Epasto et al., 2015, Hu et al., 2017, Nasir et al., 2017] and streaming settings [Angel et al., 2012, Bahmani et al., 2012, Bhattacharya et al., 2015, McGregor et al., 2015], respectively. The average-degree density itself has also been generalized by modifying the numerator [Mitzenmacher et al., 2015, Miyauchi and Kakimura, 2018, Tsourakakis, 2015] or the denominator [Kawase and Miyauchi, 2018] of $d(S) = \frac{w(S)}{|S|}$, for some specific purposes.

5.2 Problem Statement

In this section, we describe the online densest subgraph problem in the bandit setting formally.

5.2.1 Densest Subgraph Problem

The densest subgraph problem is defined as follows. Let $G = (V, E, w)$ be an undirected graph, consisting of $n = |V|$ vertices and $m = |E|$ edges, with an edge weight $w : E \rightarrow \mathbb{R}_{>0}$, where $\mathbb{R}_{>0}$ is the set of positive reals. Recall that for a subset of vertices $S \subseteq V$, let $G[S]$ denote the subgraph induced by S , i.e., $G[S] = (S, E(S))$ where $E(S) = \{\{u, v\} \in E : u, v \in S\}$. The *degree density* (or

simply called the *density*) of $S \subseteq V$ is defined as $f_w(S) = w(S)/|S|$, where $w(S)$ is the sum of edge weights of $G[S]$, i.e., $w(S) = \sum_{e \in E(S)} w(e)$. In the densest subgraph problem, given an edge-weighted undirected graph $G = (V, E, w)$, we are asked to find $S \subseteq V$ that maximizes the density $f_w(S)$. There is an LP-based exact algorithm Charikar [2000], which is used in our proposed algorithm (see the next section for the entire procedure).

5.2.2 Densest Subgraph Bandits (DS Bandits)

Here we formally define DS bandits. Suppose that we are given an (unweighted) undirected graph $G = (V, E)$ with $|V| = n$ and $|E| = m$. Assume that each edge $e \in E$ is associated with an unknown distribution ϕ_e over reals. $w : E \rightarrow \mathbb{R}_{>0}$ is the expected edge weights, where $w(e) = \mathbb{E}_{X \sim \phi_e}[X]$. Following the standard assumptions of stochastic multi-armed bandits, we assume that all edge-weight distributions have R -sub-Gaussian tails for some constant $R > 0$. We define the optimal solution as $S^* = \operatorname{argmax}_{S \subseteq V} f_w(S)$.

We first address the setting in which the agent can stop the game at any round if she can return an ϵ -optimal solution for $\epsilon > 0$ with high probability. Let $k > 2$ be the minimal size of queryable subsets of vertices; notice that the agent has no access to a sampling oracle for single edges. The problem is formally defined below.

Problem 5.1 (DS bandits with no access to single edges). *We are given an undirected graph $G = (V, E)$ and a family of queryable subsets of at least k (> 2) vertices $\mathcal{S} \subseteq 2^V$. Let $\epsilon > 0$ be a required accuracy and $\delta \in (0, 1)$ be a confidence level. Then, the goal is to find $S_{\text{OUT}} \subseteq V$ that satisfies $\Pr[f_w(S^*) - f_w(S_{\text{OUT}}) \leq \epsilon] \geq 1 - \delta$, while minimizing the number of samples required by an algorithm (a.k.a. the sample complexity).*

We next consider the setting in which the number of rounds in the exploration phase is fixed and is known to the agent, and the objective is to maximize the quality of the output solution. In this setting, we relax the condition of queryable subsets; assume that the agent is allowed to query any subset of edges. The problem is defined as follows.

Problem 5.2 (DS bandits with a fixed budget). *We are given an undirected graph $G = (V, E)$ and a fixed budget T . The goal is to find $S_{\text{OUT}} \subseteq V$ that maximizes $f_w(S_{\text{OUT}})$ within T rounds.*

Note that Problem 5.1 is often called the *fixed confidence setting* and Problem 5.2 is called the *fixed budget setting* in the bandit literature.

We give the reminder of some notation and another specific notation which will be used in this chapter in Table 5.1.

5.3 LP-based Exact Algorithm for the Densest Subgraph Problem

We describe an exact algorithm for the (offline) densest subgraph problem based on the following LP and simple rounding procedure proposed by Charikar [2000]

Table 5.1: Notation.

Notation	Description
$w(S) = \sum_{e \in E(S)} w_e$	Sum of edge weights in $E(S)$
$f_w(S) = w(S)/ S $	Degree density for weight w and $S \subseteq V$
$\chi_{E(S)} \in \{0, 1\}^E$	Indicator vector of $E(S)$
$N(v) = \{u \in V : \{u, v\} \in E\}$	Set of neighbors of $v \in V$
$\deg_{\max} = \max_{v \in V} N(v) $	Maximum degree of vertices
$\lambda = (\lambda_S)_{S \in \mathcal{S}} \in \Delta(\mathcal{S})$	Predetermined proportions of queries
$\Lambda_\lambda = \sum_{S \in \mathcal{S}} \lambda_S \chi_{E(S)} \chi_{E(S)}^\top$	Design matrix for $\lambda \in \Delta(\mathcal{S})$
$\rho_{\Lambda_\lambda} = \max_{x \in [-1, 1]^m} \ x\ _{\Lambda_\lambda}^2 - 1$	Upper bound of maximal confidence width
$N_S(v) = \{u \in S : \{u, v\} \in E\}$	Set of neighboring vertices of v in $G[S]$
$E_S(v) = \{\{u, v\} \in E : u \in N_S(v)\}$	Set of incident edges to v in $G[S]$
$\hat{X}_F(k) = \frac{1}{k} \sum_{s=1}^k X_F(s)$	Empirical mean of weights for k samples
$\widehat{\deg}_{S,v}(t) = \hat{X}_{E_S(v)}(T_{E(S)}(t))$	Empirical degree in $S \subset V$ for $v \in S$
$\hat{f}(S_{n-t+1}) = \frac{\frac{1}{2} \sum_{v \in S_{n-t+1}} \widehat{\deg}_{S_{n-t+1}}(v, t)}{ S_{n-t+1} }$	Empirical quality function at round t

which we use in our proposed algorithm.

$$\begin{aligned}
& \text{maximize} \quad \sum_{e \in E} w_e x_e \\
& \text{subject to} \quad x_e \geq y_u, \quad x_e \geq y_v \quad \forall e = \{u, v\} \in E, \\
& \quad \sum_{v \in V} y_v = 1, \\
& \quad x_e, y_v \geq 0 \quad \forall e \in E, \forall v \in V.
\end{aligned} \tag{5.1}$$

Let (x^*, y^*) be an optimal solution to the above LP. For a real number $r \geq 0$, the algorithm considers a sequence of subsets vertices $S(r) = \{v \in V : y_v^* \geq r\}$ and finds $r^* \in \arg\max_{r \in [0, 1]} f_w(S(r))$. Such a r^* can be obtained by simply examining $r = y_v^*$ for each $v \in V$. Finally, the algorithm outputs $S(r^*)$. Charikar [2000] proved that the output $S(r^*)$ is an optimal solution to the densest subgraph problem.

5.4 Algorithm for DS Bandits with No Access to Single Edges

In this section, we first present an algorithm for Problem 5.1 based on linear bandits, which we refer to as **DS-Lin**. We then show that **DS-Lin** is (ϵ, δ) -PAC, that is, the output of the algorithm satisfies $\Pr[f_w(S^*) - f_w(S_{\text{OUT}}) \leq \epsilon] \geq 1 - \delta$. Finally, we provide an upper bound of the sample complexity.

5.4.1 DS-Lin Algorithm

We first explain how to obtain the estimate of edge weights and confidence bounds. Then we discuss how to ensure a stopping condition and describe the entire procedure of **DS-Lin**.

Least-squares estimator. We construct an estimate of edge weight w using a sequential noisy observation. For $S \subseteq V$, let $\chi_{E(S)} \in \{0, 1\}^E$ be the indicator vector of $E(S) \subseteq E$, i.e., for each $e \in E$, $\chi_{E(S)}(e) = 1$ if $e \in E(S)$ and $\chi_{E(S)}(e) = 0$ otherwise. Therefore, each subset of edges $E(S)$ for $S \subseteq V$ corresponds to an arm whose feature is an indicator vector of it in linear bandits. For any $t > m$, we define a sequence of indicator vectors as $\mathbf{x}_t = (\chi_{E(S_1)}, \dots, \chi_{E(S_t)}) \in \{0, 1\}^{E \times t}$ and also define the corresponding sequence of observed rewards as $(r_1(S_1), \dots, r_t(S_t)) \in \mathbb{R}^t$. We define $A_{\mathbf{x}_t}^\omega$ as

$$A_{\mathbf{x}_t}^\omega = \sum_{i=1}^t \chi_{E(S_i)} \chi_{E(S_i)}^\top + \omega I \in \mathbb{R}^{E \times E}$$

for a regularized term $\omega > 0$, where I is the identity matrix. Let $b_{\mathbf{x}_t} = \sum_{i=1}^t \chi_{E(S_i)} r_i(S_i) \in \mathbb{R}^E$. Then, the *regularized least-squares estimator* for $w \in \mathbb{R}^E$ can be obtained by

$$\hat{w}_t = (A_{\mathbf{x}_t}^\omega)^{-1} b_{\mathbf{x}_t} \in \mathbb{R}^E. \quad (5.2)$$

Confidence bounds. The basic idea to deal with uncertainty is that we maintain confidence bounds that contain the parameter $w \in \mathbb{R}^E$ with high probability. In the literature of linear bandits, Abbasi-Yadkori et al. [2011] proposed a high probability bound on confidence ellipsoids with a center at the estimate of unknown expected rewards. Although we have already introduced the high probability bound in previous chapters, we state it again for reader's convenience. Plugging it into our setting, we have the following proposition on the ellipsoid confidence bounds for the estimate $\hat{w}_t = A_{\mathbf{x}_t}^{-1} b_{\mathbf{x}_t}$.

Proposition 5.1 (Adapted from Abbasi-Yadkori et al. [2011], Theorem 2). *Let η_t be an R -sub-Gaussian noise for $R > 0$ and $R' = \sqrt{\deg_{\max}} R$. Let $\delta \in (0, 1)$ and assume that the ℓ_2 -norm of edge weight w is less than L . Then, for any fixed sequence \mathbf{x}_t , with probability at least $1 - \delta$, the inequality*

$$|w(S) - \hat{w}_t(S)| \leq C_t \|\chi_{E(S)}\|_{(A_{\mathbf{x}_t}^\omega)^{-1}} \quad (5.3)$$

holds for all $t \in \{1, 2, \dots\}$ and all $S \subseteq V$, where

$$C_t = R' \sqrt{2 \log \frac{\det(A_{\mathbf{x}_t}^\omega)^{\frac{1}{2}}}{\omega^{\frac{m}{2}} \delta}} + \omega^{\frac{1}{2}} L. \quad (5.4)$$

The above bound can be used to guarantee the accuracy of the estimate.

Computing the maximal confidence bound. To identify a solution with an optimality guarantee, the learner ensures whether the estimate is valid by computing the maximal confidence bound among all subsets of vertices. We

consider the following stopping condition:

$$\begin{aligned} f_{\hat{w}_t}(\hat{S}_t) - \frac{C_t \|\chi_{E(\hat{S}_t)}\|_{(A_{\mathbf{x}_t}^\omega)^{-1}}}{|\hat{S}_t|} \\ \geq \max_{S \subseteq V: S \neq \hat{S}_t} f_{\hat{w}_t}(S) + \frac{C_t \max_{S \subseteq V} \|\chi_{E(S)}\|_{(A_{\mathbf{x}_t}^\omega)^{-1}}}{|S|} - \epsilon. \end{aligned}$$

The above stopping condition guarantees that the output satisfies $f_w(S^*) - f_w(S_{\text{OUT}}) \leq \epsilon$ with probability at least $1 - \delta$. However, computing the maximal confidence width $\max_{S \subseteq V} \|\chi_{E(S)}\|_{(A_{\mathbf{x}_t}^\omega)^{-1}}$ by brute force is intractable since it involves an exponential blow-up in the number of $S \subseteq V$. To overcome this computational challenge, we address a relaxed quadratic program:

$$\text{RQP: max. } \|x\|_{(A_{\mathbf{x}_t}^\omega)^{-1}} \quad \text{s.t. } -e \leq x \leq e, \quad (5.5)$$

where $e \in \mathbb{R}^m$ is the vector of all ones.

There is an efficient way to solve RQP using the SDP-based algorithm by Ye [1997] for the following quadratic program with bound constraints:

$$\text{UQP: max. } \sum_{1 \leq i, j \leq m} q_{ij} x_i x_j \quad \text{s.t. } -e \leq x \leq e, \quad (5.6)$$

where $Q = (q_{ij}) \in \mathbb{R}^{m \times m}$ is a given symmetric matrix. Ye [1997] modified the algorithm by Goemans and Williamson [1995] and generalized the proof technique of Nesterov [1998], and then established the constant-factor approximation result for UQP.

Proposition 5.2 (Ye [1997]). *There exists a polynomial-time $\frac{4}{7}$ -approximation algorithm for UQP.*

Note that Ye's algorithm [Ye, 1997] is a randomized algorithm, but it can be derandomized using the technique devised by Mahajan and Ramesh [1999]. The learner can compute an upper bound of the maximal confidence bound $\max_{S \subseteq V} \|\chi_{E(S)}\|_{(A_{\mathbf{x}_t}^\omega)^{-1}}$ by using an approximate solution to UQP obtained by the derandomized version of Ye's algorithm, because it is obvious that the optimal value of UQP is larger than $\max_{S \subseteq V} \|\chi_{E(S)}\|_{(A_{\mathbf{x}_t}^\omega)^{-1}}^2$. Therefore, using Algorithm 5.1, we can ensure the following stopping condition in polynomial time:

$$f_{\hat{w}_t}(\hat{S}_t) - \frac{C_t \|\chi_{E(\hat{S}_t)}\|_{(A_{\mathbf{x}_t}^\omega)^{-1}}}{|\hat{S}_t|} \geq \max_{S \subseteq V: S \neq \hat{S}_t} f_{\hat{w}_t}(S) + \frac{C_t Z_t}{2\alpha} - \epsilon, \quad (5.7)$$

where Z_t denotes the objective value of the approximate solution to RQP and α is a constant-factor approximation ratio of Algorithm 5.1.

Proposed algorithm. Let $T_t(S)$ be the number of times that $S \subseteq \mathcal{S}$ is queried before t -th round in the algorithm. We present our algorithm **DS-Lin**, which is detailed in Algorithm 5.2. Our sampling strategy is based on a given allocation strategy λ defined as follows. Recall that $\Delta(\mathcal{S})$ is a $|\mathcal{S}|$ -dimensional probability

Algorithm 5.1: Unconstrained 0–1 quadratic programming

Input : A positive semidefinite matrix $Q \in \mathbb{R}^{m \times m}$

Output: $x \in [-1, 1]^m$

- 1 Solve the following quadratic programming problem by Ye’s algorithm [Ye, 1997] with derandomization [Mahajan and Ramesh, 1999]:

$$\text{UQP: max. } \sum_{1 \leq i, j \leq m} q_{ij} x_i x_j \quad \text{s.t.} \quad -e \leq x \leq e,$$

and obtain a solution $\bar{x} \in [-1, 1]^m$;

2 return \bar{x}

simplex. We define λ as $\lambda = (\lambda_S)_{S \in \mathcal{S}} \in \Delta(\mathcal{S})$, where λ_S describes the pre-determined proportions of queries to a subset S . As a possible strategy λ , one can use the well-designed strategy called G -allocation [Pukelsheim, 2006, Soare et al., 2014], or simply use uniform allocation, which will be explained in the next paragraph. At each round t , the algorithm calls the sampling oracle for $S_t \in \mathcal{S}$ and observes $r_t(S_t)$. Then, the algorithm updates statistics $A_{\mathbf{x}_t}$ and $b_{\mathbf{x}_t}$, and also updates the estimate \hat{w}_t . To check the stopping condition, the algorithm approximately solves RQP by Algorithm 5.1 and computes the empirical best solution S_t using the LP-based exact algorithm for the densest subgraph problem for $G = (V, E, \hat{w}_t)$. Once the stopping condition is satisfied, the algorithm returns the empirical best solution S_t as output.

Arm allocation strategy. We introduce a possible allocation strategy that can be used in DS-Lin algorithm. To reduce the number of samples, good arm allocation strategy makes confidence bound shrinking fast. We define the G -allocation for a family \mathcal{S} as:

$$\lambda^G(\mathcal{S}) = \operatorname{argmin}_{\lambda \in \Delta(\mathcal{S})} \max_{S \subseteq \mathcal{S}} \|\chi_{E(S)}\|_{\Lambda_\lambda^{-1}}^2. \quad (5.8)$$

There are existing studies that proposed approximate solutions to solve it in the experimental design literature [Bouhtou et al., 2010, Sagnol, 2013]; we can solve a continuous relaxation of the problem by a projected gradient algorithm when the support size $|\mathcal{S}|$ is not so large. For details on G -allocation or standard G -optimal design, see Soare et al. [2014] or see Pukelsheim [2006].

In general, an algorithm that adaptively changes an arm selection strategy based on the past observations at each round, which is called an *adaptive algorithm*, is desired because samples should be allocated for comparison of near-optimal arms in order to reduce the number of samples. On the other hand, the algorithm that fixes all arm selections before observing any reward is called the *static algorithm*. Although the static algorithm is not able to focus on estimating near-optimal arms, it can be used to analyze the worst case optimality. In our text, each arm corresponds to an edge set; it is rare that any set is able to query since the possible choices are exponential. Therefore, we design a static algorithm DS-Lin for solving Problem 5.1.

Algorithm 5.2: DS-Lin

Input : Graph $G = (V, E)$, a family of queryable subsets of at least $k (> 2)$ vertices $\mathcal{S} \subseteq 2^V$, parameter $\epsilon > 0$, parameter $\delta \in (0, 1)$, and allocation strategy λ

Output: $S \subseteq V$

- 1 **for** $t = 1, \dots, m$ **do**
- 2 Choose $S_t \leftarrow \operatorname{argmin}_{S \in \operatorname{supp}(\lambda)} \frac{T_t(S)}{\lambda_S}$;
- 3 Call the sampling oracle for S_t ;
- 4 Observe $r_t(S_t)$;
- 5 $b_{\mathbf{x}_t} \leftarrow b_{\mathbf{x}_{t-1}} + \chi_{E(S_t)} r_t(S_t)$;
- 6 **while** *stopping condition (5.7) is not true* **do**
- 7 $t \leftarrow t + 1$;
- 8 Choose $S_t \leftarrow \operatorname{argmin}_{S \in \operatorname{supp}(\lambda)} \frac{T_t(S)}{\lambda_S}$;
- 9 Call the sampling oracle for S_t and observe $r_t(S_t)$;
- 10 $A_{\mathbf{x}_t}^\omega \leftarrow A_{\mathbf{x}_{t-1}}^\omega + \chi_{E(S_t)} \chi_{E(S_t)}^\top$;
- 11 $b_{\mathbf{x}_t} \leftarrow b_{\mathbf{x}_{t-1}} + \chi_{E(S_t)} r_{S_t}$;
- 12 $\hat{w}_t \leftarrow A_{\mathbf{x}_t}^{-1} b_t$;
- 13 **If** $\hat{w}_t(e) < 0$ **then** $\hat{w}_t(e) = 0$ for each $e \in E$;
- 14 $x \leftarrow \text{Algorithm 5.1 for } (A_{\mathbf{x}_t}^\omega)^{-1}$;
- 15 $Z_t \leftarrow C_t \sqrt{\sum_{1 \leq i, j \leq m} (A_{\mathbf{x}_t}^\omega)^{-1}(i, j) x_i x_j}$;
- 16 $\hat{S}_t \leftarrow \text{Output of the LP-based exact algorithm [Charikar, 2000] for } G(V, E, \hat{w}_t)$;
- 17 **return** $S_{\text{OUT}} \leftarrow \hat{S}_t$

5.4.2 Sample Complexity

We prove that DS-Lin is (ϵ, δ) -PAC and analyze its sample complexity. We define the design matrix for $\lambda \in \Delta(\mathcal{S})$ as $\Lambda_\lambda = \sum_{S \in \mathcal{S}} \lambda_S \chi_{E(S)} \chi_{E(S)}^\top$. We define ρ_{Λ_λ} as $\rho_{\Lambda_\lambda} = \max_{x \in [-1, 1]^m} \|x\|_{\Lambda_\lambda}^2$. Let Δ_{\min} be the minimal gap between the optimal value and the second optimal value, i.e., $\Delta_{\min} = \min_{S \subseteq V: S \neq S^*} f_w(S^*) - f_w(S)$. The next theorem shows an upper bound of the number of queries required by Algorithm 5.2 to output $S_{\text{OUT}} \subseteq V$ that satisfies $\Pr[f_w(S^*) - f_w(S_{\text{OUT}}) \leq \epsilon] \geq 1 - \delta$.

