Learning to Route Efficiently with End-to-End Feedback: The Value of (Identifiable) Networked Structure

by

Ruihao Zhu

B.Eng Electrical and Computer Engineering, Shanghai Jiao Tong University, 2015
B.Eng Computer Science and Engineering, University of Michigan, 2015

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2018

© 2018 Massachusetts Institute of Technology. All rights reserved.

Author .....
Department of Aeronautics and Astronautics
August 23, 2018

Certified by .....
Eytan Modiano
Professor, Department of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Hamsa Balakrishnan
Associate Professor, Department of Aeronautics and Astronautics
Chair, Graduate Program Committee

Learning to Route Efficiently with End-to-End Feedback: The Value of (Identifiable) Networked Structure

by Ruihao Zhu

Submitted to the Department of Aeronautics and Astronautics on August 23, 2018,
in partial fulfillment of the requirements for the degree of
Master of Science

Abstract

In this thesis, we introduce efficient algorithms which achieve nearly optimal instance-dependent and worst case regrets for the problem of stochastic online shortest path routing with end-to-end feedback. The setting is a natural application of the combinatorial stochastic bandits problem, a special case of the linear stochastic bandits problem. We show how the difficulties posed by the large scale action set can be overcome by the networked structure of the action set. Our approach presents a novel connection between bandit learning and shortest path algorithms. Our main contribution is a series of adaptive exploration algorithms that achieves nearly optimal $O\left(\left(d^2\ln(T)+d^3\right)\Delta_{\max}/\Delta_{\min}^2\right)$ instance-dependent regret and $\widetilde{O}(d\sqrt{T})$ worst case regret at the same time. Driven by the carefully designed Top-Two Comparison (TTC) technique, the algorithms are efficiently implementable. We also conduct extensive numerical experiments to show that our proposed algorithms not only achieve superior regret performances, but also reduce the runtime drastically.

Thesis Supervisor: Eytan Modiano
Title: Professor, Department of Aeronautics and Astronautics

For my parents.

Acknowledgments

I have been fortunate enough to have Prof. Eytan Modiano as my advisor. He has been incredibly insightful, patient, and encouraging in guiding my research. To me, Prof. Modiano is much more than a research advisor. He helped me plan courses, prepare qualifying exams, improve writing skills, and others. Despite he is busy, he always finds time to meet and talk about our research project. I would like to thank Prof. Modiano for his time and effort.

I would like to thank the great fellow students in the Communications and Networking Research Group (CNRG) group: Qingkai Liang, Igor Kadota, Anurag Rai, Jianan Zhang, Thomas Stahlbuhk, Rajat Talak, Hyang-Won Lee, Abhishek Sinha, Bai Liu, Xinzhe Fy, and Vishrant Tripathi. It is the intriguing communication during the group meetings and individual discussions that help to shape this work. Rajat further deserves a special thank for sharing his desk with me.

During my first two years at MIT, I have met a lot of great people: Zhi Xu, Weike Sun, Weishun Zhong, Jun Yin, Alan Malek, Chelsea Qiu, Minghao Qiu, Chin-Chia Hsu... Thank you all so much for bringing me lots of fun!

Last but not least, my deepest thanks go to my beloved parents, Mr. Zhu Huayu and Ms. Zhu Yanting. They have always been supportive and encouraging. No words can express how grateful I am for having such great parents. Thank you so much for your unselfish love, support, and everything.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

We study the problem of shortest path routing over a network, where the link delays are not known in advance. When delays are known, it is possible to compute the shortest path in polynomial time via the celebrated Dijkstra's algorithm [21] or the Bellman-Ford algorithm [13]. However, link delays are often unknown, and evolve overtime according to some unknown stochastic process. Moreover, there are many real-world scenarios in which only the end-to-end delays are observable. For example, overlay network is an communication network architecture that integrates controllable overlay nodes into an uncontrollable underlay network of legacy devices. It is generally difficult to ensure individual link delay feedback when routing in an overlay network as the underlay nodes are not necessarily cooperative. Fig. 1-1 shows a very simple overlay network, where the only overlay nodes are the source node (node 1) and destination node (node 6); while the nodes within the dotted circle are underlay nodes. Here, the decision maker (DM) can choose to route the packets from one of the five paths available, namely $(1,2,3,6)$, $(1,2,5,6)$, $(1,2,3,5,6)$, $(1,2,4,5,6)$, and $(1,4,5,6)$. If it picks path $(1,2,5,6)$, it can only get the realized delay of the whole path $(1,4,5,6)$, but not any of the realized delays of link $(1,4)$, $(4,5)$, or $(5,6)$. These uncertainties and the network architectural constraints make the problem fall into the category of stochastic online shortest path routing with end-to-end feedback [35].

Stochastic online shortest path routing is one of the most fundamental real-time decision-making problems. In its canonical form, a DM is presented a network with d links, each

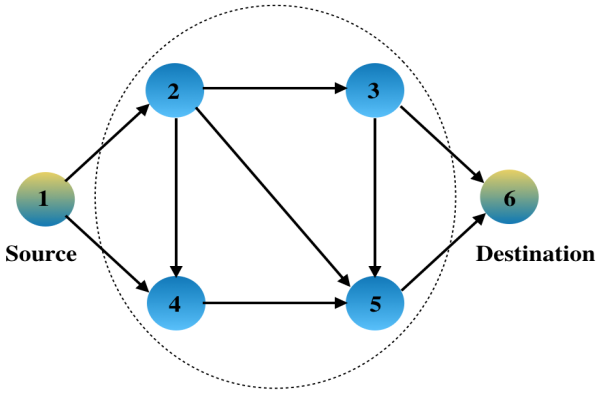


Figure 1-1: A toy example of an overlay network. Here, only the source and destination are overlay nodes. All other nodes belong to the underlay network.

link's delay is a random variable, following an unknown stochastic process with unknown fixed mean over T rounds. In each round, a packet arrives to the DM, and it chooses a path to route the packet from source to destination. The packet then incurs a delay, which is the sum of the delays realized on the associated links. Afterwards, the DM learns the end-to-end delay, *i.e.*, the realized delay of the path, but the individual link's delay remained concealed. This is often called the *bandit-feedback* setting [35, 25]. The DM's goal is to design a routing policy that minimizes the cumulative expected delay. When the DM has full knowledge of the delay distributions, it would always choose to route the packets through the path with shortest expected delay. With that in mind, a reasonable performance metric for evaluating the policy is the *expected regret*, defined to be the expected total delay of routing through the actual paths selected by the DM minus the expected total delay of routing through the path with shortest expected delay. In order to minimize the regret, the DM needs to learn the delay distributions on-the-fly. One viable approach to estimate the path delays is to inspect the end-to-end delays experienced by packets sent on different paths. This gives rise to an *exploration-exploitation dilemma*. On one hand, the DM is not able to estimate the delay of an under-explored path; while on the other, the DM wants to send the the packet via the estimated shortest path to greedily minimize the cumulative delay incurred by the packets.

The *Upper Confidence Bound* (UCB) algorithm, following the *Optimism-in-the-Face of Uncertainty* (OFU) principle, is one of the most prevalent strategies to deal with the

exploration-exploitation dilemma. In the ordinary stochastic MAB settings, the UCB algorithm proposes a very intuitive policy framework, that DM should select actions by maximizing over rewards estimated from previous data but only after biasing each estimate according to its uncertainty. Simply put, one should choose the action that maximizes the "mean plus confidence interval." Treating the inverse of delay as reward, a naive application of UCB algorithm to stochastic online shortest path routing can results in regret bounds and computation time that scale linearly with the number of paths. For small scale overlay networks, this achieves low regret efficiently. However, networks often have exponentially many paths, and direct implementation of the UCB algorithm is neither computationally efficient nor regret optimal. In the *combinatorial semi-bandits* setting, the realized delay of each individual link on the chosen path is revealed. The authors of [22] takes the advantage of the individual feedbacks, and propose a solution for the problem by computing the UCB of each link. The authors of [25, 35] further designs algorithms to match the regret lower bounds. Unfortunately, algorithms proposed for semi-bandit feedback setting cannot be extended to the bandit feedback setting as individual link feedback is not available.

To address the above mentioned shortcomings in efficiency and performance brought by the large size of action set and end-to-end feedback, existing works have tried to investigate the stochastic online shortest path routing problem with end-to-end feedback through the lens of linear stochastic bandits, see *e.g.*, [20, 1, 2]. There, tools from linear regression are employed to estimate the reward of each action, and the action with highest UCB is picked in each round. Although the regrets of the algorithms scale sub-linearly with the size of the action set, implementing the OFU algorithm requires the DM to solve a bilinear optimization problem that is polynomial time equivalent to the NP-hard negative definite linearly constrained quadratic programming problem [20]. This degrades the practicality of the algorithms significantly, especially when deployed in large-scale networks. Even worse, existing works in linear stochastic bandits literature examine the problem with a very general setup, *i.e.*, ignoring the network structure of the action set. Hence, only sub-optimal regret bounds are achieved.

As a matter of fact, finding efficient algorithms for linear stochastic bandits with optimal instance-dependent and worst case regrets remains as an open problem [14]. Although the

problem of stochastic online shortest path routing falls into the category of combinatorial stochastic bandits, a special case of linear stochastic bandits with action set constrained to be subset of $\{0,1\}^d$, the difficulty in designing efficient optimal algorithms remains the same. All of the above mentioned findings motivate us to exploit the networked structure of the action set to design efficient algorithms for the stochastic online shortest path problem with end-to-end feedback. Specifically, we aim at answering the following question:

Can we leverage the power of the networked structure to design efficient algorithms that achieve (nearly) optimal instance-dependent and worst case regret bounds simultaneously for stochastic online shortest path routing under bandit-feedback?