Theorem 5.1. Define $H_\epsilon = \frac{\rho_{\Lambda_\lambda} + \epsilon}{(\Delta_{\min} + \epsilon)^2}$. Then, with probability at least $1 - \delta$, DS-Lin (Algorithm 5.2) outputs $S \subseteq V$ whose density is at least $f_w(S^*) - \epsilon$ and the total number of samples τ is bounded as follows:

if $\omega > 4m(\sqrt{m} + \sqrt{2})^2 \deg_{\max} R^2 H_\epsilon$, then

$$\tau = O\left(\left(\deg_{\max}^2 R^2 \log \frac{1}{\delta} + \omega L^2\right) H_\epsilon\right),$$

and if $\omega \leq \frac{\deg_{\max} R^2}{L^2} \log\left(\frac{1}{\delta}\right)$, then

$$\tau = O\left(m \deg_{\max} R^2 H_\epsilon \log \frac{1}{\delta} + C_{H_\epsilon, \delta}\right)$$

where $C_{H_\varepsilon, \delta}$ is

$$O\left(m \deg_{\max} R^2 H_\varepsilon \log\left(\deg_{\max} R m H_\varepsilon \log \frac{1}{\delta}\right)\right).$$

Note that $\rho_{\Lambda_\lambda^*} = d$ holds if we are allowed to query any subset of vertices and employ G-allocation strategy, i.e., $\lambda^* = \operatorname{argmin}_{\lambda \in \Delta(2^V)} \max_{S \subseteq V} \|\chi_{E(S)}\|_{\Lambda_\lambda}^2 - 1$, which was shown in Kiefer and Wolfowitz [1960]. However, in practice, we should restrict the size of the support to reduce the computational cost as in (5.8); finding a family of subsets of vertices that minimizes ρ_{Λ_λ} may be also related to the optimal experimental design problem [Pukelsheim, 2006].

In the work of Chen et al. [2014], they proved that the lower bound on the sample complexity of general combinatorial pure exploration problems with linear rewards is $\Omega(\sum_{e \in [m]} \frac{1}{\Delta_e^2} \log \frac{1}{\delta})$, where m is the number of base arms and Δ_e is defined as follows. Let \mathcal{M} be any decision class (such as size- k , paths, matchings, and matroids). Let M^* be an optimal subset, i.e., $M^* = \operatorname{argmax}_{M \in \mathcal{M}} \sum_{e \in M} w_e$. For each base arm $e \in [m]$, the gap Δ_e is defined as

$$\Delta_e = \begin{cases} \sum_{e \in M^*} w_e - \max_{M \in \mathcal{M}: e \in M} \sum_{e \in M} w_e & (\text{if } e \notin M^*) \\ \sum_{e \in M^*} w_e - \max_{M \in \mathcal{M}: e \notin M} \sum_{e \in M} w_e & (\text{if } e \in M^*). \end{cases}$$

In the work of Huang et al. [2018], they proposed an algorithm for the combinatorial pure exploration problem with continuous and separable reward functions, and showed its sample complexity is $O(\mathbf{H}_\Lambda + \mathbf{H}_\Lambda m^{-1} \log(\delta^{-1}))$, where $\mathbf{H}_\Lambda = \sum_{i=1}^m \frac{1}{\Lambda_i^2}$. In their definition of \mathbf{H}_Λ , the term Λ_i is called *consistent optimality radius* and it measures how far the estimate can be away from true parameter while the optimal solution in terms of the estimate is still consistent with the true optimal one in the i -th dimension (see Definition 2 in Huang et al. [2018]).

Note that the problem settings in Chen et al. [2014] and Huang et al. [2018] are different from ours; in fact, in our setting the learner can query a subset of edges rather than a base arm and reward function is not linear. Therefore, their lower bound results are not directly applicable to our problem. However, we can see that our sample complexity in Theorem 5.1 is comparable with their bounds because ours is $O(H_\varepsilon \log \delta^{-1} + H_\varepsilon \log(H_\varepsilon \log \delta^{-1}))$ if we ignore the terms irrespective of H_ε and δ .

5.5 Algorithm for DS Bandits with a Fixed Budget

In this section, we propose a scalable and parameter-free algorithm for Problem 5.2 that runs in $O(n^2 T)$ time for a given budget T , and provide theoretical guarantees for the output of the algorithm.

5.5.1 DS-SR Algorithm

The design of our algorithm is based on the Successive Reject (SR) algorithm, which was designed for a regular multi-armed bandits in the fixed budget setting [Audibert et al., 2010] and is known to be the optimal strategy [Carpentier

Algorithm 5.3: DS-SR

Input : Budget $T > 0$, graph $G(V, E)$, sampling oracle
Output: $S \subseteq V$

- 1 $\tilde{\log}(n-1) \leftarrow \sum_{i=1}^{n-1} \frac{1}{i}$;
- 2 $\tilde{T}_0 \leftarrow 0$;
- 3 For $T_0(v) \leftarrow 0$ for each $v \in V$;
- 4 $S_n \leftarrow V$ and $v_0 \leftarrow \emptyset$;
- 5 **for** $t \leftarrow 1, \dots, n-1$ **do**
- 6 $\tilde{T}_t \leftarrow \left\lceil \frac{T - \sum_{i=1}^{n+1} i}{\log(n-1)(n-t)} \right\rceil$;
- 7 $T'_t \leftarrow \left\lceil \frac{\tilde{T}_t}{2|S_{n-t+1}|} \right\rceil$ and $\tau_t \leftarrow T'_t - T'_{t-1}$;
- 8 **for** $v \in S_{n-t+1}$ **do**
- 9 Run Algorithm 5.4 (sampling procedure);
- 10 $\hat{f}(S_{n-t+1}) \leftarrow \frac{\frac{1}{2} \sum_{v \in S_{n-t+1}} \widehat{\deg}_{S_{n-t+1}}(v, t)}{|S_{n-t+1}|}$;
- 11 $v_t \leftarrow \operatorname{argmin}_{v \in S_{n-t+1}} \widehat{\deg}_{S_{n-t+1}}(v, t)$;
- 12 $S_{n-t} \leftarrow S_{n-t+1} \setminus \{v_t\}$;
- 13 **return** $S_{\text{OUT}} \in \{S_2, \dots, S_n\}$ that maximizes $\hat{f}(S_i)$

and Locatelli, 2016]. In classical SR algorithm, we divide the budget T into $K-1$ phases, where K is the number of arms. During each phase, the algorithm uniformly samples an *active* arm that has not been dismissed yet. At the end of each phase, the algorithm dismisses the arm with the lowest empirical mean. After the $K-1$ phases, the algorithm outputs the last surviving arm.

For DS bandits, we employ a different strategy from the classical one because our aim is to find the best subset of vertices in a given graph. Specifically, our algorithm DS-SR is inspired by the graph algorithm called *greedy peeling* [Charikar, 2000], which was designed for approximately solving the densest subgraph problem. DS-SR removes one vertex in each phase, and after all phases are over, it selects the best subset of vertices according to the empirical observation.

Notation. For $F \subseteq 2^E$ and for all phases $t \geq 1$, we denote by $T_F(t)$ the number of times that F was sampled over all rounds from 1 to t , and denote by $X_F(1), \dots, X_F(T_F(t))$ the sequence of associated observed weights. Introduce $\hat{X}_F(k) = \frac{1}{k} \sum_{s=1}^k X_F(s)$ as the empirical mean of weights of F after k samples. For simplicity, we denote $\widehat{\deg}_{S,v}(t) = \hat{X}_{E_S(v)}(T_{E(S)}(t))$.

Proposed algorithm. All procedures of DS-SR are detailed in Algorithm 5.3. Intuitively, DS-SR proceeds as follows. Given a budget T , we divide T into $n-1$ phases. DS-SR maintains a subset of vertices. Initially $S_n \leftarrow V$. In each phase t , for $v \in S_{n-t+1}$, the algorithm uses the sampling oracle for obtaining the estimate of the degree $\widehat{\deg}_{S_{n-t+1}}(v)$, which we refer to as the *empirical degree*. After the sampling procedure, we compute *empirical quality function* $\hat{f}(S_{n-t+1})$ and specify one vertex v_t that should be removed. In Algorithm 5.4, we detail the sampling procedure for obtaining the empirical degree of $v \in S_{n-t+1}$. If v was not a neighbor of v_{t-1} in phase $t-1$, the algorithm samples $E_{S_{n-t+1}}(v)$ for τ_t times, where

Algorithm 5.4: Sampling procedure (subroutine of Algorithm 5.3)

```

1 if  $N_{S_{n-t+1}}(v) = \emptyset$  then
2   Set  $\deg_{S_{n-t+1}}(v, t) = 0$ ;
3 else
4   if  $v \notin N_{S_{n-t+2}}(v_{t-1})$  then
5     Sample  $E_{S_{n-t+1}}(v)$  for  $\tau_t$  times;
6      $Y_t \leftarrow T_{E_{S_{n-t+1}}(v)}(t-1) \widehat{\deg}_{S_{n-t+2}}(v, t)$ ;
7      $\widehat{\deg}_{S_{n-t+1}}(v, t) \leftarrow \frac{Y_t + \tau_t \hat{X}_{E_{S_{n-t+1}}(v)}(\tau_t)}{T_{E_{S_{n-t+1}}(v)}(t-1) + \tau_t}$ ;
8      $T_{E_{S_{n-t+1}}(v)}(t) \leftarrow T_{E_{S_{n-t+1}}(v)}(t-1) + \tau_t$ ;
9   else
10    Sample  $E_{S_{n-t+1}}(v)$  for  $\sum_{i=1}^t \tau_i$  times;
11     $\widehat{\deg}_{S_{n-t+1}}(v, t) \leftarrow \hat{X}_{E_{S_{n-t+1}}(v)}(\sum_{i=1}^t \tau_i)$ ;
12     $T_{E_{S_{n-t+1}}(v)}(t) \leftarrow \sum_{i=1}^t \tau_i$ ;

```

τ_t is set carefully. On the other hand, if v was a neighbor of that, the algorithm samples $E_{S_{n-t+1}}(v)$ for $\sum_{i=1}^t \tau_i$ times. Our eliminate scheme removes a vertex v_t that minimizes the empirical degree, i.e., $v_t \in \operatorname{argmin}_{v \in S_{n-t+1}} \widehat{\deg}_{S_{n-t+1}}(v, t)$. Finally, after $n-1$ phases have been done, DS-SR outputs $S_{\text{OUT}} \subseteq V$ that maximizes the empirical quality function, i.e., $S_{\text{OUT}} = \operatorname{argmax}_{S_i \in \{S_2, \dots, S_n\}} \hat{f}(S_i)$.

5.5.2 Upper Bound on Probability of Error

We provide an upper bound on the probability that the quality of solution obtained by the proposed algorithm is less than $\frac{1}{2}f_w(S^*) - \epsilon$, as shown in the following theorem.

Theorem 5.2. *Given any $T > m$, and assume that the edge weight distribution ϕ_e for each arm $e \in [m]$ has mean $w(e)$ with an R -sub-Gaussian tail. Then, DS-SR (Algorithm 5.3) uses at most T samples and outputs $S_{\text{OUT}} \subseteq V$ such that*

$$\Pr \left[f_w(S_{\text{OUT}}) < \frac{f_w(S^*)}{2} - \epsilon \right] \leq C_{G,\epsilon} \exp \left(- \frac{(T - \sum_{i=1}^{n+1} i) \epsilon^2}{4n^2 \deg_{\max} R^2 \tilde{\log}(n-1)} \right), \quad (5.9)$$

where $C_{G,\epsilon} = \frac{2 \deg_{\max} (n+1)^3 2^n R^2}{\epsilon^2}$ and $\tilde{\log}(n-1) = \sum_{i=1}^{n-1} i^{-1}$.

From the theorem, by setting the RHS of (5.9) to a constant, we see that DS-SR requires a budget of $T = O \left(\frac{n^3 \deg_{\max}}{\epsilon^2} \log \left(\frac{\deg_{\max}}{\epsilon} \right) \right)$. Besides, the upper bound on the probability of error is exponentially decreasing with T .

5.6 Experiments on Real-world Graphs

In this section, we examine the performance of our proposed algorithms DS-Lin and DS-SR. First, we conduct experiments for DS-Lin and show that DS-Lin

Table 5.2: Real-world graphs used in our experiments.

Name	n	m	Description
Karate	34	78	Social network
Lesmis	77	254	Social network
Polbooks	105	441	Co-purchased network
Adjnoun	112	425	Word-adjacency network
Jazz	198	2,742	Social network
Email	1,133	5,451	Communication network
email-Eu-core	986	16,064	Communication network
Polblogs	1,222	16,714	Blog hyperlinks network
ego-Facebook	4,039	88,234	Social network
Wiki-Vote	7,066	100,736	Wikipedia “who-votes-whom”

can find a nearly-optimal solution without sampling any single edges. Second, we perform experiments for DS-SR and demonstrate that DS-SR is applicable to large-sized graphs and significantly reduces the number of samples for single edges, compared to that of the state-of-the-art algorithm. Throughout our experiments, to solve the LPs in Charikar’s algorithm [Charikar, 2000], we used a state-of-the-art mathematical programming solver, Gurobi Optimizer 7.5.1, with default parameter settings. All experiments were conducted on a Linux machine with 2.6 GHz CPU and 130 GB RAM. The code was written in Python.

Dataset. Table 5.2 lists real-world graphs on which our experiments were conducted. Most of those can be found on Mark Newman’s website¹ or in SNAP datasets². For each graph, we construct the edge weight w using the following simple rule, which is inspired by the *knockout densest subgraph model* introduced by Miyauchi and Takeda [2018]. Let $G = (V, E)$ be an unweighted graph and let $S^* \subseteq V$ be an optimal solution to the densest subgraph problem. For each $e \in E$, we set $w(e) = \text{rand}(1, 20)$ if $e \in E(S^*)$, and $w(e) = \text{rand}(1, 100)$ if $e \in E \setminus E(S^*)$, where $\text{rand}(\cdot, \cdot)$ is the function that returns a real value selected uniformly at random from the interval between the two values. That is, we set a relatively small value for each $e \in E(S^*)$ and a relatively large value for each $e \in E \setminus E(S^*)$, which often makes the densest subgraph on $G = (V, E)$ no longer densest on the edge-weighted graph $G = (V, E, w)$. Throughout our experiments, we generate a random noise $\eta(e) \sim \mathcal{N}(0, 1)$ for all $e \in E$.

¹<http://www-personal.umich.edu/mejn/netdata/>

²<http://snap.stanford.edu/>

Algorithm 5.5: Baseline algorithm (Naive)					
<hr/>					
Input : Number of iterations T and a family of queryable subsets of at least k vertices $\mathcal{S} \subseteq 2^V$					
Output: $S \subseteq V$					
1	$w_{\text{avg}} \leftarrow \mathbf{0};$				
2	$t_e \leftarrow 0$ for $e \in E;$				
3	for $t = 1, 2, \dots, T$ do				
4	Choose $S_t \subseteq \mathcal{S}$ uniformly at random;				
5	Call the sampling oracle for S_t and observe $r_t(S_t);$				
6	$t_e \leftarrow t_e + 1$ for $e \in E(S_t);$				
7	Update $w_{\text{avg}}(e) \leftarrow \frac{w_{\text{avg}}(e)(t_e-1) + r_{S_t}/\ell}{t_e}$ for $e \in E(S_t);$				
8	$S \leftarrow$ Output of Charikar's LP-based exact algorithm [Charikar, 2000] for $G(V, E, w_{\text{avg}});$				
9	return S				

Table 5.3: Comparison between DS-Lin and the baseline algorithm (Algorithm 5.5).

Graph	k	DS-Lin	Naive	OPT	$ S^* $
Karate	10	111.08	19.94	111.08	6
	20	111.08	19.94		
	30	111.08	19.94		
Lesmis	10	179.72	177.19	179.72	15
	20	179.72	177.19		
	30	179.72	177.19		
Polbooks	10	227.43	172.69	228.67	19
	20	227.62	172.69		
	30	227.67	172.69		
Adjnoun	10	133.23	53.27	134.83	55
	40	133.62	53.27		
	70	133.53	53.27		
Jazz	10	598.39	170.03	599.43	42
	40	598.81	170.46		
	70	598.81	164.76		
Email	10	223.36	67.24	223.90	58
	40	223.37	67.24		
	70	222.29	67.24		

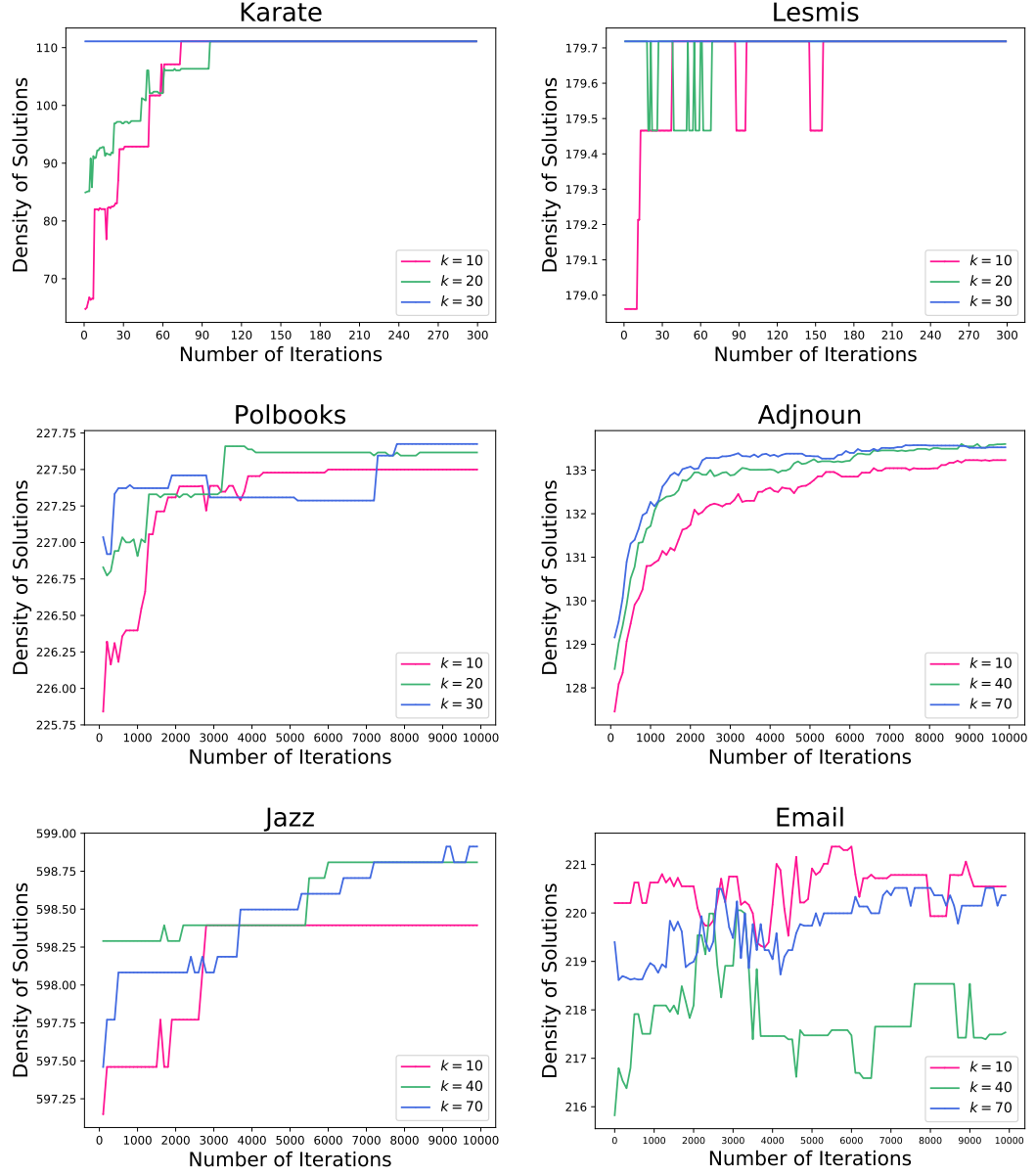


Figure 5.1: Results for the behavior of our proposed algorithm with respect to the number of iterations. Each point is an average over 10 runs of the algorithm.

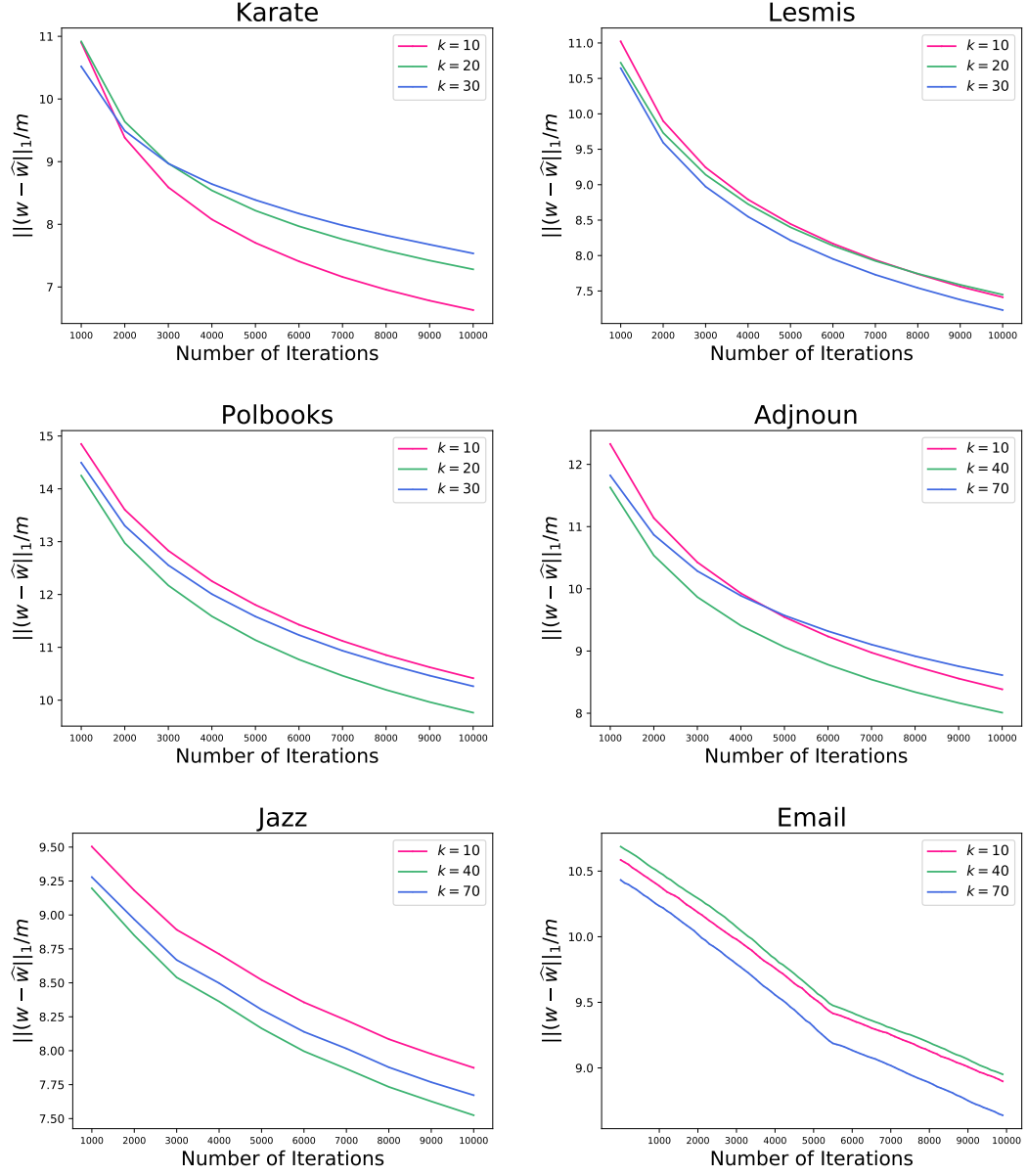


Figure 5.2: Results for the estimation of the expected weight w . Each point is an average over 10 runs of the algorithm.

Table 5.4: Performance of DS-SR. For DS-SR and R-Oracle, the quality of solutions, number of samples, and computation time are averaged over 100 executions.

Graph	DS-SR				R-Oracle			G-Oracle	OPT
	T	Quality	#Samples for single edges	Time(s)	Quality	#Samples for single edges	Time(s)		
Karate	10^3	111.08	58	0.00	111.08	10,296	0.02	111.08	111.08
Lesmis	10^4	177.66	752	0.02	179.72	51,816	0.07	176.29	179.72
Polbooks	10^4	227.43	419	0.02	228.67	214,767	0.22	227.47	228.67
Adjnoun	10^4	133.93	403	0.02	134.83	241,400	0.26	133.97	134.83
Jazz	10^5	599.42	6,837	0.4	599.43	1,115,994	1.49	599.43	599.43
Email	10^6	220.7	23,785	1.51	223.91	22,790,631	20.54	220.93	223.90
email-Eu-core	10^6	792.03	34,393	4.0	792.19	17,509,760	29.69	792.07	792.19
Polblogs	10^6	1211.37	16,508	4.38	1211.44	18,452,256	20.76	1211.44	1211.44
ego-Facebook	10^7	2654.40	103,546	42.61	2783.85	78,175,324	108.82	2654.44	2783.85
Wiki-Vote	10^8	1235.71	3,975,994	425.42	1235.95	288,205,696	638.92	1235.76	1235.95

5.6.1 Experiments for DS-Lin

Baseline. We compare our algorithm with the following naive approach, which we refer to as **Naive**. As well as our proposed algorithm, **Naive** is a kind of algorithm that sequentially accesses a sampling oracle to estimate w and uses uniform sampling strategy. The entire procedure is detailed in Algorithm 5.5.

Parameter settings. Here we use the graphs with up to ten thousand edges. We set the minimum size of queryable subsets $k = 10, 20, 30$ for **Karate**, **Lesmis**, and **Polbooks**, and $k = 10, 40, 70$ for **Adjnoun**, **Jazz**, and **Email**. We construct \mathcal{S} so that the matrix consisting of rows corresponding to the indicator vector of $S \in \mathcal{S}$ has rank m . Each $S \in \mathcal{S}$ is given as follows. We select an integer $\ell \in [k, n]$ and choose $S \subseteq V$ of size ℓ uniformly at random. A uniform allocation strategy is employed by DS-Lin as p , i.e., $p = (1/|\mathcal{S}|)_{S \in \mathcal{S}}$. We set $\lambda = 100$ and $R = 1$. In our theoretical analysis, we provided an upper bound of the number of queries required by DS-Lin for $\epsilon > 0$ and $\delta \in (0, 1)$. However, such an upper bound is usually too large in practice. Therefore, we terminate the while-loop of our algorithm once the number of iterations exceeds 10,000 except for the initialization steps. To be consistent, we also set $T = m + 10000$ in **Naive**.

Results. Here we compare our proposed algorithm DS-Lin with **Naive** in terms of the quality of solutions. The results are summarized in Table 5.3. The quality of output S is measured by its density in terms of w which is unknown to the learner. For all instances, we run each algorithm for 10 times, and report the average value. The last two columns of Table 5.3 represent the optimal value and the size of an optimal solution, respectively. As can be seen, our algorithm outperforms the baseline algorithm; in fact, our algorithm always obtains a nearly-optimal solution. It should be noted that this trend is valid even if k is quite large; in particular, even if k is larger than the size of the densest subgraph on the edge-weighted graph $G = (V, E, w)$, our algorithm succeeds in detecting a vertex subset that is almost densest in terms of w . Finally, we briefly report the running time of our proposed algorithm with 10,000 iterations. For small-sized instances, **Karate**, **Lesmis**, **Polbooks**, and **Adjnoun**, the algorithm runs in a few minutes. For medium-sized instances, **Jazz** and **Email**, the algorithm runs in a few hours.

Behavior of DS-Lin. We also report how the density of solutions approaches to such a quality and behavior of DS-Lin with respect to the number of iterations. In the previous experiment, we confirmed that the solution obtained after 10,000 iterations is almost densest in terms of unknown w . A natural question here is how the density of solutions approaches to such a sufficiently large value. In other words, does our algorithm is sensitive to the choice of the number of iterations? In this section, we answer these questions by conducting the following experiments. We terminate the while-loop of our algorithm once the number of iterations exceeds 0, 100, 200, ..., 10,000, and follow the density values of solutions in terms of w . For each instance, we again run our algorithm for ten times, and report the average value.

The results are shown in Figure 5.1. As can be seen, as the number of iterations increases, the density value converges to the sufficiently large value

Algorithm 5.6: Greedy peeling (G-Oracle)

Input : Graph $G = (V, E, w)$
Output: $S \subseteq V$
1 $S_{|V|} \leftarrow V$;
2 **for** $i \leftarrow |V|, \dots, 2$ **do**
3 Find $v_i \in \operatorname{argmin}_{v \in S_i} \deg_{S_i}(v)$;
4 $S_{i-1} \leftarrow S_i \setminus \{v_i\}$;
5 **return** $S_i \in \{S_1, \dots, S_{|V|}\}$ that maximizes $f_w(S)$

(close to the optimum). Although the density value sometimes drops down, the decrease is quite small.

Estimation of the expected weight. We next explain the reason why our proposed algorithm DS-Lin performs fairly well. To this end, we focus on the quality of the estimated edge weight obtained by the algorithm. We measure the quality of the estimated edge weight \hat{w}_t by comparing with the expected weight w ; specifically, we compute $\|w - \hat{w}_t\|_1/m$. The experimental setup is exactly the same as that in the previous section.

The results are depicted in Figure 5.2. As can be seen, as the number of iterations increases, \hat{w}_t converges to the true edge weight w . It is very likely that the high performance of our algorithm is derived from the high-quality estimation of the expected edge weight w .

5.6.2 Experiments for DS-SR

Compared algorithms. To demonstrate the performance of DS-SR for Problem 5.2, we also implement two algorithms G-Oracle and R-Oracle. G-Oracle is the greedy peeling algorithm with the knowledge of the expected weight w [Charikar, 2000], which is detailed in Algorithm 5.6. Note that we are interested in how the quality of solutions by DS-SR is close to that of G-Oracle. R-Oracle is the state-of-the-art robust optimization algorithm proposed by Miyauchi and Takeda [2018] with the use of *edge-weight space* $W = \times_{e \in E} [\min\{w(e) - 1, 0\}, w(e) + 1]$. We describe the entire procedure of R-Oracle in Algorithm 5.7. Their robust optimization model takes intervals of edge weights as its input. We generate the intervals $W = \times_{e \in E} [l_e, r_e]$ based on unknown edge weight w , i.e., $l_e = \min\{w_e - 1, 0\}$ and $r_e = w_e + 1$. Algorithm 5.7 first obtains the optimal solution $S_{w^-}^*$ in terms of extreme edge weight $w^- = (l_e)_{e \in E}$ and computes the value of $f_{w^-}(S_{w^-}^*)$. Then, for each single edge $e \in E$, the algorithm calls the sampling oracle for an appropriate number of times and obtains the empirical mean. Using the empirical means, the algorithm constructs intervals $W_{\text{out}} \leftarrow \times_{e \in E} [l_e^{\text{out}}, r_e^{\text{out}}]$, and computes a densest subgraph S_{out} on G with $w_{\text{out}}^- = (l_e^{\text{out}})_{e \in E}$. For R-Oracle, we set $\gamma = 0.9$ and $\varepsilon = 0.9$ as in Miyauchi and Takeda [2018].

Results. For DS-SR, in order to make \tilde{T}_t positive, we run the experiments with a budget $T = 10^{\lceil \log_{10} \sum_{i=1}^{n+1} i \rceil}$ for all instances. The results are summarized in Table 5.4. The quality of output is again evaluated by its density in terms of w . For DS-SR and R-Oracle, we list the total number of samples for individual

Algorithm 5.7: Robust optimization with oracle intervals (R-Oracle)

Input : Graph $G = (V, E)$, oracle intervals $W = \times_{e \in E} [l_e, r_e]$ where $l_e = \min\{w_e - 1, 0\}$ and $r_e = w_e + 1$, sampling oracle, $\gamma \in (0, 1)$, and $\varepsilon > 0$

Output: (S_{out})

```
1 for each  $e \in E$  do
2   if  $l_e = r_e$  then
3      $l_e^{\text{out}} \leftarrow l_e, r_e^{\text{out}} \leftarrow r_e;$ 
4   else
5      $S_{w^-}^* \leftarrow$  Output of Charikar's LP-based exact algorithm for
       $G(V, E, w^-);$ 
6      $t_e \leftarrow \left\lceil \frac{m(r_e - l_e)^2 \ln \frac{2m}{\gamma}}{\varepsilon^2 f_{w^-}(S_{w^-}^*)^2} \right\rceil;$ 
7     Sample  $e$  for  $t_e$  times;
8      $\hat{p}_e \leftarrow \hat{X}_e(t_e);$ 
9      $\delta \leftarrow \frac{\varepsilon f_{w^-}(S_{w^-}^*)}{\sqrt{2m}};$ 
10     $l_e^{\text{out}} \leftarrow \max\{l_e, \hat{p}_e - \delta\}$  and  $r_e^{\text{out}} \leftarrow \min\{r_e, \hat{p}_e + \delta\};$ 
11  $W_{\text{out}} \leftarrow \times_{e \in E} [l_e^{\text{out}}, r_e^{\text{out}}];$ 
12  $S_{\text{out}} \leftarrow$  Output of Charikar's LP-based exact algorithm for  $G(V, E, W_{\text{out}});$ 
13 return  $S_{\text{out}};$ 
```

edges used in the algorithms. To observe the scalability, we also report the computation time of the algorithms. We perform them 100 times on each graph. As can be seen, DS-SR required much less samples for single edges than that of R-Oracle but still can find high-quality solutions. The quality of solutions by DS-SR is comparable with that of G-Oracle, which has a prior knowledge of expected weights w . Moreover, in terms of computation time, DS-SR efficiently works on large-sized graphs with about ten thousands of edges. For the number of samples for single edges in DS-SR, Figure 5.3 depicts the fraction of the size of edge subsets queried in DS-SR. We see that in the execution of DS-SR, the fraction of the number of queries for single edges is less than 30%.

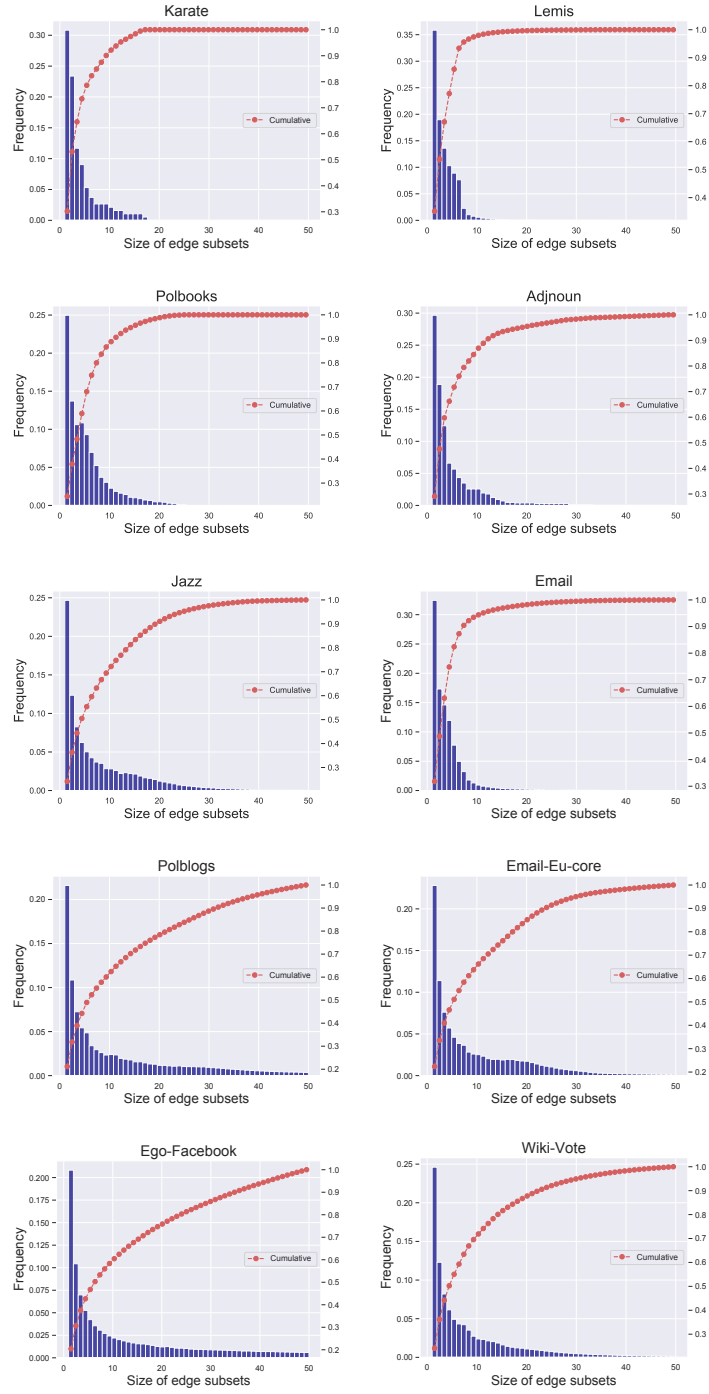


Figure 5.3: Fraction of the size of queried edge subsets in DS-SR (cumulative) over 100 runs.

5.7 Proofs of Theoretical Results

5.7.1 Technical Lemmas for Theorem 5.1

We introduce random event \mathcal{E}_t which characterizes the event that the confidence bounds of any feasible solution $S \in V$ are valid at round t . We define a random event \mathcal{E}_t as follows:

$$\mathcal{E}_t = \{\forall S \in V \text{ and } v \in S, \quad |w(S) - \hat{w}_t(S)| \leq C_t \|\chi_{E(S)}\|_{(A_{\mathbf{x}_t}^\omega)^{-1}}\}. \quad (5.10)$$

The following lemma states that event $\mathcal{E} = \bigcap_{t=1}^\infty \mathcal{E}_t$ occurs with high probability.

Lemma 5.1. *The event \mathcal{E} occurs with probability at least $1 - \delta$.*

The proof is omitted since it is straightforward from Proposition 5.1 and union bounds.

Lemma 5.2. *For a fixed round $t > m$, assume that \mathcal{E}_t occurs. Then, if the algorithm stops at round t , the output of the algorithm S_{OUT} satisfies $f_w(S^*) - f_w(S_{\text{OUT}}) \leq \epsilon$.*

Proof. If $S_{\text{OUT}} = S^*$, we obviously have the desired result. Then, we shall assume $S_{\text{OUT}} \neq S^*$.

$$\begin{aligned} f_w(S_{\text{OUT}}) &\geq f_{w_t}(S_{\text{OUT}}) - \frac{C_t \|\chi_{E(S_{\text{OUT}})}\|_{(A_{\mathbf{x}_t}^\omega)^{-1}}}{|S_{\text{OUT}}|} \\ &\geq \max_{S \neq S_{\text{OUT}}: S \subseteq V} f_{\hat{w}_t}(S) + \frac{C_t Z_t}{2\alpha} - \epsilon \\ &\geq f_{\hat{w}_t}(S^*) + \frac{\max_{x \in [-1,1]^m} \|x\|_{(A_{\mathbf{x}_t}^\omega)^{-1}}}{2} - \epsilon \\ &\geq f_{\hat{w}_t}(S^*) + \frac{\max_{S \subseteq V} \|\chi_{E(S)}\|_{(A_{\mathbf{x}_t}^\omega)^{-1}}}{2} - \epsilon \\ &\geq f_{\hat{w}_t}(S^*) + \frac{C_t \|\chi_{E(S^*)}\|_{(A_{\mathbf{x}_t}^\omega)^{-1}}}{|S^*|} - \epsilon \\ &\geq f_w(S^*) - \epsilon, \end{aligned}$$

where the first and last inequalities follow from the event \mathcal{E}_t , and the second inequality follows from the stopping condition, and the third inequality follows from the definition of Z_t and approximation ratio α . \square

In Miyauchi and Takeda [2018], they provided the following lemma, which we also use to prove Theorem 5.1.

Lemma 5.3 (Miyauchi and Takeda [2018], Lemma 2). *Let $G = (V, E)$ be an undirected graph. Let w_1 and w_2 be edge-weight vectors such that $\|w_1 - w_2\|_\infty \leq \beta$*

holds for $\beta > 0$. Then, for any $S \subseteq V$, it holds that

$$|f_{w_1}(S) - f_{w_2}(S)| \leq \sqrt{\frac{m}{2}} \cdot \beta.$$

5.7.2 Proof of Theorem 5.1

Proof. We know that the event \mathcal{E} holds with probability at least $1 - \delta$ from Lemma 5.1. Therefore, we only need to prove that, under event \mathcal{E} , the algorithm returns a set whose density is at least $f_w(S^*) - \epsilon$ and provide an upper bound of number of queries. From Lemma 5.2, on the event \mathcal{E} , the algorithm outputs $S_{\text{OUT}} \subseteq V$ that satisfies $f_w(S^*) - f_w(S_{\text{OUT}}) \leq \epsilon$.