1.1 Related Works

Stochastic multi-armed bandits is a prevalent framework for sequential decision-making. Early work on stochastic MAB problems [31, 26, 23] tended to be more focused on asymptotic guarantees, whereas more recent work [10, 9] has been directed towards a non-asymptotic analysis in which regret can be bounded over a fixed time horizons T. Two of the best-known and well-studied techniques are known as the UCB algorithm that follows the OFU principle [10] and the explore then exploit algorithm [11, 34]. Recently, the Bayesian setting accompanied by the *Thompson Sampling* (TS) technique has also been thoroughly analyzed due to the ease of implementation and favorable empirical results [33].

To model inter-dependence relationships among different arms, models for stochastic linear bandits have also been studied. In stochastic linear bandits, each action can be described by a finite number of features, and the expected reward/loss function is linear in these features. The reward/loss function can thus be expressed as a vector in $\Re^d$, and the uncertainty arises from the noisy feedbacks of the observed rewards/losses. In [8, 17], the authors consider stochastic linear bandits with fixed finite action sets; while in [20, 1, 32], stochastic linear bandits with possibly infinite cardinality of actions has been studied. The authors of [2] unify these two lines of research, and have proposed the state-of-art algorithm for the problem. All these algorithms follow essentially the OFU principle. But the

OFU-inspired algorithms are impractical to run when the number of actions become large as they all require the solution of a NP-hard bilinear optimization problem. TS algorithms proposed in [33, 5, 3] are able to bypass the high computational complexities provided that the DM can efficiently sample from the posterior on the reward function. Unfortunately, achieving optimal regret bound via TS algorithms is possible only if the true prior over the reward/loss vector is known. To further capture the non-stochastic aspect of linear bandits, adversarial linear bandits in which the reward/loss vector can change over time arbitrarily (or even adversarially) have been studied [12, 4, 15]. Among them, [15] gave an efficient strategy with optimal regret when the action set is convex, and one can do efficient linear optimization on the action set. Since adversarial setting is not the main topic of this paper, interested readers can refer to [15] and the references therein.

A special case of linear bandits is combinatorial bandits where the action set is constrained to subset of $\{0,1\}^d$. In combinatorial stochastic bandits, it is often assumed that the reward/loss vector is observed at all the coordinates sampled by the action taken. This is the so-called semi-bandit feedback setting [6]. The authors of [22] initiated the study of combinatorial stochastic bandits under semi-bandit feedback and a network-structured action set; while [16] studied the general action set case. The authors of [25] further characterized tight upper and lower bounds for this problem. Assuming the noise is independent across different coordinates, the authors of [35] improved upon the results obtained in [25]. For the bandit feedback case, the authors of [28] gives algorithms that require brute-force search over the action space. For adversarial combinatorial bandits, the authors of [7] presented the efficient and optimal algorithm for the semi-bandit feedback case while the authors of [24] described an optimal algorithm for the bandit feedback case, but its computational complexity scales linearly with the number of actions.

Table 1.1 gives a brief summary of state-of-art results for linear bandits and combinatorial bandits.

	Problem	Instance-dependent regret	Worst case regret	Efficient
[2]	linear stochastic bandits	$O\left((\ln T + d\ln\ln T)^2 \ln T/\Delta_{\min}\right)$	$\widetilde{O}\left(d\sqrt{T}\right)$	$\times$
[5, 3]	linear stochastic bandits	N.A.	$\widetilde{O}\left(d^{3/2}\sqrt{T}\right)$	✓
[15]	adversarial linear bandits	N.A.	$\widetilde{O}\left(d\sqrt{T}\right)$	✓
[24]	adversarial combinatorial bandits	N.A.	$\widetilde{O}\left(l^{3/2}\sqrt{dT}\right)$	$\times$
[25]	combinatorial stochastic semi-bandits	$O\left(dl\ln T/\Delta_{\min}\right)$	$\widetilde{O}\left(\sqrt{dlT}\right)$	✓
[7]	adversarial combinatorial semi-bandits	N.A.	$\widetilde{O}\left(\sqrt{dT}\right)$	✓
Our work	combinatorial stochastic bandits	$O\left(\left(d^2\ln T + d^3\right)\Delta_{\max}/\Delta_{\min}^2\right)$	$\widetilde{O}\left(d\sqrt{T}+d^2\right)$	✓

Table 1.1: State-of-art algorithms for linear and combinatorial bandits. Combinatorial bandits is a special case of linear bandits with action set constrained to subset of $\{0,1\}^d$. In combinatorial bandits, l denotes the maximal ℓ_0 norm among all the actions.

1.2 Main Contributions and Outline of the Thesis

In this thesis, we give an affirmative answer to the above question. We start with algorithms for the stochastic online shortest path routing problem with identifiable network structure, and gradually remove the extra assumptions to arrive at the most general case. Specifically, our contributions can be summarized as follows:

- Assuming network identifiability, we first develop an efficient non-adaptive exploration algorithm with nearly optimal instance-dependent regret and sub-optimal worst case regret when the minimum gap [1] is known.

- The main contribution is a series of adaptive exploration algorithms with nearly optimal instance-dependent and worst case regrets without any knowledge of the minimum gap. Coupled with the novel Top-Two Comparison technique, the algorithms can be efficiently implemented.

- We show that our results can be extended to unidentifiable networks without degrading the regret performances.

- We conduct extensive numerical experiments to validate that our proposed algorithms

[1] The concepts of instance-dependent regret, worst case regret, and minimum gap will be defined in Section 2

not only achieve superior regret performances, but also reduce the runtime drastically.

The rest of the thesis is organized as follows.

- In Chapter 2, we introduces some background and describe the model of stochastic online shortest path routing.
- In Chapter 3, we present the main contributions of the thesis, including the intuitions, design details, theoretical analysis, discussions, and numerical results of the proposed algorithms.
- In Chapter 4, we conclude our thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2

Background

In this chapter, we lay out the foundation for our discussions.

2.1 Notations

Throughout the thesis, all the vectors are column vectors by default unless specified otherwise. We define $[n]$ to be the set $\{1,2,\ldots,n\}$ for any positive integer n. We use $\|\boldsymbol{x}\|_p$ to denote the ℓ_p norm of a vector $\boldsymbol{x} \in \Re^d$. To avoid clutter, we often omit the subscript when we refer to the ℓ_2 norm. For a positive definite matrix $A \in R^{d\times d}$, we use $\|\boldsymbol{x}\|_A$ to denote the matrix norm $\sqrt{\boldsymbol{x}^\top A\boldsymbol{x}}$ of a vector $\boldsymbol{x} \in \Re^d$. We also denote $x \wedge y$ as the minimum between $x, y \in \Re$. We describe the growth rate using the big-O notation $O(\cdot)$, and $\widetilde{O}(\cdot)$ if logarithmic factors are ignored.

2.2 Model

Given a directed acyclic network G, an online stochastic shortest path problem is defined by a d-dimensional unknown but fixed mean link delay vector $\boldsymbol{\mu} \in [0,\mu_{\max}]^d$, paths $\boldsymbol{a}_k = \left(a_{k,1},\ldots,a_{k,d}\right)^\top \in \mathscr{A} \subseteq \{0,1\}^d$ for $1 \le k \le K = |\mathscr{A}|$, and noise terms η_t for $1 \le t \le T$, where k is the index for paths and t is the index for rounds. Here, $\mathscr{A}$ is the set of all possible paths in G, and for a path $\boldsymbol{a}_k \in \mathscr{A}$, $a_{k,j} = 1$ if and only if it traverses link j. With some abuse of notation, we use k and $\boldsymbol{a}_k$ interchangeably to denote path $\boldsymbol{a}_k$, and we

refer $\mathscr{A}$ as both a set and a matrix. Routing a packet through path $\boldsymbol{a}_k$ in round t incurs the delay $L_{t,k} = \langle \boldsymbol{a}_k, \boldsymbol{\mu} \rangle + \eta_t$. Following the convention of existing bandits literature [2], we assume that η_t is conditionally R-sub-Gaussian, where $R \geq 0$ is a fixed and known constant. Formally, this means

$$\forall \alpha \in \Re \quad \mathbb{E}\left[\exp(\alpha \eta_t) | a_{I_1}, \ldots, a_{I_{t-1}}, \eta_1, \ldots, \eta_{t-1}\right] \leq \exp\left(\frac{\alpha^2 R^2}{2}\right)$$

and

$$\mathbb{E}\left[\eta_t | a_{I_1}, \ldots, a_{I_{t-1}}, \eta_1, \ldots, \eta_{t-1}\right] = 0.$$

In each round t, a DM follows a routing policy P to choose the path I_t to route the packet based on its past selections and previously observed feedback. Here, we consider *end-to-end* (bandit) feedback setting in which only the delay of the selected path is observable as a whole rather than the *individual* (semi-bandit) feedback in which the delays of all the traversed links are revealed. We measure the performance of P via expected regret against the optimal policy with full knowledge of $\boldsymbol{\mu}$

$$\mathbb{E}\left[\text{Regret}_T(P)\right] = \mathbb{E}\left[\sum_{t=1}^{T} L_{t,I_t} - \min_{k \in [K]} \sum_{t=1}^{T} L_{t,k}\right] = \sum_{t=1}^{T} \langle \boldsymbol{a}_{I_t}, \boldsymbol{\mu} \rangle - T \langle \boldsymbol{a}_*, \boldsymbol{\mu} \rangle,$$

where $\boldsymbol{a}_* = \arg\min_{\boldsymbol{a}_k \in \mathscr{A}} \langle \boldsymbol{a}_k, \boldsymbol{\mu} \rangle$ is the optimal path. In this paper, we require that $\boldsymbol{a}_*$ is unique[1]. For any path $\boldsymbol{a}_k \neq \boldsymbol{a}_*$, we define $\Delta_k = \langle \boldsymbol{a}_k - \boldsymbol{a}_*, \boldsymbol{\mu} \rangle$ as the difference of expected delay, *i.e.*, the gap, between $\boldsymbol{a}_k$ and $\boldsymbol{a}_*$. The maximum and minimum of Δ_k over all $k \in [K]$ with $\boldsymbol{a}_k \neq \boldsymbol{a}_*$ are denoted as $\Delta_{\max}$ and $\Delta_{\min}$, and are referred to as the maximum and minimum gap, respectively.Without loss of generality, we assume $\mu_{\max} = 1$[2] so that each path's expected delay is within $[0, d]$ and hence,

$$\Delta_{\max} \leq d. \tag{2.1}$$

As it is common in stochastic bandit learning settings [10, 2], we distinguish two different kinds of regret measures, namely the *instance-dependent* regret and the *worst case*

[1] We shall comment on this assumption in Section 3.4

[2] We shall relax this in the numerical experiments in Section 3.5.

regret

- **Instance-dependent regret:** A regret upper bound is called instance-dependent if it is comprised of quantities that only depend on T, d, Δ_k's, and absolute constants.
- **Worst case regret:** A regret upper bound is called worst case if it is comprised of quantities that only depend on to T, d, and absolute constants.