Next, we focus on bounding the number of queries. Recall that Algorithm 5.2 employs a stopping condition:

$$f_{\hat{w}_t}(\hat{S}_t) - \frac{C_t \|\chi_{E(\hat{S}_t)}\|}{|\hat{S}_t|} \geq \max_{S \neq \hat{S}_t : S \subseteq V} f_{\hat{w}_t}(S) + \frac{C_t Z_t}{2\alpha} - \epsilon, \quad (5.11)$$

where Z_t denotes the objective of the approximate solution to P1. A sufficient condition of the stopping condition is that for S^* and for $t > m$,

$$f_{\hat{w}_t}(S^*) - \frac{C_t \|\chi_{E(S^*)}\|}{|S^*|} \geq \max_{S \neq S^* : S \subseteq V} f_{\hat{w}_t}(S) + \frac{C_t Z_t}{2\alpha} - \epsilon, \quad (5.12)$$

Since $Z_t \leq \max_{x \in [-1,1]^m} \|x\|_{(A_{\mathbf{x}_t}^\omega)^{-1}}$ and $\|\chi_{E(S^*)}\| \leq \max_{x \in [-1,1]^m} \|x\|_{(A_{\mathbf{x}_t}^\omega)^{-1}}$, the following inequality is also a sufficient condition to stop:

$$\begin{aligned} & f_{\hat{w}_t}(S^*) - \frac{C_t \max_{x \in [-1,1]^m} \|x\|_{(A_{\mathbf{x}_t}^\omega)^{-1}}}{|S^*|} \\ & \geq \max_{S \neq S^* : S \subseteq V} f_{\hat{w}_t}(S) + \frac{C_t \max_{x \in [-1,1]^m} \|x\|_{(A_{\mathbf{x}_t}^\omega)^{-1}}}{2\alpha} - \epsilon. \end{aligned} \quad (5.13)$$

Recall that $T_t(S)$ be the number of times that $S \subseteq \mathcal{S}$ is queried before t -th round in the algorithm. We denote $\lambda_{\mathbf{x}_t}$ by $\lambda_{\mathbf{x}_t} = (T_t(S)/t)_{S \in \mathcal{S}}$. From the above definitions, the design matrix is $A_{\mathbf{x}_t} = t\Lambda_{\lambda_{\mathbf{x}_t}}$. Recall that $\rho_{\Lambda_\omega} = \max_{x \in [-1,1]^m} \|x\|_{\Lambda_p^{-1}}^2$, and let $\rho_{\Lambda_{\lambda_{\mathbf{x}_t}}} = \max_{x \in [-1,1]^m} \|x\|_{\Lambda_{\lambda_{\mathbf{x}_t}}^{-1}}^2$. From the fact that $\lim_{t \rightarrow \infty} \Lambda_{\lambda_{\mathbf{x}_t}} = \Lambda_\omega$, for sufficiently large $t \gg m$ we have that $|\rho_{\Lambda_\omega} - \rho_{\Lambda_{\lambda_{\mathbf{x}_t}}}| \leq \epsilon$ with probability at least $1 - \frac{\delta}{2}$ where $\delta \in (0, 1)$ and $\epsilon > 0$. Let $\bar{S} = \arg\max_{S \neq S^* : S \subseteq V} f_{\hat{w}_t}(S)$. Then, (5.13) is rewritten as follows:

$$f_{\hat{w}_t}(S^*) - f_{\hat{w}_t}(\bar{S}) \geq \left(\frac{1}{|S^*|} + \frac{1}{2\alpha} \right) C_t \sqrt{\frac{\rho_{\Lambda_\omega} + \epsilon}{t}} - \epsilon. \quad (5.14)$$

Next, we show the following inequality.

$$f_{\hat{w}_t}(S^*) - f_{\hat{w}_t}(\bar{S}) \geq \Delta_{\min} - \sqrt{2m}C_t \sqrt{\frac{\rho_{\Lambda_\omega} + \epsilon}{t}}. \quad (5.15)$$

From Proposition 5.1, with probability $1 - \frac{\delta}{2}$, we have

$$\|w - \hat{w}_t\|_\infty \leq C_t \max_S \|\chi_{E(S)}\|_{(A_{\mathbf{x}_t}^\omega)^{-1}}.$$

From Lemma 5.3, we see that $f_{\hat{w}_t}(S^*) - f_w(S^*) \geq -\sqrt{\frac{m}{2}}C_t \max_{S \subseteq V} \|\chi_{E(S)}\|_{(A_{\mathbf{x}_t}^\omega)^{-1}}$ and $f_w(\bar{S}) - f_{\hat{w}_t}(\bar{S}) \geq -\sqrt{\frac{m}{2}}C_t \max_{S \subseteq V} \|\chi_{E(S)}\|_{(A_{\mathbf{x}_t}^\omega)^{-1}}$. Therefore, for sufficiently large $t \gg m$ such that $|\rho_{\Lambda_\omega} - \rho_{\Lambda_{\lambda_{\mathbf{x}_t}}}| \leq \epsilon$ with probability at least $1 - \frac{\delta}{2}$, we have that

$$\begin{aligned} f_{\hat{w}_t}(S^*) - f_{\hat{w}_t}(\bar{S}) &\geq f_w(S^*) - f_w(\bar{S}) - \sqrt{2m}C_t \max_{S \subseteq V} \|\chi_{E(S)}\|_{(A_{\mathbf{x}_t}^\omega)^{-1}} \\ &\geq f_w(S^*) - f_w(\bar{S}) - \sqrt{2m}C_t \max_{x \in \{0,1\}^m} \|x\|_{(A_{\mathbf{x}_t}^\omega)^{-1}} \\ &= f_w(S^*) - f_w(\bar{S}) - \sqrt{2m}C_t \sqrt{\frac{\rho_{\Lambda_{\lambda_{\mathbf{x}_t}}}}{t}} \\ &\geq f_w(S^*) - f_w(\bar{S}) - \sqrt{2m}C_t \sqrt{\frac{\rho_{\Lambda_\omega} + \epsilon}{t}} \\ &\geq \Delta_{\min} - \sqrt{2m}C_t \sqrt{\frac{\rho_{\Lambda_\omega} + \epsilon}{t}}. \end{aligned}$$

Then, we obtain (5.15). Combining (5.14) and (5.15), we see that the following inequality is a sufficient condition to stop:

$$\Delta_{\min} - \sqrt{2m}C_t \sqrt{\frac{\rho_{\Lambda_\omega} + \epsilon}{t}} \geq \left(\frac{1}{|S^*|} + \frac{1}{2\alpha} \right) C_t \sqrt{\frac{\rho_{\Lambda_\omega} + \epsilon}{t}} - \epsilon. \quad (5.16)$$

Solving (5.16) with respect to t , we obtain

$$t \geq (\sqrt{2m} + |S^*|^{-1} + 2\alpha^{-1})^2 C_t^2 H_\epsilon, \quad (5.17)$$

where recall that H_ϵ is defined as

$$H_\epsilon = \frac{\rho_{\Lambda_\omega} + \epsilon}{(\Delta_{\min} + \epsilon)^2}.$$

Therefore, from the above, we obtain $t \geq (\sqrt{2m} + |S^*|^{-1} + 2\alpha^{-1})^2 C_t^2 H_\epsilon$ as a sufficient condition to stop. Let $\tau > m$ be the stopping time of the algorithm. From the above discussion and $|S^*| \geq 2$, we see that

$$\tau \leq \left(\sqrt{2m} + \frac{\alpha + 1}{2} \right)^2 C_\tau^2 H_\epsilon. \quad (5.18)$$

Now we bound C_τ in (5.18). We have $\det(A_{\mathbf{x}_\tau}^\omega) \leq (\lambda + \tau)^m$, which is obtained by Lemma 10 in Abbasi-Yadkori et al. [2011]. Then, we obtain the following inequality in the similar manner in Theorem 2 in Xu et al. [2018]:

$$\begin{aligned} C_\tau &\leq R' \sqrt{2 \log \frac{\det(A_{\mathbf{x}_\tau}^\omega)^{\frac{1}{2}} \det(\omega I)^{-\frac{1}{2}}}{\delta}} + \omega^{\frac{1}{2}} L \\ &\leq R' \sqrt{2 \log \frac{1}{\delta} + m \log \left(1 + \frac{\tau}{\omega}\right)} + \omega^{\frac{1}{2}} L. \end{aligned} \quad (5.19)$$

Using (5.19), we give an upper bound of τ . We also use a similar proof strategy as in Xu et al. [2018]. First, let us consider the case $\lambda > 4m(\sqrt{m} + \sqrt{2})^2 R'^2 H_\varepsilon$, where recall that $R' = \sqrt{\deg_{\max}} R$. Using the facts that $(a + b)^2 \leq 2(a^2 + b^2)$ for $a, b > 0$ and $\log(1 + x) \leq x$, we have

$$\begin{aligned} \tau &\leq \left(\sqrt{2m} + \frac{\alpha + 1}{2} \right)^2 C_\tau^2 H_\varepsilon \\ &\leq 2 \left(\sqrt{2m} + \frac{\alpha + 1}{2} \right)^2 H_\varepsilon \left(4R'^2 \log \frac{1}{\delta} + \frac{R'^2 m}{\omega} \tau + \omega L^2 \right). \end{aligned}$$

Thus, we obtain

$$\begin{aligned} \tau &\leq \left(1 - 2 \left(\sqrt{2m} + \frac{\alpha + 1}{2} \right)^2 \frac{R'^2 m H_\varepsilon}{\omega} \right)^{-1} (8H_\varepsilon R'^2 \log 1/\delta + 2H_\varepsilon \omega L^2) \\ &\leq 2(8H_\varepsilon R'^2 \log 1/\delta + 2H_\varepsilon \omega L^2), \end{aligned}$$

where the last inequality holds from $\lambda > 4m(\sqrt{m} + \sqrt{2})^2 R'^2 H_\varepsilon$ and $\alpha = \sqrt{\frac{4}{7}}$. Therefore, in this case, we obtain

$$\tau = O \left(\left(\deg_{\max}^2 R^2 \log \frac{1}{\delta} + \omega L^2 \right) H_\varepsilon \right).$$

Second, we consider $\omega \leq \frac{R'^2}{L^2} \log \left(\frac{1}{\delta} \right)$. From this bound and (5.19), we have

$$C_\tau \leq 2R' \sqrt{2 \log \frac{1}{\delta} + m \log \left(1 + \frac{\tau}{\omega} \right)}.$$

Let $V(m, \alpha) = \left(\sqrt{2m} + \frac{\alpha+1}{2} \right)^2$. Therefore, we obtain

$$\tau \leq V(m, \alpha) C_\tau^2 H_\varepsilon \leq 4V(m, \alpha) R'^2 \left(2 \log \frac{1}{\delta} + m \log \left(1 + \frac{\tau}{\omega} \right) \right) H_\varepsilon.$$

Let $N = 8V(m, \alpha)R'^2 \log \frac{1}{\delta} H_\varepsilon$ and let τ' be a parameter satisfying:

$$\tau = N + 4V(m, \alpha)R'^2 m \log \left(1 + \frac{\tau'}{\omega} \right) H_\varepsilon. \quad (5.20)$$

It is easy to see $\tau' \leq \tau$. Then, we have

$$\begin{aligned} \tau' &\leq \tau = N + 4V(m, \alpha)R'^2 m \log \left(1 + \frac{\tau'}{\omega} \right) H_\varepsilon \\ &\leq N + 4V(m, \alpha)R'^2 m \sqrt{\frac{\tau'}{\omega}} H_\varepsilon, \end{aligned}$$

where the second inequality follows from the fact $\log(1+x) \leq \sqrt{x}$. Solving this quadratic inequality for $\sqrt{\tau'}$, we have

$$\begin{aligned} \sqrt{\tau'} &\leq \frac{2V(m, \alpha)R'^2 m H_\varepsilon}{\sqrt{\omega}} + \sqrt{\frac{4V(m, \alpha)^2 R'^4 m^2 H_\varepsilon^2}{\omega} + N} \\ &\leq 2\sqrt{\frac{4V(m, \alpha)^2 R'^4 m^2 H_\varepsilon^2}{\omega} + N}. \end{aligned} \quad (5.21)$$

Let $M = 2\sqrt{\frac{4V(m, \alpha)^2 R'^4 m^2 H_\varepsilon^2}{\omega} + N}$. We can give an upper bound τ by (5.20) and (5.21) as follows:

$$\tau \leq 8V(m, \alpha)R'^2 H_\varepsilon \log \frac{1}{\delta} + 4H_\varepsilon V(m, \alpha)R'^2 m \log \left(1 + \frac{M}{\omega} \right).$$

Note that

$$\log M = O \left(\log \left(R'^4 m^4 H_\varepsilon^2 + m R'^2 \log \frac{1}{\delta} H_\varepsilon \right) \right)$$

since $V(m, \alpha) = O(m)$. From the above and $R' = \sqrt{\deg_{\max}} R$, we obtain

$$\tau = O \left(m \deg_{\max} R^2 H_\varepsilon \log \frac{1}{\delta} + C(H_\varepsilon, \delta) \right)$$

where

$$\begin{aligned} C(H_\varepsilon, \delta) &= m \deg_{\max} R^2 H_\varepsilon \log \left(\deg_{\max}^2 R^4 m^4 H_\varepsilon^2 + m \deg_{\max} R^2 H_\varepsilon \log \frac{1}{\delta} \right) \\ &= O \left(m \deg_{\max} R^2 H_\varepsilon \log \left(\deg_{\max} R m H_\varepsilon \log \frac{1}{\delta} \right) \right). \end{aligned}$$

□

5.7.3 Technical lemmas for Theorem 5.2

First we introduce a standard concentration inequality of sub-Gaussian random variables [Chen et al., 2014].

Lemma 5.4 (Chen et al. [2014], Lemma 6). *Let X_1, \dots, X_k be k independent random variables such that, for each $i \in [k]$, random variable $X_i - \mathbb{E}[X_i]$ is R -sub-Gaussian distributed, i.e., $\forall a \in \mathbb{R}$, $\mathbb{E}[\exp(aX_i - a\mathbb{E}[X_i])] \leq \exp(R^2 a^2)/2$. Let $\bar{X} = \frac{1}{k} \sum_{i=1}^k X_i$ denote the average of these random variables. Then, for any $\lambda > 0$, we have*

$$\Pr[|\bar{X} - \mathbb{E}[\bar{X}]| \geq \lambda] \leq 2 \exp\left(-\frac{k\lambda^2}{2R^2}\right).$$

For $S \subseteq V$, $v \in S$, and expected weight w , we denote by $\deg_S^*(v)$ the weighted degree of $v \in S$ on $G[S]$ in terms of the true edge weight w . We show the following lemma used for analysis of Algorithm 5.3.

Lemma 5.5. *Given an phase $t \in \{1, \dots, n-1\}$, we define random event*

$$\mathcal{E}'_t = \left\{ \forall v \in S_{n-t+1}, \left| \deg_{S_{n-t+1}}^*(v) - \widehat{\deg}_{S_{n-t+1}}(v, t) \right| \leq \epsilon \right\}. \quad (5.22)$$

Then, we have

$$\Pr\left[\bigcap_{t=1}^{n-1} \mathcal{E}'_t\right] \geq 1 - \frac{2\deg_{\max} 2^n R^2}{\epsilon^2} (n+1)^3 \exp\left(-\frac{(T - \sum_{i=1}^{n+1} i)\epsilon^2}{4n^2 \deg_{\max} \log(n-1)}\right). \quad (5.23)$$

Proof. For any $S \subseteq V$, and $v \in S$, recall that $X_{E_S(v)}(i)$ is i -th observation of edge-weights $E_S(v)$ for $i \in [k]$. Then, $X_{E_S(v)}(i) - \mathbb{E}[X_{E_S(v)}(i)]$ follows a $(\sqrt{|E_S(v)|} R)$ -sub-Gaussian distribution. We can assume that the sequence of weights for each subset of edges is drawn before the beginning of the game. Thus $\hat{X}_{E_S(v)}(k)$ is well defined even if $E_S(v)$ has not been actually sampled k times. Therefore, from Lemma 5.4, for any $\epsilon > 0$ we have that

$$\begin{aligned} & \Pr\left[\left|\mathbb{E}[\hat{X}_{E_S(v)}(k)] - \hat{X}_{E_S(v)}(k)\right| \geq \epsilon\right] \\ & \leq 2 \exp\left(-\frac{k\epsilon^2}{2|E_S(v)|R^2}\right) \leq 2 \exp\left(-\frac{k\epsilon^2}{2\deg_{\max} R^2}\right). \end{aligned} \quad (5.24)$$

Fix $t \in \{1, \dots, n-1\}$ and fix a vertex $v \in S_{n-t+1}$ in a phase t . If it holds that $|N_{S_{n-t+1}}(v)| = 0$, it is obvious that $\deg_{S_{n-t+1}}(v) = \widehat{\deg}_{S_{n-t+1}}(v) = 0$. Therefore, we will consider a vertex $v \in S_{n-t+1}$ such that $|N_{S_{n-t+1}}(v)| \geq 1$ in the rest of the proof. By the definition of \tilde{T}_t for $t \in [1, 2, \dots, n-1]$, we have

$$\tilde{T}_t \geq \frac{T - \sum_{i=1}^{n+1} i}{(n-t)\log(n-1)} \geq \frac{T - \sum_{i=1}^{n+1} i}{n\log(n-1)}.$$

Then for $T_{E_{S_{n-t+1}}}(v)(t)$, we have:

$$T_{E_{S_{n-t+1}}}(v)(t) = \sum_{i=1}^t \tau_i = \sum_{i=1}^t (T'_i - T'_{i-1}) = T'_t \geq \frac{\tilde{T}_t}{2|S_{n-t+1}|} \geq \frac{T - \sum_{i=1}^{n+1} i}{2n^2 \log(n-1)}. \quad (5.25)$$

Let k' be the RHS of (5.25), i.e. $k' = \frac{T - \sum_{i=1}^{n+1} i}{2n^2 \log(n-1)}$. For $v \in S_{n-t+1}$, we have

$$\begin{aligned} & \Pr \left[\left| \deg_{S_{n-t+1}}^*(v) - \widehat{\deg}_{S_{n-t+1}}(v, t) \right| \geq \epsilon \right] \\ &= \Pr \left[\left| \mathbb{E}[\hat{X}_{E_{S_{n-t+1}}}(v)(T_{E_{S_{n-t+1}}}(v)(t))] - \hat{X}_{E_{S_{n-t+1}}}(v)(T_{E_{S_{n-t+1}}}(v)(t)) \right| \geq \epsilon \right] \\ &\leq \Pr \left[\exists S \subseteq V, u \in S \left| \mathbb{E}[\hat{X}_{E_S}(u)(T_{E_{S_{n-t+1}}}(v)(t))] - \hat{X}_{E_S}(u)(T_{E_{S_{n-t+1}}}(v)(t)) \right| \geq \epsilon \right] \\ &\leq \sum_{S \in 2^V} \sum_{u \in S} \sum_{k=k'}^{\infty} \Pr \left[\left| \mathbb{E}[\hat{X}_{E_S}(u)(k)] - \hat{X}_{E_S}(u)(k) \right| \geq \epsilon \right] \\ &\leq \sum_{S \in 2^V} \sum_{u \in S} \sum_{k=k'}^{\infty} 2 \exp \left(-\frac{k\epsilon^2}{2\deg_{\max} R^2} \right) \\ &= \sum_{S \in 2^V} \sum_{u \in S} \frac{2 \exp \left(-\frac{k'\epsilon^2}{2\deg_{\max} R^2} \right)}{\exp \left(\frac{\epsilon^2}{2\deg_{\max} R^2} \right) - 1} \\ &\leq \sum_{S \in 2^V} \sum_{u \in S} \frac{2 \exp \left(-\frac{k'\epsilon^2}{2\deg_{\max} R^2} \right)}{\frac{\epsilon^2}{2\deg_{\max} R^2}} \\ &= \sum_{S \in 2^V} \sum_{u \in S} \frac{4\deg_{\max} R^2}{\epsilon^2} \exp \left(-\frac{(T - \sum_{i=1}^{n+1} i)\epsilon^2}{4n^2 \deg_{\max} R^2 \log(n-1)} \right) \\ &\leq \frac{4\deg_{\max} 2^n n R^2}{\epsilon^2} \exp \left(-\frac{(T - \sum_{i=1}^{n+1} i)\epsilon^2}{4n^2 \deg_{\max} R^2 \log(n-1)} \right), \end{aligned} \quad (5.26)$$

where the third inequality follows by (5.24) and the fourth inequality follows by $e^{-x} \geq 1 - x$.

Now using (5.26) and taking a union bound for all $t \in \{1, \dots, n-1\}$ and all $v \in S_{n-t+1}$, we obtain

$$\begin{aligned} & \Pr \left[\bigcap_{t=1}^{n-1} \mathcal{E}'_t \right] \\ &= 1 - \Pr \left[\exists t \in \{1, \dots, n-1\}, v \in S_{n-t+1} \left| \deg_{S_{n-t+1}}^*(v) - \widehat{\deg}_{S_{n-t+1}}(v, t) \right| \geq \epsilon \right] \\ &\geq 1 - \sum_{t=1}^{n-1} \sum_{v \in S_{n-t+1}} \Pr \left[\left| \deg_{S_{n-t+1}}^*(v) - \widehat{\deg}_{S_{n-t+1}}(v, t) \right| \geq \epsilon \right] \end{aligned}$$

$$\begin{aligned}
&\geq 1 - \sum_{t=1}^{n-1} \sum_{v \in S_{n-t+1}} \frac{4 \deg_{\max} 2^n n R^2}{\epsilon^2} \exp \left(-\frac{(T - \sum_{i=1}^{n+1} i) \epsilon^2}{4n^2 \deg_{\max} R^2 \log(n-1)} \right) \\
&= 1 - \frac{4 \deg_{\max} 2^n n R^2}{\epsilon^2} \sum_{t=1}^{n-1} |S_{n-t+1}| \exp \left(-\frac{(T - \sum_{i=1}^{n+1} i) \epsilon^2}{4n^2 \deg_{\max} R^2 \log(n-1)} \right) \\
&= 1 - \frac{4 \deg_{\max} 2^n n R^2}{\epsilon^2} \sum_{t=1}^{n-1} (n-t+1) \exp \left(-\frac{(T - \sum_{i=1}^{n+1} i) \epsilon^2}{4n^2 \deg_{\max} R^2 \log(n-1)} \right) \\
&= 1 - \frac{2 \deg_{\max} 2^n R^2}{\epsilon^2} n^2 (n+1) \exp \left(-\frac{(T - \sum_{i=1}^{n+1} i) \epsilon^2}{4n^2 \deg_{\max} R^2 \log(n-1)} \right) \\
&\geq 1 - \frac{2 \deg_{\max} 2^n R^2}{\epsilon^2} (n+1)^3 \exp \left(-\frac{(T - \sum_{i=1}^{n+1} i) \epsilon^2}{4n^2 \deg_{\max} R^2 \log(n-1)} \right).
\end{aligned}$$

□

5.7.4 Proof of Theorem 5.2

Proof. First, we verify that the algorithms requires at most T queries. In each phase t , the number of samples Algorithm 5.4 requires is at most $\tilde{T}_t + |S_{n-t+1}|$, since we have that

$$\begin{aligned}
\sum_{v \in S_{n-t+1}} \sum_{i=1}^t \tau_i &= \sum_{v \in S_{n-t+1}} \sum_{i=1}^t T'_i - T'_{i-1} = \sum_{v \in S_{n-t+1}} T'_t \\
&\leq \sum_{v \in S_{n-t+1}} \left(\frac{\tilde{T}_t}{|S_{n-t+1}|} + 1 \right) \leq \tilde{T}_t + |S_{n-t+1}|.
\end{aligned}$$

Therefore, the total number of queries used by the algorithm is bounded by

$$\begin{aligned}
\sum_{t=1}^{n-1} (\tilde{T}_t + |S_{n-t+1}|) &\leq \sum_{t=1}^{n-1} \tilde{T}_t + \sum_{t=1}^{n-1} (n-t+1) \\
&\leq \sum_{t=1}^{n-1} \left(\frac{T - \sum_{i=1}^{n+1} i}{(n-t) \log(n-1)} + 1 \right) + \sum_{t=1}^{n-1} (n-t+1) \\
&\leq \sum_{t=1}^{n-1} \frac{T - \sum_{i=1}^{n+1} i}{(n-t) \log(n-1)} + \sum_{i=1}^{n+1} i \\
&= \frac{(T - \sum_{i=1}^{n+1} i) \log(n-1)}{\log(n-1)} + \sum_{i=1}^{n+1} i \\
&= T - \sum_{i=1}^{n+1} i + \sum_{i=1}^{n+1} i = T.
\end{aligned}$$

Lemma 5.5 implies that the random event $\mathcal{E}' := \bigcap_{t=1}^{n-1} \mathcal{E}'_t$ occurs with probability at least $1 - \frac{2 \deg_{\max} 2^n R^2}{\epsilon^2} (n+1)^3 \exp \left(-\frac{(T - \sum_{i=1}^{n+1} i) \epsilon^2}{4n^2 \deg_{\max} R^2 \log(n-1)} \right)$. We shall assume

the event \mathcal{E}' occurs in the rest of the proof, because we only need to show that the algorithm outputs a solution S_{OUT} that guarantees $f_w(S_{\text{OUT}}) \geq \frac{f_w(S^*)}{2} - \epsilon$ under \mathcal{E}' .

Let $S^* \subseteq V$ be an optimal solution in terms of the expected weight w . Choose an arbitrary vertex $v \in S^*$. From the optimality of $S^* \subseteq V$, it holds that

$$f_w(S^*) = \frac{w(S^*)}{|S^*|} \geq \frac{w(S^* \setminus \{v\})}{|S^*| - 1} = f_w(S^* \setminus \{v\}).$$

By using the fact that $w(S^* \setminus \{v\}) = w(S^*) - \deg_{S^*}^*(v)$, the above inequality can be transformed into

$$\deg_{S^*}^*(v) \geq f_w(S^*). \quad (5.27)$$

Let $S_\tau \subseteq V$ be the last subset over the phases that satisfies $S_\tau \supseteq S^*$ and let $\tau \in [1, \dots, n]$ be its phase. Let τ_{OUT} be the phase t such that $S_{n-t+1} = S_{\text{OUT}}$. Then we have

$$\begin{aligned} f_w(S_{\text{OUT}}) &= \frac{\frac{1}{2} \sum_{v \in S_{\text{OUT}}} \deg_{S_{\text{OUT}}}^*(v)}{|S_{\text{OUT}}|} \\ &\geq \frac{\frac{1}{2} \sum_{v \in S_{\text{OUT}}} \left(\widehat{\deg}_{S_{\text{OUT}}}(v, \tau_{\text{OUT}}) - \epsilon \right)}{|S_{\text{OUT}}|} \\ &\geq \frac{\frac{1}{2} \sum_{v \in S_\tau} \widehat{\deg}_{S_\tau}(v, \tau)}{|S_\tau|} - \frac{\epsilon}{2}. \end{aligned}$$

where the first inequality follows from event \mathcal{E}' , the second inequality follows from the greedy choice of S_{OUT} . Recall that the algorithm removes the vertex that satisfies $v_\tau \in \operatorname{argmin}_{v \in S_\tau} \widehat{\deg}_{S_\tau}(v, \tau)$ in the phase τ . Therefore, from the definition of S_τ , it is clear that $v_\tau \in S^*$. Using this property, we further have that

$$\begin{aligned} \frac{\frac{1}{2} \sum_{v \in S_\tau} \widehat{\deg}_{S_\tau}(v, \tau)}{|S_\tau|} - \frac{\epsilon}{2} &\geq \frac{\frac{1}{2} \sum_{v \in S_\tau} \widehat{\deg}_{S_\tau}(v_\tau, \tau)}{|S_\tau|} - \frac{\epsilon}{2} \\ &= \frac{1}{2} \widehat{\deg}_{S_\tau}(v_\tau, \tau) - \frac{\epsilon}{2} \\ &\geq \frac{1}{2} \deg_{S_\tau}^*(v_\tau) - \epsilon \\ &\geq \frac{1}{2} \deg_{S^*}^*(v_\tau) - \epsilon \\ &\geq \frac{1}{2} f_w(S^*) - \epsilon, \end{aligned}$$

where the second inequality follows from event \mathcal{E}' , and third inequality follows from the fact $S_\tau \supseteq S^*$, and the last inequality follows from the fact $v_\tau \in S^*$ and inequality (5.27). Therefore, we obtain $f_w(S_{\text{OUT}}) \geq \frac{1}{2} f_w(S^*) - \epsilon$. That concludes

the proof. □

5.8 Conclusion

In this chapter, we have investigated the research question: how can we identify a dense component in networks with less access to information on two specific users? We have introduced a novel online variant of the densest subgraph problem by bringing the concepts of combinatorial pure exploration, which we refer to as the DS bandits. We have proposed an (ϵ, δ) -PAC algorithm called **DS-Lin**, and provided a polynomial sample complexity guarantee. Our key technique is to utilize an approximation algorithm using SDP for confidence bound maximization. Then, to deal with large-sized graphs, we have proposed an algorithm called **DS-SR** by combining the successive reject strategy and the greedy peeling algorithm. We have provided an upper bound of the probability that the quality of the solution obtained by the algorithm is less than $\frac{1}{2}\text{OPT} - \epsilon$. Computational experiments using well-known real-world graphs have demonstrated the effectiveness of the proposed algorithm.

We have focused on specific graph optimization in the bandit setting in this chapter. In the next chapter, we will discuss how to deal with more general class of reward functions and what kind of assumptions for reward functions will be required to design polynomial-time algorithms with sample complexity bounds. We will also investigate how to go beyond the full-bandit settings.

Chapter 6

Combinatorial Pure Exploration with Partial-linear Feedback for Nonlinear Rewards

In the previous chapters, we have studied the full-bandit setting, where an agent observes the total sum of random rewards from chosen super arm. Although full-bandit settings can capture many practical situations, it may happen that we cannot always observe outcomes from some of the chosen arms due to privacy concerns or system constraints. To overcome this issue, we now investigate how to deal with partial-linear feedback, which encompasses several families of feedback models such as full-bandit and semi-bandit feedback. In this chapter, we propose a novel generalization of combinatorial pure exploration with full-bandit feedback (CPE-BL) with flexible feedback structures, called combinatorial pure exploration with partial linear feedback (CPE-PL), of which all the problems addressed in Chapters 3–5 are special cases. For CPE-PL, we develop the first polynomial-time algorithm, which simultaneously addresses limited feedback, general reward functions and combinatorial action spaces (e.g., matroids, matchings and s - t paths), and provide its sample complexity analysis.

6.1 Preliminaries

In this section, we provide the formal problem statement. Then, we introduce necessary assumptions in this chapter. We also give some illustration examples of the problem setting and partial-linear feedback to understand our model clearly.

6.1.1 Problem Statement

CPE-PL is a generalization of CPE-BL to partial linear feedback and nonlinear reward functions. In CPE-PL, each super arm $x \in \mathcal{X}$ is associated with a transformation matrix $M_x \in \mathbb{R}^{m_x \times d}$, whose row dimension m_x depends on x . At each timestep t , an agent pulls a super arm x_t and observes a random linear feedback vector $y_t = M_{x_t}(\theta + \eta_t) \in \mathbb{R}^{m_{x_t}}$, where η_t is the noise vector. Meanwhile, the agent gains a random reward with expectation of $\bar{r}(x_t, \theta)$. Note that for each pull of super arm x_t , the actual expected reward $\bar{r}(x_t, \theta)$ may not be a part of the linear feedback vector y_t and thus may not be directly observed by the agent.

We consider the fixed confidence setting; given a confidence $\delta \in (0, 1)$, the agent aims to use as few samples as possible to identify the optimal super arm with probability at least $1 - \delta$.

CPE-PL allows more flexible feedback structures than CPE-BL or BAI-LB, and encompasses several families of sub-problems including full-bandit feedback, semi-bandit feedback and nonlinear reward functions. For example, when $M_x = x^\top \in \mathbb{R}^{1 \times d}$ for all $x \in \mathcal{X}$, this model reduces to CPE-BL. When $M_x = \text{diag}(x) \in \mathbb{R}^{d \times d}$ for all $x \in \mathcal{X}$, this model reduces to combinatorial pure exploration with semi-bandit feedback.

6.1.2 Assumptions

The regret minimization algorithms for combinatorial bandits with partial-linear feedback have been proposed in Lin et al. [2014], Chaudhuri and Tewari [2016]. In this chapter, we study the pure exploration version and inherit the two technical assumptions from Lin et al. [2014], Chaudhuri and Tewari [2016] in order to design an efficient algorithm.

Assumption 6.1 (Lipschitz continuity of the expected reward function). *There exists a constant L_p such that for any $x \in \mathcal{X}$ and any $\theta_1, \theta_2 \in \mathbb{R}^d$, $|\bar{r}(x, \theta_1) - \bar{r}(x, \theta_2)| \leq L_p \|\theta_1 - \theta_2\|_2$.*

Assumption 6.2 (Global observer set). *There exists a global observer set $\sigma = \{x_1, x_2, \dots, x_{|\sigma|}\} \subseteq \mathcal{X}$, such that the stacked $\sum_{i=1}^{|\sigma|} m_{x_i} \times d$ transformation matrix $M_\sigma = (M_{x_1}; M_{x_2}; \dots; M_{x_{|\sigma|}})$ is of full column rank ($\text{rank}(M_\sigma) = d$).*

Then, the Moore-Penrose pseudoinverse M_σ^+ satisfies $M_\sigma^+ M_\sigma = I_d$, where I_d is the $d \times d$ identity matrix. We justify Assumption 6.2 by the fact that without the existence of global observer set, the agent cannot recover θ and may not distinguish two different actions. With Assumption 6.2, we can systematically construct a global observer set with $|\sigma| \leq d$ by sequentially adding an action that strictly increases the rank of M_σ , until M_σ reaches the full rank.

6.1.3 Illustration Examples

In this section, we will provide specific examples to illustrate the feedback model of CPE-PL by considering the crowdsourcing scenario (see Figure 6.1). There are $N = 4$ workers and $M = 4$ tasks, which can be represented as a complete bipartite graph $G = (L, R, E)$. In this bipartite graph, each edge corresponds to a pair between a worker and a task, and its edge-weight corresponds to the utility that is unknown to the agent. In the model of CPE-PL, each base arm $i \in \{1, 2, \dots, d\}$ prescribes each edge $e \in E$ where $d = |E|$ and its mean is the unknown edge-weight θ_e . Let edges $(1, 1), (2, 3), (3, 4), (4, 2)$ be the first, second, third, fourth base arms, respectively. Suppose that random outcome at round t is $\theta + \eta_t = (0.2, 0.3, 0.4, 0.6, \dots)^\top \in \mathbb{R}^d$, and the agent will pull the action $x_t = (1, 1, 1, 1, 0, 0, \dots, 0)^\top$. In this case, the examples of transformation matrix $M_x \in \mathbb{R}^{m_x \times d}$ and corresponding random feedback $y_t = M_{x_t}(\theta + \eta_t) \in \mathbb{R}^{m_{x_t}}$ can be written as follows.

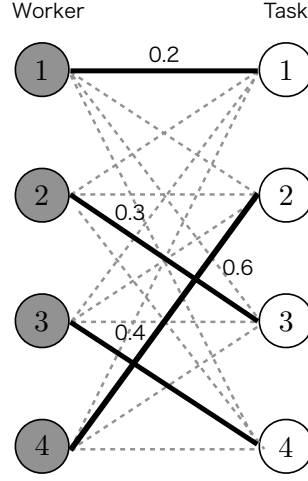


Figure 6.1: Example of the crowdsourcing application.

If $M_{x_t} = (1, 0, 0, \dots, 0)$, the agent can observe only first base arm, that is,

$$M_{x_t}(\theta + \eta_t) = (1, 0, 0, \dots, 0) \begin{pmatrix} 0.2 \\ 0.3 \\ 0.4 \\ 0.6 \\ \vdots \end{pmatrix} = 0.2 = y_t.$$

If $M_{x_t} = \text{diag}(x)$, the agent can obtain semi-bandit feedback, that is,

$$\begin{aligned} M_{x_t}(\theta + \eta_t) &= \begin{pmatrix} 1 & 0 & 0 & 0 & \dots \\ 0 & 1 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & \dots \\ 0 & 0 & 0 & 1 & \dots \end{pmatrix} \begin{pmatrix} 0.2 \\ 0.3 \\ 0.4 \\ 0.6 \\ \vdots \end{pmatrix} \\ &= \begin{pmatrix} 0.2 \\ 0.3 \\ 0.4 \\ 0.6 \end{pmatrix} \\ &= y_t. \end{aligned}$$

If $M_{x_t} = x_t^\top$, the agent can only observe the sum of the rewards (i.e., full-bandit feedback), that is,

$$M_{x_t}(\theta + \eta_t) = (1, 1, 1, 1, 0, \dots, 0) \begin{pmatrix} 0.2 \\ 0.3 \\ 0.4 \\ 0.6 \\ \vdots \end{pmatrix} = 1.5 = y_t.$$

As another example, the agent may obtain the sum of the rewards from two base

arms as follows,

$$\begin{aligned}
M_{x_t}(\theta + \eta_t) &= \begin{pmatrix} 1 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 1 & 1 & \cdots \end{pmatrix} \begin{pmatrix} 0.2 \\ 0.3 \\ 0.4 \\ 0.6 \\ \vdots \end{pmatrix} \\
&= \begin{pmatrix} 0.5 \\ 1.0 \end{pmatrix} \\
&= y_t.
\end{aligned}$$

Note that reward functions in CPE-PL can be nonlinear, while feedback model is linear as illustrated above.

6.2 Related Work and Chapter Contribution

Related work. Huang et al. [2018] designed the algorithm for CPE-MB with continuous and separable reward functions. Although their model considers nonlinear rewards, it cannot deal with either full-bandit or partial linear feedback and thus cannot be applied to our setting. Recently, the best arm identification in cascading bandits is investigated by Zhong et al. [2020]. Their problem setting can be seen as another model of top- k identification with partial feedback. However, their algorithm is designed for identifying the top- k actions and thus it cannot be applied to the case under other combinatorial structures such as paths, matchings, and matroids. In Chapters 3 and 4, we have studied the case where the reward function is linear and the agent observes full-bandit feedback, which is a sub-problem of CPE-PL. In Chapter 5, we also have studied another special case of CPE-PL, where the offline optimization is the densest subgraph problem and the agent observes full-bandit feedback for a set of edges. The proposed algorithms and analysis use the property of the average degree. All the proposed algorithms in Chapters 3–5 cannot deal with partial-linear feedback. Chen et al. [2020] studied an adaptation of CPE to the dueling bandit setting, where at each timestep the agent plays a pair of edges (base arms) in a bipartite graph and observes a random outcome of the comparison with the objective of identifying the optimal matching. They only consider relative feedback between two compared base arms in the matching case, and thus their algorithms cannot be applied to CPE-PL. It should be noted that CPE-PL cannot be translated to either CPE-MB, CPE-BL, or BAI-LB. For example, when the reward function is $(x^\top \theta) / \|x\|_1$ and $M_x = \text{diag}(x)$, CPE-PL reduces to a semi-bandit problem with nonlinear reward function, and no existing algorithm could solve this problem.

Contribution. We introduce a novel model of combinatorial pure exploration with partial linear feedback, a generalization of CPE-BL, which simultaneously models more flexible partial feedback, general (possibly nonlinear) reward and combinatorial action spaces, and finds various applications. For example, in on-line ranking [Chaudhuri and Tewari, 2017], a company recommends their products to users by presenting rankings of entire items, and wants to find the best

ranking with limited feedback on the top-ranked item due to user burden constraints and privacy concerns. In crowdsourcing [Lin et al., 2014], an employer assigns crowdworkers to tasks according to the worker-task performance, and wants to find the best matching with limited feedback on a small subset of the completed tasks, owing to the burden of entire feedback and privacy issues (see Section 6.5 for detailed applications). For CPE-PL, we propose a general polynomial-time algorithmic framework **GCB-PE** with sample complexity analysis. Our algorithm and analysis provide an efficient solution to identifying the optimal action under combinatorial action space and partial feedback. Our result demonstrates that (a) our algorithm runs much faster than all the others, and (b) ours is the only one that correctly outputs the optimal action for a nonlinear reward function among all the compared algorithms.

6.3 Static Algorithm for CPE-PL

In this section, we present the first polynomial-time algorithm **GCB-PE** for CPE-PL with sample complexity analysis, and discuss its further improvements via a non-uniform allocation strategy. We also give practical applications for CPE-PL and explain the corresponding global observer set and sample complexity result in these scenarios.