It is commonly known that when Δ_k's are allowed in the regret expressions, the regret can fall into the $\log T/\Delta_{\min}$ regime [10]. But depending on the choice of $\boldsymbol{\mu}$, $\Delta_{\min}$ can become extremely small for any given T, d, and R, and the instance-dependent regret guarantee becomes meaningless. We therefore have to turn to the worst-case regret bound. We note that, definitely, the regret is given by the minimum of the instance-dependent regret and worst case regret. Hence, it is desirable to obtain algorithms that have good instance-dependent and worst case regrets at the same time.

2.3 Design Challenges and Solution Strategies

Since the mean link delay vector $\boldsymbol{\mu}$ is unknown, and we only get to know the end-to-end delay of the chosen path in each round, the DM falls into the so called exploration-exploitation dilemma. On one hand, the DM needs to explore the network to acquire accurate estimate of the expected delay of each path; while on the other, it needs to exploit the path with least delay to ensure low regret. As our problem resembles the stochastic multi-armed bandits problem, there are at least two natural approaches to address it:

- **Optimism-in-the-Face-of-Uncertainty (OFU):** Following this principle, the DM balances exploration and exploitation by optimistically choosing the action with lowest confidence bound, *i.e.*, the empirical mean loss with the confidence interval subtracted. In [20, 2], this approach has been shown to work in the general linear stochastic bandits setting, yet as pointed out in Section 1, a direct adoption of the OFU principle to our problem cannot work. First, it fails to capture the underlying network structure, and brings a sub-optimal $O\left(\frac{\ln T}{\Delta_{\min}}(\ln T + d\ln\ln T)^2\right)$ instance-dependent and $O(d\log T\sqrt{T})$ worst case regret bounds [2]. Even worse, the practi-

cality of the algorithm is hindered by the high computational complexity in choosing the path to route. Indeed, it has been shown in [20] that the algorithm for path selection is polynomial time equivalent to a NP-hard negative definite linearly constrained quadratic programming.

- **Explore-then-Exploit:** Instead of doing exploration and exploitation simultaneously, the DM can collect data to construct accurate estimates for all actions' losses by first performing uniform exploration over all possible actions, and eliminates an action whenever it is confident that this action is sub-optimal. This procedure runs until there is only one action left. It has been shown in [11] that the adaptive exploration approach works well for the ordinary stochastic multi-armed bandits setting.

As it is unclear (at least to the authors) how to get the OFU approach to work efficiently in our setting, we adopt the explore-then-exploit approach here. An immediate difficulty in implementing this approach is that the DM cannot afford to uniformly explore exponentially many paths. It's thus of great importance to devise a way to efficiently collect data in the stochastic online shortest path routing setting.

2.4 Exploration Basis

In order to execute the uniform exploration efficiently, the DM relies on the basis of the network. Intuitively, a set $\mathscr{B} \subseteq \mathscr{A}$ is the basis of $\mathscr{A}$ if it "spans" the set $\mathscr{A}$, *i.e.*, each path in $\mathscr{A}$ can be expressed as a linear combination of the paths in $\mathscr{B}$. If the DM is able to accurately estimates the delays of the basis paths, it can also construct accurate delay estimators for all the paths in $\mathscr{A}$ thanks to the linearity of expectation. It is worth noting that the concept of exploration basis has been raised in adversarial linear bandits before [12], and we review it here as it is going to be useful for our problem.

2.4.1 Barycentric Spanners and Network Identifiability

Note that we have two requirements for $\mathscr{B}$, the first is that all the paths in $\mathscr{B}$ should exist in $\mathscr{A}$, *i.e.*, $\mathscr{B} \subseteq \mathscr{A}$; while the second is that the set $\mathscr{B}$ should span the original path set $\mathscr{A}$,

i.e., $\text{rank}(\mathscr{B}) = d$. To this end, we introduce the concept of *barycentric spanner* that has been invented in [12]:

Definition 1 (Barycentric spanner [12])**.** *Let $\mathscr{W}$ be a vector space over the real numbers, and $\mathscr{W}_0 \subseteq \mathscr{W}$ a subset whose linear span is a d-dimensional subspace of $\mathscr{W}$. A set $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d\} \subseteq \mathscr{W}_0$ is a barycentric spanner for $\mathscr{W}_0$ if every $\boldsymbol{x} \in \mathscr{W}_0$ may be expressed as a linear combination of elements of X using coefficients in $[-1,1]$. X is a C-approximate barycentric spanner if every $\boldsymbol{x} \in \mathscr{W}_0$ may be expressed as a linear combination of elements of X using coefficients in $[-C,C]$.*

The authors of [12] also presented a result regarding the existence and search of barycentric spanner.

Proposition 2 ([12])**.** *Suppose $\mathscr{W}_0 \subseteq \Re^d$ is a compact set not contained in any proper linear subspace. Given an oracle for optimizing linear functions over $\mathscr{W}_0$, for any $C > 1$ we may compute a C-approximate barycentric spanner for $\mathscr{W}_0$ in polynomial time, using $O\left(d^2 \log_C(d)\right)$ calls to the optimization oracle.*

The authors of [12] also present an algorithm for finding 2-approximate barycentric spanners. For completeness of presentation, we include this in Appendix B. The assumption stated in Proposition 2 that the set $\mathscr{W}_0$ is not contained in any proper subspace is closely related to network identifiability. Informally, we say that a network G with d links is *identifiable* if $\mathscr{A}$, its set of paths, spans the space $\Re^d$. In Theorem 3.1 of [29], the authors showed that it is in general impossible for G to be identifiable if all the paths in $\mathscr{A}$ originate from and end at the same pair of nodes, but Theorem 3.2 of [29] also states that it is possible for a subgraph of G to be identifiable. To accelerate our discussion, we call each of the links that is incident to either the source or the destination as an *external link*, and all other links the *internal links*. A network $G_0 \subseteq G$ with both the source and destination nodes as well as all the external links of G removed is called the *internal network*. In Fig. 1-1, links $(1,2),(1,4),(3,6)$, and $(5,6)$ are external links; while the rest are internal links. We can see that the internal network with node $2,3,4,5$ is identifiable as the paths $(2,3),(2,3,5),(2,5),(2,4,5)$, and $(4,5)$ span the space $\Re^5$. To this end, we temporarily make the following additional assumption (to be relaxed in Section 3.3)

Assumption 1. *The internal network of G is identifiable, and the expected delays of all the external links are known a priori. To avoid clutters, we further assume that the expected delays of the external links are deterministically 0.*

With some abuse of notations, d refers to the number of internal links whenever Assumption 1 is imposed. Given Proposition 2 and Assumption 1, the DM should be able implement Algorithm 5 in Appendix B to identify in polynomial time the 2-approximate barycentric spanners $\mathscr{B}$, *i.e.*, for any path $\boldsymbol{a} \in \mathscr{A}$, there exists some $\boldsymbol{\nu}_{\boldsymbol{a}} \in [-2,2]^d$, such that $\mathscr{B}\boldsymbol{\nu}_{\boldsymbol{a}} = \boldsymbol{a}$. By the definition of 2-approximate barycentric spanners, the maximal ℓ_2 norm of $\boldsymbol{\nu}_{\boldsymbol{a}}$ over all $\boldsymbol{a} \in \mathscr{A}$ is upper bounded by $2\sqrt{d}$, *i.e.*,

$$\max_{\boldsymbol{a}\in\mathscr{A}} \|\boldsymbol{\nu}_a\| \leq \sqrt{\sum_{i=1}^{d} 4} \leq 2\sqrt{d}. \tag{2.2}$$

Chapter 3

Main Results

In this chapter, we present the main contributions of the thesis.

3.1 Explore-then-Commit Algorithm: A Warm-Up

In this section, we develop the *Explore-then-Commit* (EC) algorithm based on non-adaptive exploration to solve the problem.

3.1.1 Design Intuitions

The design of the EC algorithm follows an intuitive rationale: if the DM is able to recover the expected delay of each path of the barycentric spanners accurately, it will also be able to accurately estimate the expected delay of each path as the delay of each path is the linear combination of the barycentric spanners. Once the DM believes that the optimal path have been detected with high probability, it could choose to commit to this path, and incurs low regret. To begin, we assume that the DM knows the minimum gap $\Delta_{\min}$. We will later relax this assumption to obtain practical algorithms.

3.1.2 Design Details

We denote the barycentric spanners of G as $\mathscr{B}$. Given a positive integer n ($\leq \lfloor T/d \rfloor$), we aim at getting a good estimate of $\boldsymbol{\mu}$ in the first $n \cdot d$ rounds, and then chooses the estimated

best path in each of the remaining $T - n \cdot d$ rounds. We thus call the first $n \cdot d$ rounds as the exploration stage, and the remaining $T - n \cdot d$ rounds as the committing stage. The EC algorithm divides the exploration stage into epochs of length d, and chooses each path in $\mathscr{B}$ once in every epoch until the end of the exploration stage. Afterwards, the EC algorithm makes use of the *Ordinary Least Square* (OLS) estimator to construct an estimate for $\boldsymbol{\mu}$. Specifically, the paths used in the first n epochs (or $n \cdot d$ rounds) form the design matrix

$$\mathscr{D}_n = \left(\boldsymbol{a}_{I_1}, \ldots, \boldsymbol{a}_{I_{nd}}\right)^\top$$

and the observed losses form the response vector

$$\boldsymbol{r}_n = \left(L_{1,I_1}, \ldots, L_{nd,I_{nd}}\right)^\top.$$

The OLS estimator then gives us

$$\hat{\boldsymbol{\mu}}_n = \left(D_n^\top D_n\right)^{-1} D_n^\top \boldsymbol{r}_n. \tag{3.1}$$

Thanks to the identifiability assumption, $D_n^\top D_n$ is full rank, and $\hat{\boldsymbol{\mu}}_n$ is well-defined. One can easily verify $\mathbb{E}[\hat{\boldsymbol{\mu}}_n] = \boldsymbol{\mu}$. Finally, the EC algorithm applies an arbitrary shortest path algorithm to compute the path with the lowest estimated delay, and commits to this path in the exploitation stage.