We illustrate **GCB-PE** in Algorithm 6.1. **GCB-PE** estimates the environment vector θ by repeatedly pulling the global observer set $\sigma = \{x_1, x_2, \dots, x_{|\sigma|}\}$, which in turn helps estimate the expected rewards $\bar{r}(x, \theta)$ of all super arms $x \in \mathcal{X}$ using the Lipschitz continuity (Assumption 6.1). We call one pull of global observer set σ *one exploration round*, the specific procedure of which is described as follows: for the n -th exploration round, the learner plays all actions in $\sigma = \{x_1, x_2, \dots, x_{|\sigma|}\}$ once and respectively observes feedback $y_1, y_2, \dots, y_{|\sigma|}$, the stacked vector of which is denoted by $\vec{y}_n = (y_1; y_2; \dots; y_{|\sigma|})$. The estimate of environment vector θ in this exploration round is $\hat{\theta}_n = M_\sigma^+ \vec{y}_n$, where M_σ^+ is the Moore-Penrose pseudoinverse of M_σ . From Assumption 6.2, we have $\mathbb{E}[\hat{\theta}_n] = \theta$. Then, we can use the independent estimates in multiple rounds, i.e., $\hat{\theta}(n) = \frac{1}{n} \sum_{j=1}^n \hat{\theta}_j$, to obtain an accurate estimate of θ .

As in Lin et al. [2014], we define a constant

$$\beta_\sigma := \max_{\eta_1, \dots, \eta_{|\sigma|} \in [-1, 1]^d} \|(M_\sigma^\top M_\sigma)^{-1} \sum_{i=1}^{|\sigma|} M_{x_i}^\top M_{x_i} \eta_i\|_2,$$

which only depends on global observer set σ , and bounds the estimate error of one exploration round.

Lemma 6.1. *For any n , $\|\hat{\theta}_n - \theta\|_2 \leq \beta_\sigma$.*

Based on the upper bound β_σ , we further design a global confidence radius $\text{rad}_n = \sqrt{2\beta_\sigma^2 \log(4n^2 e^2 / \delta) / n}$ for the estimate $\hat{\theta}(n)$, and show that with high probability, rad_n bounds the estimate error of $\hat{\theta}(n)$.

Compared with **GCB** in Lin et al. [2014], which works for the regret minimization metric of the combinatorial partial monitoring game with linear feedback problem, **GCB-PE** targets the best action identification and mainly controls the

Algorithm 6.1: GCB-PE

Input : Confidence level δ , global observer set σ , constant β_σ ,
Lipschitz constant L_p

1 **for** $s = 1, \dots, |\sigma|$ **do**
2 Pull x_s in observer set σ , and observe y_s ;
3 $n \leftarrow 1$;
4 $\vec{y}_1 \leftarrow (y_1; y_2; \dots; y_{|\sigma|})$;
5 $\hat{\theta}_1 \leftarrow M_\sigma^+ \vec{y}_1$ and $\hat{\theta}(1) \leftarrow \hat{\theta}_1$;
6 **while** *true* **do**
7 $\hat{x} \leftarrow \operatorname{argmax}_{x \in \mathcal{X}} \bar{r}(x, \hat{\theta}(n))$;
8 $\hat{x}^- \leftarrow \operatorname{argmax}_{x \in \mathcal{X} \setminus \{\hat{x}\}} \bar{r}(x, \hat{\theta}(n))$;
9 $\text{rad}_n \leftarrow \sqrt{\frac{2\beta_\sigma^2 \log(\frac{4n^2 e^2}{\delta})}{n}}$;
10 **if** $\bar{r}(\hat{x}, \hat{\theta}(n)) - \bar{r}(\hat{x}^-, \hat{\theta}(n)) > 2L_p \cdot \text{rad}_n$ **then**
11 **return** \hat{x} ;
12 **else**
13 **for** $s = 1, \dots, |\sigma|$ **do**
14 Pull x_s in observer set σ , and observe y_s ;
15 $n \leftarrow n + 1$;
16 $\vec{y}_n \leftarrow (y_1; y_2; \dots; y_{|\sigma|})$;
17 $\hat{\theta}_n \leftarrow M_\sigma^+ \vec{y}_n$;
18 $\hat{\theta}(n) \leftarrow \frac{1}{n} \sum_{j=1}^n \hat{\theta}_j$;
19 **end**

Output: \hat{x}

stopping time of the exploration phase rather than balancing the frequency of exploration and exploitation phases. For the pure exploration metric, our global confidence radius rad_n is novelly designed to bound the estimate error. In addition, the stopping condition, which uses the designed confidence radius and Lipschitz continuity of the expected reward function, is also novelly adopted to fit the CPE-PL setting.

The computational efficiency of GCB-PE relies on the polynomial-time offline maximization oracle for the specific combinatorial instance, which is used in the two argmax operations in GCB-PE. It is reasonable to assume the existence of polynomial-time offline maximization oracle, otherwise we cannot efficiently address the exponentially large action space even if the real environment vector θ is known.

GCB-PE provides a general algorithmic framework to simultaneously address the partial linear feedback, general expected reward and combinatorial action space.

6.4 Theoretical Analysis of GCB-PE

We give the sample complexity of GCB-PE below.

Theorem 6.1. *With probability at least $1 - \delta$, the GCB-PE algorithm (Algo-*

arithm 6.1) will return the optimal super arm x^* with sample complexity

$$O\left(\frac{|\sigma|\beta_\sigma^2 L_p^2}{\Delta_{\min}^2} \log\left(\frac{\beta_\sigma^2 L_p^2}{\Delta_{\min}^2 \delta}\right)\right),$$

where $|\sigma| \leq d$.

When the expected reward function is linear, i.e. $\bar{r}(x, \theta) = x^\top \theta$, we have $L_p = \sqrt{m}$, where m ($\leq d$) is the maximum number of base arms a super arm contains. In addition, $\beta_\sigma = \text{Poly}(d)$ in several practical applications of CPE-PL (see Section 6.5 for our detailed discussion).

While the sample complexity of GCB-PE is sometimes worse than the CPE-BL or BAI-LB algorithms, it solves a more general class of problems than CPE-BL and BAI-LB. We emphasize that our contribution mainly focuses on proposing the first polynomial-time algorithm GCB-PE that simultaneously addresses combinatorial action space, partial linear feedback and nonlinear reward function. The lower bound for CPE-PL is still an open problem left for future work.

GCB-PE can be further improved by employing a *non-uniform* allocation strategy when considering the global observer set σ with multiplicity: we can obtain such an allocation by solving an optimization $\arg\min_{\lambda \in \Delta(\sigma)} \beta_\sigma(\lambda)$ and rounding the result, where

$$\beta_\sigma(\lambda) := \max_{\eta_1, \dots, \eta_{|\sigma|} \in [-1, 1]^d} \|(M_\sigma^\top M_\sigma)^{-1} \sum_{i=1}^{|\sigma|} \lambda_i M_{x_i}^\top M_{x_i} \eta_i\|_2.$$

Since uniform sampling is not essential in our analysis, the proposed improvement for Assumption 6.2 and keeps our theoretical analysis.

6.5 Application Examples

CPE-PL characterizes more flexible feedback structures than CPE-BL (or BAI-LB) and finds many real-world applications. Below we present two practical applications and discuss the global observer set (Assumption 6.2) and parameter β_σ .

Online ranking. Consider that a company wishes to recommend their products to users by presenting the ranked list of items. Due to user burden constraints and privacy concerns, collecting a large amount of data on the relevance of all items might be infeasible, and thus the company usually collects the relevance of only the top-ranked item [Chaudhuri and Tewari, 2015, 2016, 2017]. In this scenario, an agent selects a permutation of d items (each action x is a permutation) at each step, and observes the relevance of the top-ranked item, i.e., M_x contains a single row with 1 in the place of the top-ranked item and 0 everywhere else. The objective is to identify the best permutation as soon as possible. Then, we can construct a global observable set σ to be the set of any d actions which places a distinct item at top. Here M_σ is the $d \times d$ identity matrix and $\beta_\sigma = \sqrt{d}$.

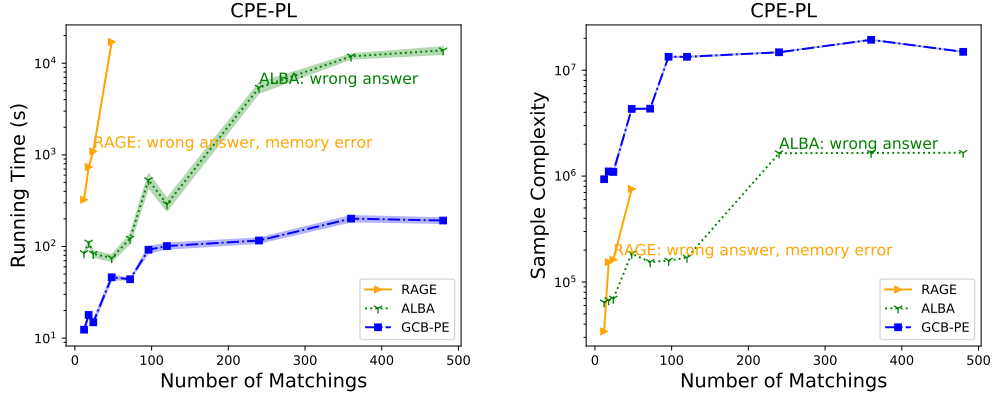


Figure 6.2: Experimental results of running time and sample complexity for CPE-PL.

Task assignments in crowdsourcing. Consider that an employer wishes to assign crowdworkers to tasks with high quality performance, and it wants to avoid the high cost and the privacy concern of collecting each individual worker-task pair performance [Lin et al., 2014]. Thus, the employer sequentially chooses an assignment from N workers to M tasks (each action x is a worker-task matching) and only collects the sum of performance feedback for $1 \leq s < N$ matched worker-task pairs, i.e., M_x contains a single row with 1s in the places of s matched pairs and 0 everywhere else. The objective is to find the best worker-task matching as soon as possible. For $1 \leq s < N$, Lin et al. [2014] provide a systematic method to construct a global observer set.

6.6 Experiments

We conduct experiments CPE-PL on the matching and compare our algorithm with existing BAI-LB algorithms, RAGE by Fiez et al. [2019] and ALBA by Tao et al. [2018] in both running time and sample complexity. We evaluate all the compared algorithms on Intel Xeon E5-2640 v3 CPU at 2.60GHz with 132GB RAM.

As in Chapter 4, we set action space \mathcal{X} as matchings in 3-by-3, 4-by-4 and 5-by-5 complete bipartite graphs. The dimension d , i.e. the number of edges, is set from 9 to 25. The number of matchings $|\mathcal{X}|$ are set from 12 to 480. $\theta_1, \dots, \theta_d$ is set as a geometric sequence in $[0, 1]$. We simulate the random feedback for action x by a Gaussian distribution with mean of $x^\top \theta$ and unit variance. In our experiments, we use the full-bandit feedback ($M_x = x^\top$) but a nonlinear reward function $\bar{r}(x, \theta) = x^\top \theta / \|x\|_1$. For each algorithm, we perform 20 independent runs and present the average running time and sample complexity with 95% confidence intervals across runs. In the experiments, RAGE reports memory errors when $|\mathcal{X}| > 48$ due to its heavy memory burden, and thus we only obtain its results on small- $|\mathcal{X}|$ instances.

We compare GCB-PE with BAI-LB algorithms ALBA and RAGE in running time and sample complexity on a more challenging nonlinear reward task. The result is reported in Figure 6.2. In the experiments, ALBA and RAGE returned wrong answers because they are not designed to handle nonlinear reward func-

tions. Nevertheless, we can still analyze the running times presented in Figure 6.2. It shows that our GCB-PE runs two orders of magnitude faster than ALBA and RAGE while reporting the correct answer. In addition, as $|\mathcal{X}|$ increases, the running time of GCB-PE increases in a much slower pace than the others. The experimental results demonstrate the capability of GCB-PE to simultaneously deal with combinatorial action space, nonlinear reward function and partial feedback in a computationally efficient way.

6.7 Proofs of Theoretical Results

6.7.1 Proof of Lemma 6.1

Proof. We prove inequality $\|\hat{\theta}_n - \theta\|_2 \leq \beta_\sigma$ in Section 6.3 using similar techniques in Lin et al. [2014].

Recall that in the GCB-PE algorithm (Algorithm 6.1), $\hat{\theta}_n$ is the estimate of the environment vector θ in the n -th exploration round. For any n ,

$$\begin{aligned} \|\hat{\theta}_n - \theta\|_2 &= \|M_\sigma^+ \vec{y}_n - M_\sigma^+ M_\sigma \theta\|_2 \\ &= \|M_\sigma^+ \cdot [M_{x_1} \eta_1; \dots; M_{x_{|\sigma|}} \eta_{|\sigma|}]\|_2 \\ &= \left\| (M_\sigma^\top M_\sigma)^{-1} \sum_{i=1}^{|\sigma|} M_{x_i}^\top M_{x_i} \eta_i \right\|_2 \\ &\leq \max_{\eta_1, \dots, \eta_{|\sigma|} \in [-1, 1]^d} \left\| (M_\sigma^\top M_\sigma)^{-1} \sum_{i=1}^{|\sigma|} M_{x_i}^\top M_{x_i} \eta_i \right\|_2 \\ &= \beta_\sigma. \end{aligned}$$

□

6.7.2 Proof of Theorem 6.1

In order to prove Theorem 6.1, we first present the following three lemmas, Lemma 6.2-6.4.

Lemma 6.2. *For Algorithm 6.1, after n exploration rounds,*

$$\Pr[\|\theta - \hat{\theta}(n)\|_2 \geq \text{rad}_n] \leq \frac{\delta}{2n^2}.$$

Proof. In Lemma A.3 in Lin et al. [2014], let $\gamma = \text{rad}_n$. Then, we have

$$\begin{aligned} \Pr[\|\theta - \hat{\theta}(n)\|_2 \geq \text{rad}_n] &\leq 2e^2 \exp \left\{ -\frac{n}{2\beta_\sigma^2} \cdot \frac{2\beta_\sigma^2 \log(\frac{4n^2 e^2}{\delta})}{n} \right\} \\ &= \frac{\delta}{2n^2}. \end{aligned}$$

□

Define the following events

$$\mathcal{E}_n := \{\forall x \in \mathcal{X}, |\bar{r}(x, \theta) - \bar{r}(x, \hat{\theta}(n))| < L_p \cdot \text{rad}_n\}, \forall n \geq 1$$

$$\mathcal{E} := \bigcap_{n=1}^{\infty} \mathcal{E}_n.$$

Lemma 6.3. *We have that $\Pr[\mathcal{E}] \geq 1 - \delta$.*

Proof. From the continuity of the expected reward function,

$$\bar{r}(x, \theta) - \bar{r}(x, \hat{\theta}(n)) < L_p \cdot \|\theta - \hat{\theta}(n)\|_2.$$

From Lemma 6.2, we have that with probability at least $1 - \frac{\delta}{2n^2}$,

$$\|\theta - \hat{\theta}(n)\|_2 < \text{rad}_n.$$

Thus, with probability at least $1 - \frac{\delta}{2n^2}$,

$$\bar{r}(x, \theta) - \bar{r}(x, \hat{\theta}(n)) < L_p \cdot \text{rad}_n.$$

In other words,

$$\Pr[\mathcal{E}_n] \geq 1 - \frac{\delta}{2n^2}.$$

Thus, we have

$$\begin{aligned} \Pr[\mathcal{E}] &= 1 - \Pr[\bar{\mathcal{E}}] \\ &\geq 1 - \sum_{n=1}^{\infty} \Pr[\bar{\mathcal{E}}_n] \\ &\geq 1 - \sum_{n=1}^{\infty} \frac{\delta}{2n^2} \\ &\geq 1 - \delta. \end{aligned}$$

□

Lemma 6.4. *Suppose that \mathcal{E} occurs. If $\text{rad}_n < \frac{\Delta_{\min}}{4L_p}$, Algorithm 6.1 will terminate.*

Proof. Suppose that \mathcal{E} occurs. From the definition of \mathcal{E} , we have

$$\begin{aligned} \bar{r}(\hat{x}, \hat{\theta}(n)) - \bar{r}(\hat{x}^-, \hat{\theta}(n)) &> \bar{r}(\hat{x}, \theta) - \bar{r}(\hat{x}^-, \theta) - 2L_p \cdot \text{rad}_n \\ &= \Delta_{\min} - 2L_p \cdot \text{rad}_n \\ &> 2L_p \cdot \text{rad}_n. \end{aligned}$$

Thus, the stop condition holds, and then Algorithm 6.1 will terminate. \square

Now we prove Theorem 6.1.

Proof. First, we prove the correctness of Algorithm 6.1. From the stop condition, we have that when Algorithm 6.1 terminates, for all $x \in \mathcal{X} \setminus \{\hat{x}\}$,

$$\bar{r}(\hat{x}, \hat{\theta}(n)) - \bar{r}(x, \hat{\theta}(n)) > 2L_p \cdot \text{rad}_n.$$

Then, conditioning on \mathcal{E} , when Algorithm 6.1 terminates, for all $x \in \mathcal{X} \setminus \{\hat{x}\}$,

$$\begin{aligned} \bar{r}(\hat{x}, \theta) &> \bar{r}(\hat{x}, \hat{\theta}(n)) - L_p \cdot \text{rad}_n \\ &> \bar{r}(x, \hat{\theta}(n)) + L_p \cdot \text{rad}_n \\ &> \bar{r}(x, \theta), \end{aligned}$$

which complete the proof of correctness.

Next, we prove the sample complexity of Algorithm 6.1. Let N denote the total number of the exploration rounds. If $N = 1$, Theorem 6.1 trivially holds. In the If $N > 1$, from Lemma 6.4, we have that after $N - 1$ exploration rounds,

$$\sqrt{\frac{2\beta_\sigma^2 \log(\frac{4(N-1)^2 e^2}{\delta})}{N-1}} \geq \frac{\Delta_{\min}}{4L_p}.$$

That is, we have

$$N \leq \frac{32\beta_\sigma^2 L_p^2}{\Delta_{\min}^2} \log\left(\frac{4N^2 e^2}{\delta}\right) + 1$$

Let $\tilde{H} := \frac{\beta_\sigma^2 L_p^2}{\Delta_{\min}^2}$. In the following, we prove $N \leq 655\tilde{H} \log(\frac{\tilde{H}}{\delta})$. We can write $N = C\tilde{H} \log(\frac{\tilde{H}}{\delta})$ for some $C > 0$. In order to prove the theorem, it suffices to prove $C \leq 655$. Suppose, on the contrary, that $C > 655$. Then, we have

$$\begin{aligned} N &\leq 32\tilde{H} \log\left(\frac{4N^2 e^2}{\delta}\right) + 1 \\ &= 64\tilde{H} \log\left(\frac{2eC\tilde{H} \log \frac{\tilde{H}}{\delta}}{\delta}\right) + 1 \\ &\leq 64\tilde{H} \log(2eC) + 64\tilde{H} \log \frac{\tilde{H}}{\delta} + 64\tilde{H} \log\left(\log \frac{\tilde{H}}{\delta}\right) + \tilde{H} \log \frac{\tilde{H}}{\delta} \\ &\leq 64\tilde{H} \log(2eC) + 129\tilde{H} \log \frac{\tilde{H}}{\delta} \\ &< C\tilde{H} \log \frac{\tilde{H}}{\delta} \\ &= N, \end{aligned}$$

which makes a contradiction. Thus, we get

$$N \leq 655\tilde{H} \log \frac{\tilde{H}}{\delta}.$$

Since an exploration round contains $|\sigma| \leq n$ actions, the total number of samples is bounded as follows.

$$T = |\sigma| \cdot N \leq \frac{655\beta_\sigma^2 L_p^2}{\Delta_{\min}^2} \log \left(\frac{\beta_\sigma^2 L_p^2}{\Delta_{\min}^2 \delta} \right).$$

□

6.8 Conclusion

In this chapter, we have proposed a novel model of combinatorial pure exploration with partial-linear feedback, which finds various applications such as recommendation systems and crowdsourcing. For CPE-PL, we have designed a general polynomial-time algorithmic framework **GCB-PE** with sample complexity analysis. Our algorithm and its analysis can be applied to any nonlinear reward function with Lipschitz continuity. The experimental results have show the superiority of our algorithm over the existing ones in speed, and have demonstrated that **GCB-PE** is the first algorithm which can simultaneously deal with combinatorial action space, partial feedback and nonlinear reward functions.

Chapter 7

Conclusion and Future Directions

In this chapter, we conclude the thesis. Specifically, we summarize the contribution of the thesis and discuss future directions.