3.1.3 Regret Analysis

An essential tool in our analysis is a deviation inequality on the OLS estimator.

Lemma 3. *For a given positive integer m, the probability that the difference between $\hat{\boldsymbol{\mu}}_m$ and $\boldsymbol{\mu}$ under the $\mathscr{V}_m$ norm is not less than $R\sqrt{2d + 3\ln\delta^{-1}}$ is at most δ, after m epochs of explorations, i.e.,*

$$\Pr\left(\|\hat{\boldsymbol{\mu}}_m - \boldsymbol{\mu}\|_{\mathscr{V}_m} \geq R\sqrt{2\ln(2)d + 4\ln\delta^{-1}}\right) \leq \delta,$$

where $\mathscr{V}_m = \mathscr{D}_m^\top \mathscr{D}_m$.

Proof. The proof of Lemma 3 is mostly adopted from [27], and is deferred to Section A.1. □

With Lemma 3, we further develop the key deviation inequality that we will be working with throughout the rest of the paper

Lemma 4. *For a given positive integer m, the probability that there exists a path $\boldsymbol{a} \in \mathscr{A}$, such that the estimated mean delay of $\boldsymbol{a}$ deviates from its mean delay by at least $R\sqrt{8\ln(2)d^2 + 16d\ln\delta^{-1}/m}$ is at most δ, after m epochs of explorations, i.e.,*

$$\Pr\left(\exists \boldsymbol{a} \in \mathscr{A} : |\langle \boldsymbol{a}, \boldsymbol{\mu}\rangle - \langle \boldsymbol{a}, \hat{\boldsymbol{\mu}}_m\rangle| \geq R\sqrt{\frac{8\ln(2)d^2 + 16d\ln\delta^{-1}}{m}}\right) \leq \delta.$$

Proof. The proof of Lemma 4 makes use of Lemma 3 and the fact that the maximal ℓ_2 norm of $\boldsymbol{\nu}_{\boldsymbol{a}}$ over all $\boldsymbol{a} \in \mathscr{A}$ is upper bounded by $2\sqrt{d}$, *i.e.*, eq. (2.2). Due to limitation of space, we defer the full proof to Section A.2. □

We are now ready to present the regret bound of EC algorithm.

Theorem 5. *With the knowledge of* $\Delta_{\min}$, EC algorithm *can have the following regret bounds*

- *(Instance-dependent regret)*

$$O\left(\frac{\left(d^2\ln(dT) + d^3\right)\Delta_{\max}}{\Delta_{\min}^2}\right).$$

- *(Worst case regret)*

$$\widetilde{O}\left(d^{\frac{5}{4}}T^{\frac{2}{3}}\right).$$

Proof. See Section A.3. □

Remark 1. *Note that the instance-dependent regret bound obtained in Theorem 5 is a significant improvement compared to the direct application of OFU approach, and the worst case regret can be achieved without knowing* $\Delta_{\min}$. *Nevertheless, we should be aware*

that the choice of n for the instance-dependent regret bound relies on knowing $\Delta_{\min}$*, which is never the case in practice.*

Though being computationally efficient, the above remark indicates that the non-adaptive EC algorithm is not enough to achieve the optimal regret bounds.

3.2 Top-Two Comparison Algorithm: An Adaptive Exploration Approach

As we have seen from the previous discussions, the non-adaptive EC algorithm fail to make full use of the observed delays to explore adaptively, and the success of it relies almost solely on knowing $\Delta_{\min}$ ahead of time.

In this section, we study adaptive exploration algorithms that have been shown to achieve nearly optimal regret bounds in stochastic MAB [11, 34] to obtain nearly optimal instance-dependent and worst case regret bounds. Different from those in ordinary stochastic MAB settings, the algorithm builds on top of a novel *top two comparison* (TTC) method to allow efficient computation. We start by attaining a nearly optimal instance-dependent regret bound, and then show that how to attain a nearly optimal worst case regret bound simultaneously.

3.2.1 Design Intuitions

Adaptive exploration algorithms often serve as an alternative for UCB algorithms in stochastic multi-armed bandits [11, 34]. In [11, 34], the DM uniformly explores all remaining actions, and periodically executes an action elimination rule to ensure with high probability that:

- The optimal action remains with high probability;
- The sub-optimal actions can be removed effectively.

until only one action is left, and commits to that action in the rest of the rounds. The adaptive exploration algorithms achieve optimal $O(K\log T)$ instance-dependent and $O(\sqrt{KT\log T})$

worst case regret bounds for stochastic multi-armed bandits.

We start by demonstrating how an adaptive exploration algorithm can achieve the nearly optimal $O\left(\left(d\log T + d^2\right)\Delta_{\max}/\Delta_{\min}\right)$ instance-dependent regret bound. Similar to the EC algorithm, the adaptive exploration algorithm also splits the T rounds into an exploration stage and a committing stage: in each epoch $m = 1, 2, \ldots$ of the exploration stage, the DM selects every path in $\mathscr{B}$ once so that all of them have m samples. To ease our presentation, we denote the estimated shortest path after m epochs of uniform exploration as $\tilde{a}_m$, *i.e.*,

$$\tilde{\boldsymbol{a}}_m \leftarrow \arg\min_{\boldsymbol{a}\in\mathscr{A}} \langle \boldsymbol{a}, \hat{\boldsymbol{\mu}}_m \rangle,$$

and follow Lemma 4 to denote the $1-\delta$ confidence bound as $\tilde{\Delta}_m$, *i.e.*,

$$\tilde{\Delta}_m = R\sqrt{\frac{8\ln(2)d^2 + 16d\ln\delta^{-1}}{m}}. \tag{3.2}$$

We denote the total length of exploration stage by a random variable N. We then use a simple union bound to show the probability that there exists a path $\boldsymbol{a} \in \mathscr{A}$, such that the estimated mean delay of $\boldsymbol{a}$ deviates from its mean delay by at least $\tilde{\Delta}_m$ at the end of any epoch in the committing stage can be upper bounded as

$$\begin{aligned}
&\Pr\left(\exists m \in [N], \boldsymbol{a} \in \mathscr{A} : |\langle \boldsymbol{a}, \boldsymbol{\mu}\rangle - \langle \boldsymbol{a}, \hat{\boldsymbol{\mu}}_m\rangle| \geq \tilde{\Delta}_m\right) \\
\leq &\sum_{N=1}^{\lfloor T/d \rfloor} \sum_{m=1}^{N} \Pr\left(\exists \boldsymbol{a} \in \mathscr{A} : |\langle \boldsymbol{a}, \boldsymbol{\mu}\rangle - \langle \boldsymbol{a}, \hat{\boldsymbol{\mu}}_m\rangle| \geq \tilde{\Delta}_m\right) \\
\leq &\sum_{N=1}^{\lfloor T/d \rfloor} \sum_{m=1}^{T} \delta \\
\leq &\frac{T^2\delta}{d},
\end{aligned} \tag{3.3}$$

where we have used Lemma 4 and the fact that $N \leq T$ in inequality (3.3). In other words, if we denote the event E as following: any path $\boldsymbol{a}_k$'s estimated delay $\langle \boldsymbol{a}_k, \hat{\boldsymbol{\mu}}_m \rangle$ is within $\tilde{\Delta}_m$

distance from its true expected delay $\langle \boldsymbol{a}_k, \boldsymbol{\mu} \rangle$ for all $m \in [N]$, *i.e.*,

$$E = \{\forall m \in [N] \forall \boldsymbol{a} \in \mathscr{A} : |\langle \boldsymbol{a}, \boldsymbol{\mu} \rangle - \langle \boldsymbol{a}, \hat{\boldsymbol{\mu}}_m \rangle| \leq \tilde{\Delta}_m\} \tag{3.4}$$

then event E holds with probability at least $(1 - T^2\delta/d)$ in the adaptive exploration algorithm. From inequality (2.1), the worst possible total regret (*i.e.*, choosing the path with maximum gap in each round) an algorithm can incur is $T\Delta_{\max} \leq Td$, we can tune δ properly, *i.e.*, setting $\delta = T^{-3}$, so that the regret incurred by the algorithm in case E does not hold is at most 1. Therefore, we only need to focus the case when E holds.

Conditioned on E, we assert that the DM could detect if any of the remaining paths $\boldsymbol{a}_k$ is sub-optimal by checking whether

$$\langle \boldsymbol{a}_k, \hat{\boldsymbol{\mu}}_m \rangle - \langle \tilde{\boldsymbol{a}}_m, \hat{\boldsymbol{\mu}}_m \rangle > 2\tilde{\Delta}_m \tag{3.5}$$

holds at the end of each epoch m. Afterwards, the identified sub-optimal paths are eliminated. We use Figure 3-1 to illustrate the rationale behind this criterion. Note that in both Fig. 3-1(a) and 3-1(b), the horizontal right arrow is the positive number axis.

In Fig. 3-1(a), suppose $\langle \tilde{\boldsymbol{a}}_m, \hat{\boldsymbol{\mu}}_m \rangle$ and $\langle \boldsymbol{a}_k, \hat{\boldsymbol{\mu}}_m \rangle$ lie at B and F, respectively. Conditioned on event E, $\langle \tilde{\boldsymbol{a}}_m, \boldsymbol{\mu} \rangle$ should locate within the interval $[A, C]$ while $\langle \boldsymbol{a}_k, \boldsymbol{\mu} \rangle$ should locate within the interval $[D, H]$. Now if further B and F are more than $2\tilde{\Delta}_m$ away from each other, then

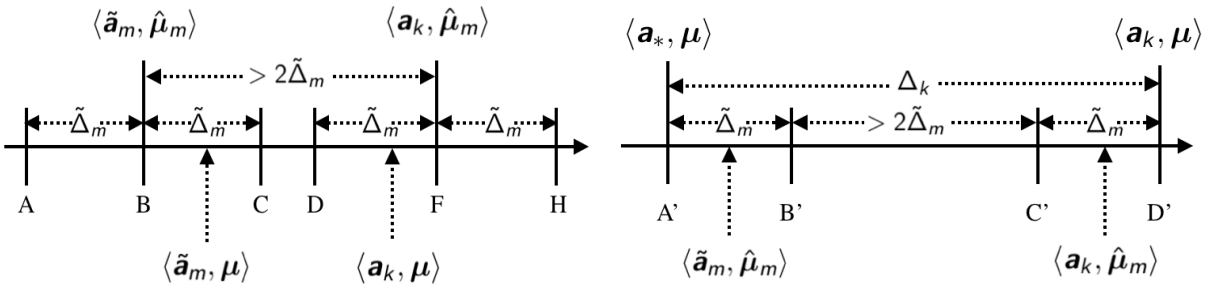
$$\langle \tilde{\boldsymbol{a}}_m, \boldsymbol{\mu} \rangle < \langle \boldsymbol{a}_k, \boldsymbol{\mu} \rangle. \tag{3.6}$$

In other words, path $\boldsymbol{a}_k$ is sub-optimal as its expected delay is at least longer than $\tilde{\boldsymbol{a}}_m$.

Similarly in Fig. 3-1(b), suppose $\langle \tilde{\boldsymbol{a}}_*, \boldsymbol{\mu} \rangle$ and $\langle \boldsymbol{a}_k, \boldsymbol{\mu} \rangle$ lie at A' and D', respectively. Conditioned on event E, $\langle \tilde{\boldsymbol{a}}_m, \hat{\boldsymbol{\mu}}_m \rangle$ ($\leq \langle \boldsymbol{a}_*, \hat{\boldsymbol{\mu}}_m \rangle$) should locate to the left of B' while $\langle \boldsymbol{a}_k, \hat{\boldsymbol{\mu}}_m \rangle$ should locate to the right of C'. Now if $\Delta_k > 4\tilde{\Delta}_m$, then

$$\langle \boldsymbol{a}_k, \hat{\boldsymbol{\mu}}_m \rangle - \langle \tilde{\boldsymbol{a}}_m, \hat{\boldsymbol{\mu}}_m \rangle > 2\tilde{\Delta}_m, \tag{3.7}$$

which means the sub-optimal path $\boldsymbol{a}_k$ is detected according to criterion (3.5).



(a) The removed arm is sub-optimal (b) All sub-optimal actions can be detected effectively

Figure 3-1: Intuitions underpinning criterion (3.5)

We formalize these observations in the following lemma.

Lemma 6. *Conditioned on event E, if criterion (3.5) holds, then*

1. *path $\boldsymbol{a}_k$ is sub-optimal;*

2. *any sub-optimal path $\boldsymbol{a}_k$ with $\Delta_k > 4\tilde{\Delta}_m$ is detected.*

Proof. See Section A.4. □

These two nice properties of criterion (3.5) jointly guarantees that the optimal path remains in $\mathscr{A}$, and any sub-optimal path $\boldsymbol{a}_k$ is removed once $\tilde{\Delta}_m$ shrinks down to below $\Delta_k/4$. Specifically, if m arrives to a value $\overline{m}$ that $\tilde{\Delta}_{\overline{m}} \leq \Delta_{\min}/4$ (or $\overline{m} = R^2(128\ln(2)d^2 + 256d\ln\delta^{-1})/\Delta_{\min}^2$), all sub-optimal paths should have been eliminated.

Roughly speaking, conditioned on E, the regret of the adaptive algorithm is

$$d\overline{m}\Delta_{\max} = O\left(\frac{\left(d^2\ln\delta^{-1} + d^3\right)\Delta_{\max}}{\Delta_{\min}^2}\right). \tag{3.8}$$

Recalling that the regret conditioned on $\neg E$ is at most Td, setting $\delta = T^{-3}$, the expected regret of this algorithm is upper bounded as $O\left(\left(d^2\ln T + d^3\right)\Delta_{\max}/\Delta_{\min}^2\right)$, and we shall formalize this analysis in Theorem 7. Surprisingly, adaptivity saves us from a lack of knowledge on the exact value of $\Delta_{\min}$.

3.2.2 Efficient Implementation

One may note that implementing the criterion (3.5) requires an enumeration over the set $\mathscr{A}$, which is typically exponential in size (in terms of d). In this subsection, we further propose an polynomial time implementation, namely the *Top Two Comparison* (TTC) algorithm, for our problem.

Different from the adaptive exploration algorithms proposed for stochastic multi-armed bandit problems [11, 34], which uniformly explores the set of remaining actions, our strategy decouples the exploration basis $\mathscr{B}$ from path elimination by making use of the barycentric spanners $\mathscr{B}$. In other words, the DM does not need to eliminate the sub-optimal paths one by one. It can instead remove all of them at the same time once the difference between the delay of the estimated shortest path and the delay of the estimated second shortest path is larger than $2\tilde{\Delta}_m$ for some epoch m.

To find the estimated second shortest path, we make the observation that the estimated second shortest path should traverse at least one link that is different than those in the estimated shortest path. The DM could start by iteratively setting the delay of links traversed by the shortest path to a large number, *i.e.*, $100d$, one at a time, while keeping the estimated delays of all other links intact, and find the delay of the shortest path with respect to the "perturbed" estimated delay vector. Finally, the minimum delay over these "perturbed" delays is the second shortest delay.

3.2.3 Design Details

We are now ready to formally present the TTC algorithm. Following the design guidelines presented in Sections 3.2.1 and 3.2.2, the TTC algorithm initializes the set of remaining path as $\mathscr{A}_1 = \mathscr{A}$, and divides the time horizon into epochs. In the m^{th} epoch, TTC algorithm distinguishes two cases:

1. If $\mathscr{A}_m$ contains only one path, TTC algorithm chooses this path, and sets $\mathscr{A}_{m+1} = \mathscr{A}_m$;

2. Otherwise, the TTC algorithm picks each path in $\mathscr{B}$ once so that every path in $\mathscr{B}$ has been selected m times. It then computes the OLS estimate $\hat{\boldsymbol{\mu}}_m$ for $\boldsymbol{\mu}$, and identifies the

path $\tilde{\boldsymbol{a}}_m$ with least estimated delay, *i.e.*, $\tilde{\boldsymbol{a}}_m = \arg\min_{\boldsymbol{a}\in\mathscr{A}_m}\langle \boldsymbol{a}, \hat{\boldsymbol{\mu}}_m\rangle$ and the path with estimated second shortest delay, *i.e.*, $\overline{\boldsymbol{a}}_m = \arg\min_{\boldsymbol{a}\in\mathscr{A}_m\setminus\{\tilde{a}_m\}}\langle \boldsymbol{a}, \hat{\boldsymbol{\mu}}_m\rangle$ via a second shortest path sub-routine. Afterwards, TTC algorithm checks the gap between $\tilde{\boldsymbol{a}}_m$ and $\overline{\boldsymbol{a}}_m$: If $\langle \overline{\boldsymbol{a}}_m, \hat{\boldsymbol{\mu}}_m\rangle - \langle \tilde{\boldsymbol{a}}, \hat{\boldsymbol{\mu}}_m\rangle \geq 2\tilde{\Delta}_m$. The set of remaining path for the $(m+1)^{\text{th}}$ epoch is denoted as $\mathscr{A}_{m+1} = \{\tilde{\boldsymbol{a}}_m\}$; otherwise, $\mathscr{A}_{m+1} = \mathscr{A}_m$.

The pseudo-code of TTC algorithm is shown in Algorithm 1 and the pseudo-code of the sub-routine for finding second shortest path is shown in Algorithm 2. Please note that the algorithms are run in epochs (indexed by m), and $\mathscr{A}$ can be represented by the incidence matrix of G.

Algorithm 1 Top-Two Comparison Algorithm

Input: A set of paths $\mathscr{A}$, barycentric spanners $\mathscr{B} \subseteq \mathscr{A}$, time horizon T.
Initialization: $\mathscr{A}_1 \leftarrow \mathscr{A}, \tilde{\Delta}_m \leftarrow R\sqrt{(8\ln(2)d^2 + 48d\ln T)/m}$ for $m = 1, 2, \ldots$.
for epoch $m = 1, 2, \ldots$ **do**
  if $|\mathscr{A}_m| = 1$, **then**
    Choose the path in $\mathscr{A}_m$.
    $\mathscr{A}_{m+1} \leftarrow \mathscr{A}_m$.
  else
    Choose each path in $\mathscr{B}$ once.
    $\hat{\boldsymbol{\mu}}_m \leftarrow \left(\mathscr{D}_m^\top \mathscr{D}_m\right)^{-1} \mathscr{D}_m^\top \boldsymbol{r}_m$.
    $\tilde{\boldsymbol{a}}_m \leftarrow \arg\min_{\boldsymbol{a}\in\mathscr{A}_m}\langle \boldsymbol{a}, \hat{\boldsymbol{\mu}}_m\rangle$.
    $\overline{\boldsymbol{a}}_m \leftarrow \mathrm{SSP}(\mathscr{A}, \hat{\boldsymbol{\mu}}_m, \tilde{\boldsymbol{a}}_m)$ (calls the second shortest path sub-routine).
    if $\langle \overline{\boldsymbol{a}}_i, \hat{\boldsymbol{\mu}}_m\rangle - \langle \tilde{\boldsymbol{a}}_i, \hat{\boldsymbol{\mu}}_m\rangle > 2\tilde{\Delta}_m$, **then**
      $\mathscr{A}_{m+1} \leftarrow \{\tilde{\boldsymbol{a}}_m\}$.
    else
      $\mathscr{A}_{m+1} \leftarrow \mathscr{A}_m$.
    end if
  end if
end for

3.2.4 Regret Analysis

The analysis essentially follows the intuition presented in Section 3.2.1, and the instance-dependent regret of the TTC algorithm is given by the following theorem.