7.1 Summary

This thesis has been devoted to investigating the learnability of the stochastic combinatorial pure exploration problems with limited feedback. Our main goal was to develop polynomial-time algorithms for such combinatorial pure exploration problems with theoretical guarantee, while most of the existing algorithms require the exponential time (e.g. [Soare et al., 2014, Karnin, 2016, Xu et al., 2018, Tao et al., 2018, Fiez et al., 2019, Degenne et al., 2020, Katz-Samuels et al., 2020]) or strong feedback (e.g. [Bubeck et al., 2013, Cao et al., 2015, Gabillon et al., 2012, 2011b, Kalyanakrishnan and Stone, 2010, Kalyanakrishnan et al., 2012, Kaufmann and Kalyanakrishnan, 2013, Roy Chaudhuri and Kalyanakrishnan, 2017, Chaudhuri and Kalyanakrishnan, 2019, Zhou et al., 2014, Cao and Krishnamurthy, 2019, Chen et al., 2014, 2016a, 2017, Gabillon et al., 2016]). Our results provide novel insights into how to solve online decision making problems in combinatorial action spaces only with limited observation.

In Chapter 3, the problem of top- k arm identification with full-bandit feedback has been discussed. We have designed a novel approximation algorithm with theoretical guarantee for a 0-1 quadratic programming problem arising in confidence ellipsoid maximization. Based on our approximation algorithm, we have proposed the (ϵ, δ) -probably approximately correct (PAC) algorithms **SAQM** that run in $O(\log K)$ time and provided an upper bound of the sample complexity, which is still worst-case optimal; the result indicates that our algorithm provided an exponential speedup over exhaustive search algorithms while keeping the statistical efficiency. We have also designed two heuristic algorithms that empirically perform well: **SA-FOA** using first-order approximation and **CLUCEB-QM** based on lower-upper confidence bound algorithm. Experimental results have demonstrated the superiority of our algorithms in terms of both the computation time and the sample complexity.

In Chapter 4, the problem of combinatorial pure exploration with full-bandit feedback (CPE-BL) has been discussed. We have designed the first adaptive algorithm **PolyALBA** that runs in polynomial time with the aid of G-optimal design. We have shown that the sample complexity of **PolyALBA** achieves the

worst-case optimality for a family of instances. The proposed method empirically performed better than a static algorithm proposed in Chapter 3 for the top- k case in terms of sample complexity, and ran much faster than some of existing BAI-LB algorithms for both top- k and matching cases.

In Chapter 5, the problem of online dense subgraph discovery with linear feedback has been discussed. We have proposed an (ϵ, δ) -PAC algorithm called DS-Lin, and provided a polynomial sample complexity guarantee. Our key technique was to utilize an approximation algorithm using semidefinite programming (SDP) for confidence bound maximization. Then, to deal with large-sized graphs, we have proposed an algorithm called DS-SR by combining the successive reject strategy and the greedy peeling algorithm. We have provided an upper bound of the probability that the quality of the solution obtained by the algorithm is less than $\frac{1}{2}\text{OPT} - \epsilon$, where OPT is the optimal value. Computational experiments using well-known real-world graphs have demonstrated the effectiveness of the proposed algorithm.

In Chapter 6, the problem of combinatorial pure exploration with partial-linear feedback has been discussed. This problem was highly challenging since we should deal with partial-linear feedback and nonlinear reward functions simultaneously in combinatorial action spaces and no existing (even exponential-time) algorithms can solve it. We have designed the general polynomial-time algorithmic framework GCB-PE with sample complexity analysis. Our algorithm and its analysis can be applied to any nonlinear reward function with Lipschitz continuity. The experimental results have shown the superiority of our algorithm in speed, and demonstrated that GCB-PE is the first algorithm which can simultaneously deal with combinatorial action space, partial feedback and nonlinear reward functions.

7.2 Future Directions

In the final section of this thesis, we discuss limitations of the current results and show important subjects for the future. While this thesis has focused on the challenges of combinatorial action spaces and limited observation, there are many more, equally important theoretical or real-world challenges that need to be addressed as follows.

7.2.1 Lower Bounds of Polynomial-time δ -PAC Algorithms

The optimality of the presented sample complexity bounds for full-bandit settings has been discussed by comparing them with the information theoretic lower bounds for the best arm identification in linear bandits (BAI-LB). We have seen the gap between our sample complexity bounds and the information theoretic lower bounds, and we believe that this gap is needed for reducing computational cost. To understand whether or not this gap is inevitable for the problems of combinatorial pure exploration, it is important to investigate lower bounds of polynomial-time δ -PAC algorithms. For the nonlinear rewards or partial-linear feedback, no prior work has provided a lower bound. Therefore, we have the following future work.

Problem 7.1. *Prove a lower bound of polynomial-time algorithms for combinatorial pure exploration problems with full-bandit or partial-linear feedback.*

7.2.2 Non-Stationary Environment

All the studies in this thesis and most existing work focused on the *stationary* case, where the distribution of rewards never changes over time. However, in real-world applications, we are faced with an extremely *non-stationary* world and it is not reasonable to assume that the distribution stays the same [Besbes et al., 2014]. For example, in online advertising and recommendation systems, a user's preferences may likely change when some events happen, which greatly influence the users, and typically exhibit trends on seasonal, weekly, and even daily scales. Therefore, solving the non-stationary case is promising for increasing the applicability of combinatorial pure exploration methods.

Problem 7.2. *Design a method for combinatorial pure exploration problems in non-stationary environments.*

7.2.3 Nonlinear Reward Functions and Nonlinear Feedback

In Chapter 6, the problem of combinatorial pure exploration with partial-linear feedback for nonlinear rewards has been discussed, and a general polynomial-time algorithmic framework has been given. However, our algorithm and its analysis are restricted to nonlinear reward functions that satisfy the Lipschitz continuity, which often fails to hold in practice. We also considered linear feedback in all previous chapters. For a class of reward functions that belong to some reproducing kernel hilbert space (RKHS), Valko et al. [2013] proposed a kernelized UCB algorithm, and provided a cumulative regret bound.

To discuss this issue more precisely, we will first give the definition of RKHS. Let \mathcal{H} be a functional Hilbert space and let \mathcal{D} be the domain of the functions in \mathcal{H} . We use $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ to denote the inner product in a Hilbert space \mathcal{H} . The inner product has the following properties for elements $f, g \in \mathcal{H}$ and a scalar a

1. Linearity: $\langle a_1 f_1 + a_2 f_2, g \rangle_{\mathcal{H}} = a_1 \langle f_1, g \rangle_{\mathcal{H}} + a_2 \langle f_2, g \rangle_{\mathcal{H}}$;
2. Symmetry: $\langle f, g \rangle = \overline{\langle g, f \rangle_{\mathcal{H}}}$;
3. $\langle f, f \rangle_{\mathcal{H}} \geq 0$, $\langle f, g \rangle_{\mathcal{H}} = 0$ if and only if $f = 0$,

where $\bar{\cdot}$ denotes the complex conjugate of a scalar.

The *reproducing kernel* $K(x, x')$ is a bivariate function defined on $\mathcal{D} \times \mathcal{D}$ that satisfies the following conditions.

1. For any fixed $x' \in \mathcal{D}$, $K(x, x')$ belongs to \mathcal{H} as a function of x .
2. It holds that $\langle f(\cdot), K(\cdot, x') \rangle_{\mathcal{H}} = f(x')$ for any $f \in \mathcal{H}$, any $x' \in \mathcal{D}$.

The reproducing kernel has the following properties:

1. $K(x, x') = \overline{K(x', x)}$ for any $x, x' \in \mathcal{D}$;
2. $K(x, x') \geq 0$ for any $x \in \mathcal{D}$.

A Hilbert space that has the reproducing kernel is called a *reproducing kernel Hilbert space*.

In the kernelized linear bandit, we let $\mathcal{D} = \mathbb{R}^d$ and let an orthonormal basis ϕ be the mapping $\phi : \mathbb{R}^d \rightarrow \mathcal{H}$. Then, we define reproducing kernel function as

$$K(x, x') = \phi(x)^\top \phi(x'), \quad \forall x, x' \in \mathbb{R}^d.$$

In the model of Valko et al. [2013], the existence of ϕ is assumed for which there exists $\theta \in \mathcal{H}$ such that the expected reward for $x \in \mathcal{X}$ is expressed by $\phi(x)^\top \theta$. In RKHS, we can utilize the ridge regression in \mathcal{H} to build an estimator for θ . To weaken the assumptions on Lipschitz continuity for reward functions and linear feedback in our model, we may also consider a wide class of reward functions that belong to RKHS. In the literature in BAI-LB, no existing work have considered nonlinear reward functions. Thus we may begin with non-combinatorial settings, where action size $|\mathcal{X}|$ is not so large.

Problem 7.3. *Design a method for the best arm identification in kernelized linear bandits.*

Note that $\max_{x \in \mathcal{X}} \phi(x)^\top \theta$ may be hard since it involves nonconvex optimization. When \mathcal{X} is a combinatorial action space, whether or not there still exists a polynomial-time algorithm for finding the optimal arm $x^* = \operatorname{argmax}_{x \in \mathcal{X}} \phi(x)^\top \theta$ is controversial, since $\phi(x) \in \mathcal{H}$ may not be a 0–1 vector and thus it is not trivial to utilize combinatorial structures. If such a polynomial-time algorithm is not available in RKHS, we may try to find another class of nonlinear reward functions and devise a specific algorithm to solve it.

Problem 7.4. *Devise a method for combinatorial pure exploration with a subclass of nonlinear reward functions.*

7.2.4 Optimal Experimental Design with Combinatorial Actions

To investigate the above future work and further improve our current results, we believe that developing algorithms and theory for the optimal experimental design in combinatorial action spaces is one of the very important topics. Taking an example of the sample complexity bound provided in Chapter 4, we will discuss further directions and a connection between pure exploration problems and optimal experimental design.

G-optimal design. Some of the BAI-LB algorithms (e.g. Soare et al. [2014], Tao et al. [2018], Fiez et al. [2019]) require solving the following G-optimal design problem for their design:

$$\min_{\lambda \in \Delta(\mathcal{X})} \max_{x \in \mathcal{X}} \|x\|_{M(\lambda)^{-1}},$$

where $M(\lambda)^{-1} = \sum_{x \in \mathcal{X}} (\lambda(x) x x^\top)^{-1}$ is the error covariance matrix. The G-optimal design aims to minimize the maximum prediction variance, and as shown in Kiefer and Wolfowitz [1960], the continuous G-optimal design and D-optimal

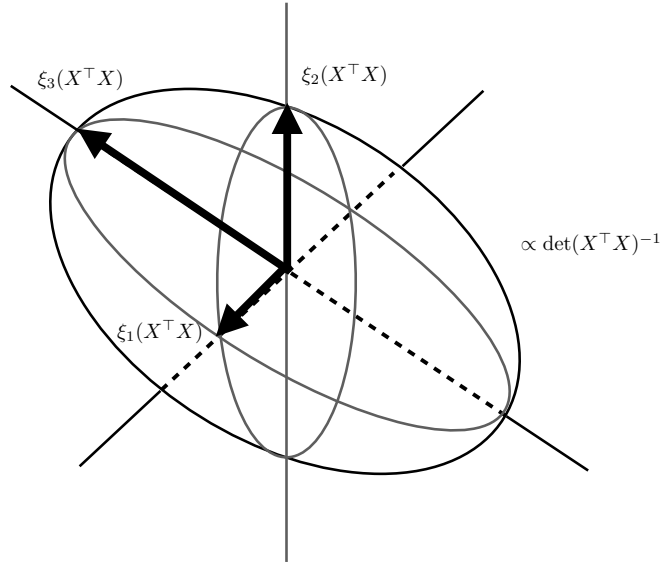


Figure 7.1: The illustration of a confidence ellipsoid, where $X = (x_1, \dots, x_d)$ is the model matrix. The shape of the ellipsoid depends on the information matrix $X^\top X$.

design are equivalent when the errors are homoscedastic. Geometrically, this corresponds to designing the experiment to minimize the volume of the resulting confidence ellipsoid [Boyd and Vandenberghe, 2004] (see Figure 7.1).

The above G-optimal design problem is hard to compute when $|\mathcal{X}|$ is exponentially large, since we have an exponential number of variables and the inner optimization is already hard as discussed in Chapter 3. To avoid the high computational cost, the proposed algorithm in Chapter 4 chose a set of super arms $\mathcal{X}_\sigma \subseteq \mathcal{X}$ (the proposed algorithms in Chapters 3 and 5 also did so). Then we can obtain

$$\lambda_\sigma^* = \operatorname{argmin}_{\lambda \in \Delta(\mathcal{X}_\sigma)} \max_{x \in \mathcal{X}_\sigma} \|x\|_{M(\lambda)^{-1}},$$

and guarantee that

$$\max_{x \in \mathcal{X}_\sigma} \|x\|_{M(\lambda_\sigma^*)^{-1}} = d,$$

using the theorem in Kiefer and Wolfowitz [1960] (see also Proposition 4.1 in Chapter 4). Let $m = \max_{x \in \mathcal{X}} \|x\|_2^2$. In Lemma 4.2 in Chapter 4, we have already seen that for any $z \in \mathcal{X}$, we have

$$\|z\|_{M(\lambda_\sigma^*)^{-1}} \leq d \sqrt{m \cdot \xi_{\max}(\widetilde{M}(\lambda_{\mathcal{X}_\sigma}^*)^{-1})},$$

where recall that $\widetilde{M}(\lambda_{\mathcal{X}_\sigma}^*) = \sum_{x \in \operatorname{supp}(\lambda_{\mathcal{X}_\sigma}^*)} (xx^\top)$. Due to this upper bound, our sample complexity depends on $\xi_{\max}(\widetilde{M}(\lambda_{\mathcal{X}_\sigma}^*)^{-1})$. Therefore, to reduce the sample complexity, we want to choose a subset $\mathcal{X}_\sigma \subseteq \mathcal{X}$ which minimizes the largest eigenvalue $\xi_{\max}(\widetilde{M}(\lambda_{\mathcal{X}_\sigma}^*)^{-1})$. However, such a selection of good super arms corresponds to the *E-optimal design*, i.e., the goal is to minimizing the largest eigenvalue of the error covariance matrix. This problem is no easier than G-optimal design, which will be discussed in the next paragraph. We note that in general, it seems hard to give an upper bound of the largest eigenvalue of

the error covariance matrix as the following example for the top- k case implies. Therefore, we wish to design some efficient methods for the E-optimal design and achieve a better sample complexity.

Example 7.1. We assume that d and k in a given instance of the top- k identification problem are coprime. For $i \in [d]$, let the $((i + n) \bmod d)$ -th element of $x_i \in \mathcal{X}$ be 1 for $n = 1, 2, \dots, k$ and other elements of x_i be 0. Let $\mathcal{X}_\sigma = \{x_1, x_2, \dots, x_d\} \subseteq \mathcal{X}$. Then the information matrix $\sum_{x \in \mathcal{X}_\sigma} (xx^\top)$ is defined as a following circulant matrix C :

$$C = \begin{bmatrix} c_0 & c_{d-1} & \dots & c_2 & c_1 \\ c_1 & c_0 & c_{d-1} & & c_2 \\ \vdots & c_1 & c_0 & \ddots & \vdots \\ c_{d-2} & & \ddots & \ddots & c_{d-1} \\ c_{d-1} & c_{d-2} & \dots & c_1 & c_0 \end{bmatrix}, \text{ where } \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{k-1} \\ c_k \\ \vdots \\ c_{d-2k} \\ c_{d-2k+1} \\ \vdots \\ c_{d-2} \\ c_{d-1} \end{bmatrix} = \begin{bmatrix} k \\ k-1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ k-2 \\ k-1 \end{bmatrix}.$$

The corresponding eigenvalues are then given by

$$\xi_j = c_0 + c_{d-1}\omega_j + c_{d-2}\omega_j^2 + \dots + c_1\omega_j^{n-1}, \quad j = 0, 1, \dots, d-1,$$

where $\omega_j = \exp(i \frac{2\pi j}{d})$ are the d -th root of unity and i is the imaginary unit.

E-optimal design. The E-optimal design can be interpreted geometrically as minimizing the diameter of the confidence ellipsoids. Obtaining the E-optimal (discrete) design is known to be NP-hard [Çivril and Magdon-Ismail, 2009], and the best known result is $(1+\epsilon)$ -approximation by Allen-Zhu et al. [2017] for $\epsilon > 0$. We remark that existing methods assume $|\mathcal{X}|$ to be small, while $|\mathcal{X}| = O(2^d)$ in our setting [Allen-Zhu et al., 2017, Madan et al., 2019, Pukelsheim, 2006].

In the rest of this section, let L be the number of non-zero elements of $\lambda \in \{0, 1\}^{|\mathcal{X}|}$. The E-optimal design problem with combinatorial action space \mathcal{X} can be casted as follows.

$$\begin{aligned} \text{Primal: minimize } & \lambda_{\max} \left(\sum_{x_i \in \mathcal{X}} \lambda_i x_i x_i^\top \right)^{-1} \\ \text{subject to } & \lambda \in \{0, 1\}^{|\mathcal{X}|}, \quad \sum_{x_i \in \mathcal{X}} \lambda_i \leq L. \end{aligned}$$

Relaxing the discrete variables $\lambda \in \{0, 1\}^{|\mathcal{X}|}$ to continuous one and taking the dual of the relaxation problem, we will obtain the following dual problem.

$$\begin{aligned} \text{Dual 1: maximize } & \text{Tr}(W) \\ \text{subject to } & x_i^\top W x_i \leq L, \quad \forall x_i \in \mathcal{X} \\ & W \succeq 0, \end{aligned}$$

with variable $W \in \mathbb{S}_+^d$, where \mathbb{S}_+^d denotes the set of symmetric positive semidefinite $d \times d$ matrices. The optimal solution W^* determines the minimum volume ellipsoid, centered at the origin, given by $\{x : x^\top W x \leq L\}$, that contains the points $x_1, \dots, x_{|\mathcal{X}|}$. Due to complementary slackness,

$$\lambda_i^*(L - x_i^\top W^* x_i) = 0 \text{ for } i = 1, \dots, |\mathcal{X}|.$$

From the above condition, the optimal design only uses x_i that lies on the surface of the ellipsoid defined by W^* . Notice, however, that this dual problem still has an exponentially large number of constraints. We may consider a further relaxation problem. Let $P(\mathcal{X})$ be a polytope, which is a convex hull of the indicator vectors of all super arms.

$$\begin{aligned} \text{Dual 2: maximize } & (W) \\ \text{subject to } & x^\top W x \leq L, \quad \forall x \in P(\mathcal{X}) \\ & W \succeq 0, \end{aligned}$$

with variable $W \in \mathbb{S}_+^d$. We can investigate how to solve this problem for specific cases such as matroids or matchings and design a rounding procedure with theoretical guarantee, or take a totally different approach.

To sum up, important future work in this line is as follows.

Problem 7.5. *Devise a method for optimal experimental design with combinatorial action spaces.*

Bibliography

- Top arm identification in multi-armed bandits with batch arm pulls, author = Kwang-Sung Jun and Kevin Jamieson and Robert Nowak and Xiaojin Zhu, booktitle = Proc. AISTATS'16, pages = 139–148, year = 2016.
- Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári. Improved algorithms for linear stochastic bandits. In *Proc. NIPS'14*, pages 2312–2320, 2011.
- J. D. Abernethy, E. Hazan, and A. Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *Proc. COLT'08*, 2008.
- E. Adar and C. Ré. Managing uncertainty in social networks. *IEEE Data Engineering Bulletin*, 30:15–22, 2007.
- R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proc. SIGMOD'00*, pages 439–450, 2000.
- R. Agrawal, M. Hegde, and D. Teneketzis. Multi-armed bandit problems with multiple plays and switching cost. *Stochastics and Stochastic Reports*, 29: 437–459, 1990.
- Z. Allen-Zhu, Y. Li, A. Singh, and Y. Wang. Near-optimal design of experiments via regret minimization. In *Proc. ICML'17*, pages 126–135, 2017.
- U. Alon. Biological networks: The tinkerer as an engineer. *Science*, 301:1866–1867, 2003.
- V. Anantharam, P. Varaiya, and J. Walrand. Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays-Part I: I.I.D. rewards. *IEEE Transactions on Automatic Control*, 32:968–976, 1987.
- R. Andersen and K. Chellapilla. Finding dense subgraphs with size bounds. In *Proc. WAW'09*, pages 25–37, 2009.
- A. Angel, N. Sarkas, N. Koudas, and D. Srivastava. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. In *Proc. VLDB'12*, pages 574–585, 2012.
- Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama. Greedily finding a dense subgraph. *Journal of Algorithms*, 34(2):203–221, 2000.
- J.-Y. Audibert, S. Bubeck, and R. Munos. Best arm identification in multi-armed bandits. In *Proc. COLT'10*, pages 41–53, 2010.