Theorem 7. *For any $T \geq d\overline{m}$, the instance-dependent expected regret of* TTC algorithm *is*

Algorithm 2 Second Shortest Path (SSP) Sub-Routine

Input: A set of paths $\mathscr{A}$, a vector ψ of link delays, and the shortest path $\boldsymbol{a}$ with respect to $\mathscr{A}$ and ψ.
Output: The second shortest path with respect to $\mathscr{A}$ and ψ.
Initialization: $\boldsymbol{s} \leftarrow 0^d$.
for all $j \in [d]$ **do**
    if $a_j = 1$ **then**
        $\psi'_j \leftarrow \psi$.
        $\psi'_{j,j} \leftarrow 10d$.
        $\boldsymbol{c}_j \leftarrow \arg\min_{\boldsymbol{a}' \in \mathscr{A}} \langle \boldsymbol{a}', \psi'_j \rangle$.
        $s_j \leftarrow \langle \boldsymbol{c}_j, \boldsymbol{\nu} \rangle$.
    end if
end for
$j' = \arg\min_{j \in [d]: a_j = 1} s_j$.
return $\boldsymbol{c}_{j'}$.

bounded as

$$\mathbb{E}\left[\text{Regret}_T\left(\text{TTC algorithm}\right)\right] = O\left(\frac{\left(d^2 \ln T + d^3\right)\Delta_{\max}}{\Delta_{\min}^2}\right).$$

Proof. See Section A.5. □

We now pause for a while to comment on the bound provided in Theorem 7. In the worst case, *i.e.*, when $\Delta_{\max} = d$, if $\Delta_{\min} \leq d^{3/2}T^{-1/4}$, the RHS of Theorem 7 is of order $\widetilde{\Omega}\left(d\sqrt{T} + d^2\right)$. As the regret bound from adversarial linear bandits is of order $\widetilde{O}(d\sqrt{T})$, this indicates that the instance-dependent regret bound becomes vacuous once $\Delta_{\min}$ becomes smaller than $d^{3/2}T^{-1/4}$. Even though adaptive exploration saves us from not knowing $\Delta_{\min}$, it cannot achieve nearly optimal worst case regret bound automatically. This is because the TTC algorithm shares similar structure to EC algorithm, and as we have seen in Theorem 5 that tuning the parameter n to achieve sub-optimal $\widetilde{O}(d^{5/4}T^{2/3})$ worst case regret bound does not require any knowledge of $\Delta_{\min}$, neither. Some other techniques are needed if we want to get nearly optimal instance-dependent and worst case regrets at the same time.

3.2.5 Getting Nearly Optimal Worst Case Regret

It turns out that we can get nearly optimal instance-dependent and worst case regrets at the same time with just a bit more work. The key idea is to limit the length of the exploration stage so that once the smallest gap $\Delta_{\min}$ is believed to be smaller than $dT^{-1/4}$ with high probability, the DM switches to an efficient alternative algorithm for adversarial linear bandits to solve the problem. A candidate for the alternative algorithm can be found in [15]. Specifically, we set

$$\overline{n} = \sqrt{T}R^2(8\ln(2)d + 48\ln T)/d^2,$$

and modifies the TTC algorithm as following:

1. For each epoch $m \leq \overline{n}$, the DM runs the TTC algorithm;

2. If the set $\mathscr{A}_{\overline{n}+1}$ contains only one path, the DM selects this path in the rest of the rounds;

3. Else if the set $\mathscr{A}_{\overline{n}+1}$ contains more than one path, the DM finds that $\Delta_{\min} \leq 4\tilde{\Delta}_{\overline{n}} = \widetilde{O}\left(d^{3/2}T^{-1/4}\right)$ holds with probability at least $(1 - T^2\delta/d)$, and thus terminates the TTC algorithm, and runs the efficient algorithm for adversarial linear bandits in [15] over the network to solve the problem.

We name this as the *Modified Top Two Comparison* (MTTC) algorithm, and its pseudo-code is shown in Algorithm 3. We are now ready to state the regret bound of MTTC algorithm.

Theorem 8. *For any $T \geq d\overline{n}$, the expected regret of* MTTC algorithm *is bounded as*

- *(Instance-dependent regret)*

$$\mathbb{E}\left[\text{Regret}_T\left(\text{MTTC algorithm}\right)\right] = O\left(\frac{\left(d^2\ln T + d^3\right)\Delta_{\max}}{\Delta_{\min}^2}\right).$$

Algorithm 3 Modified Top-Two Comparison Algorithm

Input: A set of paths $\mathscr{A}$, barycentric spanners $\mathscr{B} \subseteq \mathscr{A}$, time horizon T.
Initialization: $\mathscr{A}_1 \leftarrow \mathscr{A}, \bar{n} \leftarrow \sqrt{T}R^2(8\ln(2)d + 48\ln T)/d^2, \tilde{\Delta}_m \leftarrow R\sqrt{(8\ln(2)d^2 + 48d\ln T/m}$ for $m = 1,2,\ldots$
for epoch $m = 1,2,\ldots,\bar{n}$ **do**
    Run TTC algorithm
end for
if $|\mathscr{A}_{\bar{n}+1}| = 1$ **then**
    Choose the path in $\mathscr{A}_{\bar{n}+1}$ for the rest of the rounds.
else
    Run the efficient algorithm for adversarial linear bandits in [15].
end if

- *(Worst case regret)*

$$\mathbb{E}\left[\text{Regret}_T\left(\text{MTTC algorithm}\right)\right] = \widetilde{O}\left(d\sqrt{T}\right).$$

Proof. See Section A.6. □

3.3 Algorithm for General Networks

The success of TTC algorithm and MTTC algorithm in achieving nearly optimal regrets rely on being able to identify the barycentric spanners, and make efficient exploration over them. In finding barycentric spanners for the network G, it is required that Assumption 1 or the following two conditions hold:

1. The internal network is identifiable.

2. The expected delays of all the external links are known a priori.

In practice, this is not always the case. For example, if the network scale grows large, it is very likely that even the internal network of G is not fully identifiable. Also, if the external links are shared among many entities, it is hard to obtain the expected delays of all the external links. Without Assumption 1, the network becomes unidentifiable, and one possible way to overcome this shortcoming is to project $\mathscr{A}$ into some sub-space so that it is still full rank in that sub-space. But it is unclear how to implement the projection without

enumerating all the paths in $\mathscr{A}$, which is computationally inefficient. Therefore, we are in need of a new technique for our problem. Fortunately, the explore-then-commit nature of the algorithms proposed allows us to replace the barycentric spanners with an arbitrary basis set of $\mathscr{A}$, and use them as a template to achieve low regret. Throughout this section, we shall assume that the rank of $\mathscr{A}$ is $d_0 < d$.

3.3.1 Additional Notations

In this section, we will make use of matrix notations heavily. For any matrix $M \in \Re^{d_1 \times d_2}$, we use $M(i,j)$ to denote its element at the i^{th} row and j^{th} column, $M(i,:)$, and $M(:,j)$ to denote its i^{th} row and j^{th} column vectors, respectively, and $M([i_1,i_2],:)$, and $M(:,[j_1,j_2])$ to denote the matrices obtained by keeping only the i_1^{th} to i_2^{th} rows and j_1^{th} to j_2^{th} columns, respectively Moreover, $M(-i,:)$ and $M(:,-j)$ are the matrices obtained by removing the i^{th} row and j^{th} column of M, respectively. $M(-i,-j)$ is the (d_1-1)-by-(d_2-1) matrix obtained by removing the i^{th} row and j^{th} column of M simultaneously.

3.3.2 Efficient Algorithm for Identifying the Basis

As a first step, we present an algorithm that finds a basis of $\mathscr{A}$ even when the network G is unidentifiable. Inspired by the algorithm for finding barycentric spanners for identifiable networks, *i.e.*, Algorithm 5 in Appendix B, the high-level idea can be described as following:

1. Initiate a matrix $\mathscr{C}$ to the d-dimensional identity matrix;
2. Greedily replace as many columns of $\mathscr{C}$ as possible by paths in $\mathscr{A}$ while keeping $\mathscr{C}$ full rank.
3. All the columns in $\mathscr{C}$ that are obtained from $\mathscr{A}$ constitute $\mathscr{B}$.

Since steps (1) and (3) can be easily implemented, we further elaborate on an iterative algorithm for step (2). For ease of presentation, we use $\mathscr{C}_u$ to denote the resulted matrix after the u^{th} iteration with $\mathscr{C}_0 = \mathscr{C}$. At the beginning of the $(u+1)^{\text{th}}$ iteration, suppose $\mathscr{C}_u$

can be written as

$$\mathscr{C}_u = \left(\mathscr{C}'_u, \mathscr{C}''_u\right), \tag{3.9}$$

where $\mathscr{C}'_u$ are the columns obtained from $\mathscr{A}$; while $\mathscr{C}''_u$ are the columns inherited from $\mathscr{C}_0$, the algorithm then finds a column $\boldsymbol{c} \in \mathscr{C}''_u$ such that replacing $\boldsymbol{c}$ with an element in $\boldsymbol{a} \in \mathscr{A}$ can result in a full rank matrix, and sets

$$\mathscr{C}_{u+1} = \left(\boldsymbol{a}, \mathscr{C}_u(:, -j)\right), \tag{3.10}$$

where j is the column index of $\boldsymbol{c}$. This algorithm terminates once such $\boldsymbol{c}$ cannot be found in $\mathscr{C}_u$ after some iterations u.

To efficiently implement the above iterative algorithm, *i.e.*, to find such $\boldsymbol{a}$ in each iteration if it exists, we note that the matrix $\mathscr{C}_{u+1}$ is full rank if and only if the determinant of $\mathscr{C}_{u+1}$ is nonzero, *i.e.*,

$$\operatorname{rank}(\mathscr{C}_{u+1}) = d \quad \Leftrightarrow \quad \det \mathscr{C}_{u+1} \neq 0. \tag{3.11}$$

For now, suppose we are given a full rank matrix $\mathscr{C}_u$, if the j^{th} column of $\mathscr{C}_u$ is replaced by an $\boldsymbol{a} \in \mathscr{A}$ to form

$$\mathscr{C}_u^j = \left(\mathscr{C}_u(1, j-1), \boldsymbol{a}, \mathscr{C}_u(j+1, d)\right),$$

the determinant of $\mathscr{C}_u^j$ can be written as a linear function of $\boldsymbol{a}$, *i.e.*,

$$\det \mathscr{C}_u^j = \sum_{i=1}^{d} \left[(-1)^{i+j} \det\left(\mathscr{C}_u(-i, -j)\right)\right] a_i \tag{3.12}$$

by the Laplace expansion, and the value of $\det\left(\mathscr{C}_u(-i, -j)\right)$ can be computed efficiently using the LU decomposition. Now to find an index $j(> u)$ and $\boldsymbol{a}$ that satisfies $\det \mathscr{C}_u^j \neq 0$, we can equivalently solve the following optimization problem

$$\max_{\boldsymbol{a} \in \mathscr{A}} \left|\det \mathscr{C}_u^j\right|, \tag{3.13}$$

for all $j > u$. If there exists some $j > u$ such that the solution $\boldsymbol{a}$ satisfies $\left|\det \mathscr{C}_u^j\right| > 0$, we can then replace the j^{th} column of $\mathscr{C}_u$ by $\boldsymbol{a}$ to form $\mathscr{C}_{u+1}$ according to eq. (3.10).

For a given j, defining a vector $\boldsymbol{c}_j \in \Re^d$ with each entry defined by eq. (3.12), *i.e.*,

$$\forall i \in [d] \quad c_{j,i} = \left[(-1)^{i+j} \det\left(\mathscr{C}_u(-i,-j)\right)\right], \tag{3.14}$$

the optimal solution of (3.13) can be obtained by first solving the following two sub-problems

$$\max_{\boldsymbol{a}\in\mathscr{A}} \langle \boldsymbol{c}_j, \boldsymbol{a}\rangle, \quad \min_{\boldsymbol{a}\in\mathscr{A}} \langle -\boldsymbol{c}_j, \boldsymbol{a}\rangle, \tag{3.15}$$

and then picking the solution with larger absolute value. To solve the first sub-problem, we can use the following steps:

1. Assign delay $c_{j,i}$ to link i of G for all $i \in [d]$;

2. Compute the longest path. This requires a call to an appropriate efficient longest path algorithm for directed acyclic graphs, *e.g.*, topological sorting [19].

The formal description of this algorithm for basis identification is shown in Algorithm 4.

We are now ready to prove the correctness of the algorithm, *i.e.*, if the rank of $\mathscr{A}$ is $d_0 < d$, then Algorithm 4 returns a basis $\mathscr{B} \subseteq \mathscr{A}$, such that the rank of $\mathscr{B}$ is d_0.

Lemma 9. *Algorithm 4 terminates in polynomial time. Upon termination, the matrix $\mathscr{B}$ returned by Algorithm 4 is a basis of $\mathscr{A}$, i.e., $\mathscr{B}$ has linearly independent columns and for every $\boldsymbol{a} \in \mathscr{A}$, there exists a vector $\boldsymbol{\nu}_a$, such that $\mathscr{B}\boldsymbol{\nu}_a = \boldsymbol{a}$.*

Proof. See Section A.7 □

3.3.3 Comparing to the Identifiable Network Setting

Again, our hope is to modify MTTC algorithm so that it can achieve low regrets for general networks. With the new basis $\mathscr{B}$ at hand, we can almost follow what we have developed in Section 3.2. But a more careful inspection suggests the following differences between

Algorithm 4 Basis Identification for General Networks

Input: A set of paths $\mathscr{A}$.
Initialization: $\mathscr{C}_0 \leftarrow I, u \leftarrow 0$,Flag$\leftarrow$True.
Output: $\mathscr{B}$, the basis $\mathscr{A}$.
while $u \leq d-1$ **and** Flag==True **do**
  for $j = u+1, \ldots, d$ **do**
    $\forall i \in [d]\ c_{j,i} \leftarrow (-1)^{i+j} \det(\mathscr{C}_u(-i,-j))$.
    $\boldsymbol{a}'_1 \leftarrow \arg\max_{\boldsymbol{a}\in\mathscr{A}} \langle \boldsymbol{c}_j, \boldsymbol{a}\rangle$.
    $\boldsymbol{a}'_2 \leftarrow \arg\min_{\boldsymbol{a}\in\mathscr{A}} \langle -\boldsymbol{c}_j, \boldsymbol{a}\rangle$.
    $\boldsymbol{a} \leftarrow \arg\max_{\boldsymbol{a}'_1,\boldsymbol{a}'_2} \left\{ |\langle \boldsymbol{c}_j, \boldsymbol{a}'_1\rangle|, |\langle \boldsymbol{c}_j, \boldsymbol{a}'_2\rangle| \right\}$
    if $|\langle \boldsymbol{c}_j, \boldsymbol{a}\rangle| > 0$ **then**
      $\mathscr{C}_{u+1} \leftarrow (\boldsymbol{a}, \mathscr{C}_u(:,-j))$.
      $u \leftarrow u+1$.
      break
    else if $j == d$
      Flag$\leftarrow$False.
    end if
  end for
end while
$\mathscr{B} \leftarrow \mathscr{C}_u(:,[1:u])$
return $\mathscr{B}$.

identifiable network setting and the general network setting. First, the number of columns of the basis $\mathscr{B}$ is d_0 instead of d. This can be easily resolved by setting the length of each epoch to d_0. However, we note that this simple change is not enough: the d-dimensional matrix $\mathscr{V}_m = \left(\mathscr{D}_m^\top \mathscr{D}_m\right) = m\mathscr{B}\mathscr{B}^\top$ is not full rank, *i.e.*, $\text{rank}(\mathscr{V}_m) = d_0$ for all $m \geq 1$, which means we cannot compute the OLS estimate of $\boldsymbol{\mu}$ the same as eq. (3.1).

3.3.4 Obtaining Low Regret for General Networks

To allow the DM to implement the MTTC algorithm for general networks, we need to resolve the issues raised by the singularity of $\mathscr{V}_m$. To this end, we use a slightly different version of OLS estimator [30], *i.e.*, the OLS estimator of $\boldsymbol{\mu}$ after m epochs of explorations is

$$\hat{\boldsymbol{\mu}}_m = \left(\mathscr{D}_m^\top \mathscr{D}_m\right)^\dagger \mathscr{D}_m \boldsymbol{r}_m, \tag{3.16}$$

where $\left(\mathscr{D}_m^\top \mathscr{D}_m\right)^\dagger$ denotes the Moore-Penrose pseudo-inverse of $\mathscr{V}_m = \left(\mathscr{D}_m^\top \mathscr{D}_m\right)$ as $\mathscr{V}_m$ is not invertible. Accompanying this new estimator is a new deviation inequality.

Lemma 10. *For a given positive integer m, the probability that there exists a path $\boldsymbol{a} \in \mathscr{A}$, such that the estimated mean delay of $\boldsymbol{a}$ deviates from its mean delay by at least $R\sqrt{32\ln(6)d_0^2 S + 32 d_0 S \ln \delta^{-1}/m}$ is at most δ, after m epochs of explorations, i.e.,*

$$\Pr\left(|\langle \boldsymbol{a}, \boldsymbol{\mu}\rangle - \langle \boldsymbol{a}, \hat{\boldsymbol{\mu}}\rangle| \geq R\sqrt{\frac{32\ln(6)d_0^2 S + 32 d_0 S \ln \delta^{-1}}{m}} \right) \leq \delta,$$

where S is the smallest positive number such that every coordinate of $\boldsymbol{\nu}_a$ lies within $[-\sqrt{S}, \sqrt{S}]$, i.e., $\boldsymbol{\nu}_{\boldsymbol{a}} \in \left[-\sqrt{S}, \sqrt{S}\right]^{d_0}$ for all $\boldsymbol{a} \in \mathscr{A}$.

Proof. See Section A.8. □

We note that this deviation inequality is in general different (and often worse) than that of Lemma 4 as $\mathscr{V}_m$ is not invertible and S is not equal to 2 since the columns of $\mathscr{B}$ are not necessarily 2-approximate barycentric spanners.