- P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2003.
- G. D. Bader and C. W. V. Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4(1): 1–27, 2003.
- B. Bahmani, R. Kumar, and S. Vassilvitskii. Densest subgraph in streaming and mapreduce. In *Proc. VLDB’12*, pages 454–465, 2012.
- M. S. Bartlett, G. Littlewort, M. Frank, C. Lainscsek, I. Fasel, and J. Movellan. Recognizing facial expression: machine learning and application to spontaneous behavior. In *Proc. of CVPR’05*, volume 2, pages 568–573, 2005.
- A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust optimization*, volume 28. Princeton University Press, 2009.
- D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.
- D. Bertsimas, D. B. Brown, and C. Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501, 2011. doi: 10.1137/080734510.
- O. Besbes, Y. Gur, and A. Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. In *Proc. NIPS’14*, pages 199–207, 2014.
- A. Bhaskara, M. Charikar, E. Chlamtac, U. Feige, and A. Vijayaraghavan. Detecting high log-densities: An $O(n^{1/4})$ approximation for densest k -subgraph. In *Proc. STOC’10*, pages 201–210, 2010.
- S. Bhattacharya, M. Henzinger, D. Nanongkai, and C. E. Tsourakakis. Space- and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *Proc. STOC’15*, pages 173–182, 2015.
- M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.
- M. Bouhtou, S. Gaubert, and G. Sagnol. Submodularity and randomized rounding techniques for optimal experimental design. *Electronic Notes in Discrete Mathematics*, 36:679–686, 2010.
- S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, USA, 2004.
- A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. *Computer Networks*, 33:309 – 320, 2000.
- S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5:1–122, 2012.

- S. Bubeck, R. Munos, and G. Stoltz. Pure exploration in finitely-armed and continuous-armed bandits. *Theoretical Computer Science*, 412(19):1832 – 1852, 2011.
- S. Bubeck, T. Wang, and N. Viswanathan. Multiple identifications in multi-armed bandits. In *Proc. ICML’13*, pages 258–265, 2013.
- A. Çivril and M. Magdon-Ismail. On selecting a maximum volume sub-matrix of a matrix and related problems. *Theoretical Computer Science*, 410(47):4801 – 4811, 2009. ISSN 0304-3975.
- T. Cao and A. Krishnamurthy. Disagreement-based combinatorial pure exploration: Sample complexity bounds and an efficient algorithm. In *Proc. COLT’19*, pages 558–588, 2019.
- W. Cao, J. Li, Y. Tao, and Z. Li. On top-k selection in multi-armed bandits and hidden bipartite graphs. In *Proc. NIPS’15*, pages 1036–1044, 2015.
- A. Carpentier and A. Locatelli. Tight (lower) bounds for the fixed budget best arm identification bandit problem. In *Proc. COLT’16*, pages 590–604, 2016.
- N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.
- N. Cesa-Bianchi and G. Lugosi. Combinatorial bandits. *Journal of Computer and System Sciences*, 78:1404–1422, 2012.
- M. Charikar. Greedy approximation algorithms for finding dense components in a graph. In *Proc. APPROX’00*, pages 84–95, 2000.
- A. R. Chaudhuri and S. Kalyanakrishnan. PAC identification of many good arms in stochastic multi-armed bandits. In *Proc. ICML’19*, pages 991–1000, 2019.
- S. Chaudhuri and A. Tewari. Online ranking with top-1 feedback. In *Proc. AIS-TATS’15*, pages 129–137, 2015.
- S. Chaudhuri and A. Tewari. Phased exploration with greedy exploitation in stochastic combinatorial partial monitoring games. In *Proc. NIPS’16*, pages 2433–2441. 2016.
- S. Chaudhuri and A. Tewari. Online learning to rank with top-k feedback. *The Journal of Machine Learning Research*, 18(1):3599–3648, 2017.
- L. Chen and J. Li. On the optimal sample complexity for best arm identification. *arXiv preprint*, arXiv:1511.03774, 2015.
- L. Chen, A. Gupta, and J. Li. Pure exploration of multi-armed bandit under matroid constraints. In *Proc. COLT’16*, pages 647–669, 2016a.
- L. Chen, A. Gupta, J. Li, M. Qiao, and R. Wang. Nearly optimal sampling algorithms for combinatorial pure exploration. In *Proc. COLT’17*, pages 482–534, 2017.

- S. Chen, T. Lin, I. King, M. R. Lyu, and W. Chen. Combinatorial pure exploration of multi-armed bandits. In *Proc. NIPS'14*, pages 379–387, 2014.
- W. Chen, Y. Wang, and Y. Yuan. Combinatorial multi-armed bandit: General framework and applications. In *Proc. ICML'13*, pages 151–159, 2013.
- W. Chen, T. Lin, Z. Tan, M. Zhao, and X. Zhou. Robust influence maximization. In *Proc. KDD'16*, pages 795–804, 2016b.
- W. Chen, Y. Wang, Y. Yuan, and Q. Wang. Combinatorial multi-armed bandit and its extension to probabilistically triggered arms. *Journal of Machine Learning Research*, 17(50):1–33, 2016c.
- W. Chen, Y. Du, L. Huang, and H. Zhao. Combinatorial pure exploration of dueling bandit. *to appear in ICML'20*, 2020.
- R. Combes, M. S. Talebi Mazraeh Shahi, A. Proutiere, and M. Lelarge. Combinatorial bandits revisited. In *Proc. NIPS'15*, pages 2116–2124, 2015.
- V. Dani, T. P. Hayes, and S. M. Kakade. Stochastic linear optimization under bandit feedback. In *Proc. COLT'08*, pages 355–366, 2008.
- R. Degenne, P. Ménard, X. Shang, and M. Valko. Gamification of pure exploration for linear bandits. *to appear in ICML'20*, 2020.
- A. Delarue, R. Anderson, and C. Tjandraatmadja. Reinforcement learning with combinatorial actions: An application to vehicle routing. *to appear in NeurIPS'20*, 2020.
- E. W. Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1:269–271, 1959.
- Y. Dourisboure, F. Geraci, and M. Pellegrini. Extraction and classification of dense communities in the web. In *Proc. WWW'07*, pages 461–470, 2007.
- J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17: 449–467, 1965. doi: 10.4153/CJM-1965-045-4.
- A. Epasto, S. Lattanzi, and M. Sozio. Efficient densest subgraph computation in evolving graphs. In *Proc. WWW'15*, pages 300–310, 2015.
- E. Even-Dar, S. Mannor, and Y. Mansour. PAC bounds for multi-armed bandit and Markov decision processes. In *Proc. COLT'02*, pages 255–270, 2002.
- E. Even-Dar, S. Mannor, and Y. Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of Machine Learning Research*, 7:1079–1105, 2006.
- U. Feige, D. Peleg, and G. Kortsarz. The dense k -subgraph problem. *Algorithmica*, 29(3):410–421, 2001.
- T. Fiez, L. Jain, K. G. Jamieson, and L. Ratliff. Sequential experimental design for transductive linear bandits. In *Proc. NeurIPS'19*, pages 10667–10677, 2019.

- V. Gabillon, M. Ghavamzadeh, A. Lazaric, and S. Bubeck. Multi-bandit best arm identification. In *Proc. NIPS'11*, pages 2222–2230, 2011a.
- V. Gabillon, M. Ghavamzadeh, A. Lazaric, and S. Bubeck. Multi-bandit best arm identification. In *Proc. NIPS'11*, pages 2222–2230, 2011b.
- V. Gabillon, M. Ghavamzadeh, and A. Lazaric. Best arm identification: A unified approach to fixed budget and fixed confidence. In *Proc. NIPS'12*, pages 3212–3220, 2012.
- V. Gabillon, A. Lazaric, M. Ghavamzadeh, R. Ortner, and P. Bartlett. Improved learning complexity in combinatorial pure exploration bandits. In *Proc. AIS-TATS'16*, pages 1004–1012, 2016.
- E. Galimberti, F. Bonchi, and F. Gullo. Core decomposition and densest subgraph in multilayer networks. In *Proc. CIKM'17*, pages 1807–1816, 2017.
- M. Ghaffari, S. Lattanzi, and S. Mitrović. Improved parallel algorithms for density-based network clustering. In *Proc. ICML'19*, pages 2201–2210, 2019.
- D. Gibson, R. Kumar, and A. Tomkins. Discovering large dense subgraphs in massive graphs. In *Proc. VLDB'05*, pages 721–732, 2005.
- A. Gionis and C. E. Tsourakakis. Dense subgraph discovery: KDD 2015 tutorial. In *Proc. KDD'15*, pages 2313–2314, 2015.
- M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42:1115–1145, 1995.
- A. V. Goldberg. Finding a maximum density subgraph. Technical report, University of California Berkeley, 1984.
- M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.
- S. Hu, X. Wu, and T.-H. H. Chan. Maintaining densest subsets efficiently in evolving hypergraphs. In *Proc. CIKM'17*, pages 929–938, 2017.
- S. Huang, X. Liu, and Z. Ding. Opportunistic spectrum access in cognitive radio networks. In *Proc. INFOCOM'08*, pages 1427–1435, 2008.
- W. Huang, J. Ok, L. Li, and W. Chen. Combinatorial pure exploration with continuous and separable reward functions and its applications. In *Proc. IJ-CAI'18*, pages 2291–2297, 2018.
- K. Jamieson, M. Malloy, R. Nowak, and S. Bubeck. lil'UCB: An optimal exploration algorithm for multi-armed bandits. In *Proc. COLT'14*, pages 423–439, 2014.
- K. Jun, A. Bhargava, R. Nowak, and R. Willett. Scalable generalized linear bandits: Online computation and hashing. In *Proc. NIPS'17*, pages 99–109, 2017.

- S. Kalyanakrishnan and P. Stone. Efficient selection of multiple bandit arms: Theory and practice. In *Proc. ICML'10*, pages 511–518, 2010.
- S. Kalyanakrishnan, A. Tewari, P. Auer, and P. Stone. PAC subset selection in stochastic multi-armed bandits. In *Proc. ICML'12*, pages 655–662, 2012.
- D. R. Karger. Random sampling and greedy sparsification for matroid optimization problems. *Mathematical Programming*, 82(1):41–81, 1998.
- Z. S. Karnin. Verification based solution for structured mab problems. In *Proc. NIPS'16*, pages 145–153, 2016.
- J. Katz-Samuels, L. Jain, Z. Karnin, and K. Jamieson. An empirical process approach to the union bound: Practical algorithms for combinatorial and linear bandits. *arXiv preprint arXiv:2006.11685*, 2020.
- E. Kaufmann and S. Kalyanakrishnan. Information complexity in bandit subset selection. In *Proc. COLT'13*, pages 228–251, 2013.
- E. Kaufmann, O. Cappé, and A. Garivier. On the complexity of best-arm identification in multi-armed bandit models. *Journal of Machine Learning Research*, 17:1–42, 2016.
- Y. Kawase and A. Miyauchi. The densest subgraph problem with a convex/concave size function. *Algorithmica*, 80(12):3461–3480, 2018.
- Y. Kawase and H. Sumita. Randomized strategies for robust combinatorial optimization. In *Proc. AAAI'19*, pages 7876–7883, 2019.
- Y. Kawase, Y. Kuroki, and A. Miyauchi. Graph mining meets crowdsourcing: Extracting experts for answer aggregation. In *Proc. IJCAI'19*, pages 1272–1279, 2019.
- S. Khuller and B. Saha. On finding dense subgraphs. In *Proc. ICALP'09*, pages 597–608, 2009.
- J. Kiefer and J. Wolfowitz. The equivalence of two extremum problems. *Canadian Journal of Mathematics*, 12:363–366, 1960.
- J. Komiyama, J. Honda, and H. Nakagawa. Optimal regret analysis of thompson sampling in stochastic multi-armed bandit problem with multiple plays. In *Proc. ICML'15*, pages 1152–1161, 2015.
- B. Korte and J. Vygen. *Combinatorial optimization*, volume 2. Springer, 2012.
- Y. Kuroki, L. Xu, A. Miyauchi, J. Honda, and M. Sugiyama. Polynomial-time algorithms for multiple-arm identification with full-bandit feedback. *Neural Computation*, 32(9):1733–1773, 2020.
- B. Kveton, Z. Wen, A. Ashkan, and C. Szepesvari. Tight Regret Bounds for Stochastic Combinatorial Semi-Bandits. In *Proc. AISTATS'15*, volume 38, pages 535–543, 2015.

- P. Lagr  e, C. Vernade, and O. Cappe. Multiple-play bandits in the position-based model. In *Proc. NIPS'16*, pages 1597–1605, 2016.
- T. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.
- P. Lather. *Getting smart: Feminist research and pedagogy with/in the postmodern*. Psychology Press, 1991.
- T. Lattimore and C. Szepesvari. The End of Optimism? An Asymptotic Analysis of Finite-Armed Linear Bandits. In *Proc. AISTATS'17*, pages 728–737, 2017.
- T. Lattimore and C. Szepesv  ri. *Bandit algorithms*. Cambridge University Press, 2020.
- E. L. Lawler. A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Management Science*, 18(7):401–405, 1972.
- F. Le Gall. Powers of tensors and fast matrix multiplication. In *Proc. ISSAC'14*, pages 296–303, 2014.
- J. Li, Y. Baba, and H. Kashima. Hyper questions: Unsupervised targeting of a few experts in crowdsourcing. In *Proc. CIKM'17*, pages 1069–1078, 2017.
- T. Lin, B. Abrahao, R. Kleinberg, J. Lui, and W. Chen. Combinatorial partial monitoring game with linear feedback and its applications. In *Proc. ICML'14*, pages 901–909, 2014.
- V. Madan, M. Singh, U. Tantipongpipat, and W. Xie. Combinatorial algorithms for optimal design. In *Proc. COLT'19*, volume 99, pages 2210–2258, 2019.
- S. Mahajan and H. Ramesh. Derandomizing approximation algorithms based on semidefinite programming. *SIAM Journal on Computing*, 28:1641–1663, 1999.
- A. McGregor, D. Tench, S. Vorotnikova, and H. T. Vu. Densest subgraph in dynamic graph streams. In *Proc. MFCS'15*, pages 472–482, 2015.
- B. Miller, N. Bliss, and P. Wolfe. Subgraph detection using eigenvector L1 norms. In *Proc. NIPS'10*, 2010.
- M. Mitzenmacher, J. Pachocki, R. Peng, C. E. Tsourakakis, and S. C. Xu. Scalable large near-clique detection in large-scale networks via sampling. In *Proc. KDD'15*, pages 815–824, 2015.
- A. Miyauchi and N. Kakimura. Finding a dense subgraph with sparse cut. In *Proc. CIKM'18*, pages 547–556, 2018.
- A. Miyauchi and A. Takeda. Robust densest subgraph discovery. In *Proc. ICDM'18*, pages 1188–1193, 2018.
- A. Miyauchi, Y. Iwamasa, T. Fukunaga, and N. Kakimura. Threshold influence model for allocating advertising budgets. In *Proc. ICML'15*, pages 1395–1404, 2015.

- V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518:529–533, 2015.
- M. A. U. Nasir, A. Gionis, G. D. F. Morales, and S. Girdzijauskas. Fully dynamic algorithm for top-k densest subgraphs. In *Proc. CIKM’17*, pages 1817–1826, 2017.
- T. Nepusz, H. Yu, and A. Paccanaro. Detecting overlapping protein complexes in protein-protein interaction networks. *Nature Methods*, 9(5):471–472, 2012.
- Y. Nesterov. Semidefinite relaxation and nonconvex quadratic optimization. *Optimization Methods and Software*, 9(1-3):141–160, 1998.
- D. S. Papailiopoulos, I. Mitliagkas, A. G. Dimakis, and C. Caramanis. Finding dense subgraphs via low-rank bilinear optimization. In *Proc. ICML’14*, pages 1890–1898, 2014.
- P. Perrault, V. Perchet, and M. Valko. Exploiting structure of uncertainty for efficient matroid semi-bandits. In *Proc. ICML’19*, pages 5123–5132, 2019.
- F. Pukelsheim. *Optimal Design of Experiments*. Society for Industrial and Applied Mathematics, 2006.
- F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *Proc. ICML’08*, pages 784–791, 2008.
- I. Rejwan and Y. Mansour. Top- k combinatorial bandits with full-bandit feedback. In *Proc. ALT’20*, pages 752–776, 2020.
- D. Retelny, S. Robaszkiewicz, A. To, W. S. Lasecki, J. Patel, N. Rahmati, T. Doshi, M. Valentine, and M. S. Bernstein. Expert crowdsourcing with flash teams. In *Proc. UIST’14*, pages 75–85, 2014.
- H. Robbins. Some aspects of the sequential design of experiments. *Bull. Amer. Math. Soc.*, 58:527–535, 09 1952.
- A. Roy Chaudhuri and S. Kalyanakrishnan. PAC identification of a bandit arm relative to a reward quantile. In *Proc. AAAI’17*, pages 1977–1985, 2017.
- P. Rusmevichientong and D. P. Williamson. An adaptive algorithm for selecting profitable keywords for search-based advertising services. In *Proc. EC’06*, pages 260–269, 2006.
- G. Sagnol. Approximation of a maximum-submodular-coverage problem involving spectral functions, with application to experimental designs. *Discrete Applied Mathematics*, 161:258–276, 2013.
- J. P. Scott and P. J. Carrington. *The SAGE Handbook of Social Network Analysis*. 2011.
- M. Soare, A. Lazaric, and R. Munos. Best-arm identification in linear bandits. In *Proc. NIPS’14*, pages 828–836, 2014.

- M. Sugiyama. *Statistical reinforcement learning: modern machine learning approaches*. CRC Press, 2015.
- R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998.
- M. S. Talebi Mazraeh Shahi. Online combinatorial optimization under bandit feedback (thesis), 2016.
- C. Tao, S. Blanco, and Y. Zhou. Best arm identification in linear bandits with linear dimension dependency. In *Proc. ICML’18*, pages 4877–4886, 2018.
- R. Taylor. Approximation of the quadratic knapsack problem. *Operations Research Letters*, 44(4):495–497, 2016.
- W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25:285–294, 1933.
- L. Tran-Thanh, S. Stein, A. Rogers, and N. R. Jennings. Efficient crowdsourcing of unknown experts using bounded multi-armed bandits. *Artificial Intelligence*, pages 89–111, 2014.
- C. E. Tsourakakis. The k-clique densest subgraph problem. In *Proc. WWW’15*, pages 1122–1132, 2015.
- C. E. Tsourakakis, T. Chen, N. Kakimura, and J. Pachocki. Novel dense subgraph discovery primitives: Risk aversion and exclusion queries. In *Machine Learning and Knowledge Discovery in Databases*, pages 378–394, 2020.
- A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. a correction. *Proceedings of the London Mathematical Society*, 2:544–546, 1938.
- A. M. Turing. Computing Machinery and Intelligence. *Mind*, LIX(236):433–460, 1950.
- M. Valko, N. Korda, R. Munos, I. Flaounas, and N. Cristianini. Finite-time analysis of kernelised contextual bandits. *arXiv preprint arXiv:1309.6869*, 2013.
- V. V. Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.
- Z. Wen, B. Kveton, M. Valko, and S. Vaswani. Online influence maximization under independent cascade model with semi-bandit feedback. In *Proc. NIPS’17*, pages 3022–3032, 2017.
- H. Whitney. On the abstract properties of linear dependence. *American Journal of Mathematics*, 57(3):509–533, 1935.
- L. Xu, J. Honda, and M. Sugiyama. A fully adaptive algorithm for pure exploration in linear bandits. In *Proc. AISTATS’18*, pages 843–851, 2018.

- Y. Ye. Approximating quadratic programming with bound constraints. *Mathematical Programming*, 84:219–226, 1997.
- M. Zaki, A. Mohan, and A. Gopalan. Towards optimal and efficient best arm identification in linear bandits. *arXiv preprint arXiv:1911.01695*, 2019.
- M. Zaki, A. Mohan, and A. Gopalan. Explicit best arm identification in linear bandits using no-regret learners. *arXiv preprint arXiv:2006.07562*, 2020.
- E. Zheleva and L. Getoor. Privacy in social networks: A survey. In *Social Network Data Analytics*, pages 277–306. Springer US, 2011.
- Z. Zhong, W. C. Cheung, and V. Y. Tan. Best arm identification for cascading bandits in the fixed confidence setting. *to appear in ICML’20*, 2020.
- Y. Zhou, X. Chen, and J. Li. Optimal PAC multiple arm identification with applications to crowdsourcing. In *Proc. ICML’14*, pages 217–225, 2014.
- Z. Zou. Polynomial-time algorithm for finding densest subgraphs in uncertain graphs. In *Proc. MLG’13*, 2013. No page numbers.