To work with this deviation inequality, we need to have at least an upper bound of S. Consider the matrix $\mathscr{C}_{d_0} = (\boldsymbol{f}_1, \ldots, \boldsymbol{f}_d)$, *i.e.*, the matrix right after the termination of the while-loop in Algorithm 4. By design of Algorithm 4, we know that $\mathscr{B}$ is the first d_0 columns of $\mathscr{C}_{d_0}$, *i.e.*, $\mathscr{B} = \mathscr{C}_{d_0}(:, [1 : d_0])$. By definition of $\boldsymbol{\nu}_a$, we can write

$$\boldsymbol{a} = \mathscr{B}\boldsymbol{\nu}_a = \sum_{i=1}^{d_0} \nu_{\boldsymbol{a},i} \mathscr{B}(:,i) = \sum_{i=1}^{d_0} \nu_{\boldsymbol{a},i} \mathscr{C}_{d_0}(:,i). \tag{3.17}$$

We then make the following observation: for every $j \leq d_0$, if we replace $\mathscr{C}_{d_0}(:,j)$ by $\boldsymbol{a}$, we have

$$\begin{aligned}
\left|\det\left(\boldsymbol{a}, \mathscr{C}_{d_0}(:,-j)\right)\right| &= \left|\det\left(\mathscr{C}_{d_0}(:,-j), \sum_{i=1}^{d_0} \nu_{\boldsymbol{a},i}\mathscr{C}_{d_0}(:,i)\right)\right| \\
&= \left|\sum_{i=1}^{d_0} \nu_{\boldsymbol{a},i} \det\left(\mathscr{C}_{d_0}(:,-j), \mathscr{C}_{d_0}(:,i)\right)\right| \qquad (3.18) \\
&= \left|\nu_{\boldsymbol{a},j} \det\left(\mathscr{C}_{d_0}(:,-j), \mathscr{C}_{d_0}(:,j)\right)\right| \qquad (3.19)
\end{aligned}$$

$$= \left|v_{\boldsymbol{a},j}\right| \left|\det \mathscr{C}_{d_0}\right|, \tag{3.20}$$

where eq. (3.18) follows from the linearity of det operator and eq. (3.19) follows from the fact that the determinant of a matrix is 0 if a matrix has two identical columns. Re-arranging the terms in eq. (3.20), we have that

$$S = \max_{j\in[d_0],\boldsymbol{a}\in\mathscr{A}} \left|v_{\boldsymbol{a},j}\right| = \max_{j\in[d_0],\boldsymbol{a}\in\mathscr{A}} \frac{\left|\det\left(\boldsymbol{a}, \mathscr{C}_{d_0}(:,-j)\right)\right|}{\left|\det \mathscr{C}_{d_0}\right|}. \tag{3.21}$$

As demonstrated in the Section 3.3.2, the optimization problem at the RHS of (3.21) can be computed efficiently by first computing $\max_{\boldsymbol{a}\in\mathscr{A}} \left|\det\left(\boldsymbol{a}, \mathscr{C}_{d_0}(:,-j)\right)\right|$ individually for every $j \in [d_0]$, and then taking maximum over $j \in [d_0]$.

Now, we are ready to apply MTTC algorithm to general networks. Inspired by the identifiable networks setting, we only need to change the initialization parameters in MTTC algorithm accordingly:

$$\tilde{\Delta}_m = R\sqrt{\frac{32\ln(6)d_0^2 S + 96 d_0 S \ln T}{m}}, \forall m = 1,2,\ldots, \tag{3.22}$$

$$\overline{n} = \sqrt{T}R^2(32\ln(2)d_0 + 96\ln T)/d_0^2 S. \tag{3.23}$$

The regret bound of MTTC algorithm for unidentifiable networks also follow immediately from that of Theorem 8.

Theorem 11. *The expected regret of* MTTC algorithm *is bounded as*

- *(Instance-dependent regret)*

$$\mathbb{E}\left[\text{Regret}_T\left(\text{MTTC algorithm}\right)\right] = O\left(\frac{\left(d^2\ln T + d^3\right) S\Delta_{\max}}{\Delta_{\min}^2}\right).$$

- *(Worst case regret)*

$$\mathbb{E}\left[\text{Regret}_T\left(\text{MTTC algorithm}\right)\right] = \widetilde{O}\left(d\sqrt{T}\right).$$

The proof of Theorem 11 is omitted as it is very similar to that of Theorem 8.

3.4 Discussions and Implications

In this section, we comment on the model assumptions and results.

Remark 2. *In Chapter 2, we have assumed that there exists an unique optimal path. Without this it, the instance-dependent regret bounds for the* TTC algorithm *and the* MTTC algorithm *break.*

Remark 3. *Other algorithms, such as that of [18] for combinatorial bandit optimization or [5] for linear contextual bandit can also serve as the alternative in the* MTTC algorithm *by setting the $\overline{n}$ accordingly.*

Remark 4. *With access to a linear optimization oracle, our proposed algorithms can be transplanted to any combinatorial stochastic bandits problem to attain nearly optimal instance-dependent and worst case regrets with polynomial computation time. Nevertheless, a simple adaptation to linear stochastic bandits is not guarantee to work as the action set is an arbitrary subset of $\Re^d$, and the estimated shortest and second shortest path may use exactly the same links. Therefore, the second shortest path sub-routine cannot work correctly.*

3.5 Numerical Experiments

In this section, we conduct extensive numerical experiments on synthetic data to validate the performances of TTC algorithm and MTTC algorithm in terms of time average regret, *i.e.*, regret/number of rounds, and computational efficiency.

3.5.1 Setups

We first present the setup of our numerical experiments. We vary T from 5000 to 25000 with a step size of 5000. We set $\mu_{\max} = 1000$ to allow enough heterogeneity in each link's delay distribution, and we use normal distribution with $R = 0.1$ and 1 for the noise terms. To demonstrate that our algorithms work well for networks with complicated topology and various scales, we use the *grid network* structure. In a grid network, the underlay network

is a $p \times p$ grid, and each node is able to route a packet to both the node on the right and at beneath (if exists). Fig. 3-2 shows the network with a 4-by-4 underlay grid. In our experiment, we consider grid networks with $p = 2, 4, 6$, and 8. For ease of implementation,

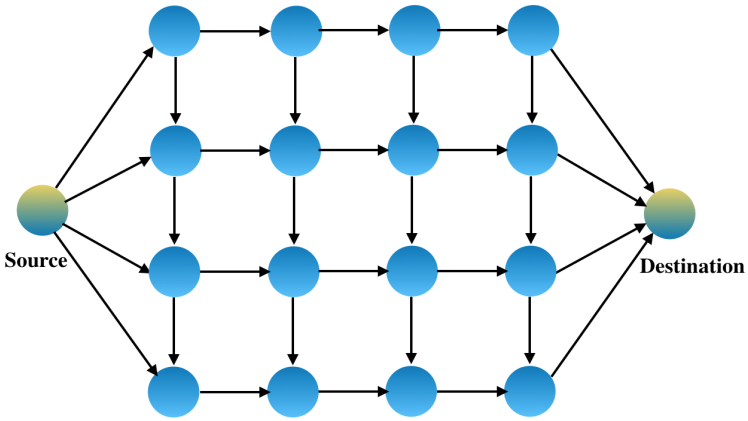


Figure 3-2: Examples of 4-by4 grid network

we use the Thompson sampling algorithm proposed in [5] as the alternative in MTTC algorithm, and compare the performance of our algorithms with the (inefficient) OFU algorithm proposed in [2] and the Thompson sampling algorithm [5]. Also for fair comparisons, we set a hard computation budget of 100 seconds for each instance, and call a runtime error once the runtime of an algorithm exceed this limit. We note that the parameters here are chosen randomly, and the results should be similar with another set of parameters. Finally, all the results presented here are averaged over 200 iterations.

3.5.2 Results

We first describe some basic statistics as well as the runtime of different algorithms in Table 3.1 to visualize the scales and complexities of the networks for different values of p. As we can see, the OFU and TS algorithms consume tens to hundreds of times more runtime than the TTC algorithm for all the cases. When $k \geq 6$, the runtime of OFU algorithm exceeds the computation budget.

The results of time average regret are shown in Fig. 3-4 and 3-3. From the plots, we can read that the time average regrets of TTC algorithm and MTTC algorithm are significantly

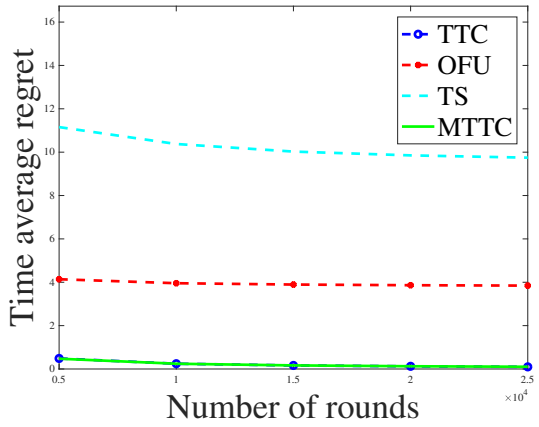
$p =$	2	4	6	8
d : #links	8	32	72	128
d_0 : size of basis	4	16	36	64
$\lvert\mathscr{A}\rvert$: #paths	4	56	792	11440
minimum #hop	3	5	7	9
maximum #hop	4	8	12	16
runtime of TTC algorithm (s)	0.01	0.16	1.00	1.14
runtime of MTTC algorithm (s)	0.10	5.78	18.65	35.08
runtime of OFU algorithm (s)	1.55	9.43	>90	>90
runtime of TS algorithm (s)	29.60	35.14	39.81	51.96

Table 3.1: Basic statistics and average runtime of different algorithms for grid networks when $R = 1$

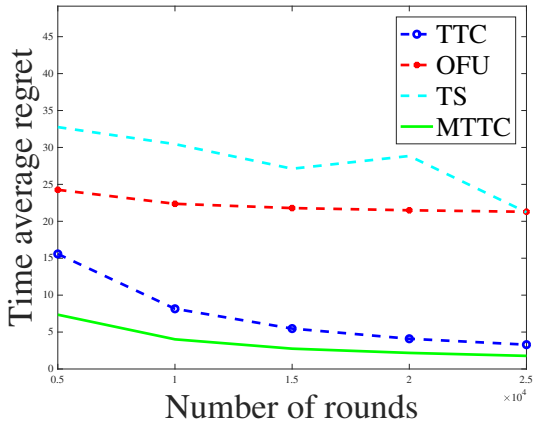
lower than those of the OFU and the TS algorithms' in the low noise (or $R = 0.1$) case, especially when the number of rounds is large. The only exception is when $p = 8$ and $T \leq 15000$, the time average regret of TTC algorithm is larger than that of the TS algorithm. This is because as p increases, the value of S also increases, and it thus takes longer time for TTC algorithm to identify the optimal path. For the high noise case, we can see that the performances of MTTC algorithm and TS algorithm are close. Although the performances of TTC algorithm is worse than the TS algorithm, we believe TTC algorithm and MTTC algorithm can outperform TS algorithm as the T increases.

3.5.3 Additional Results

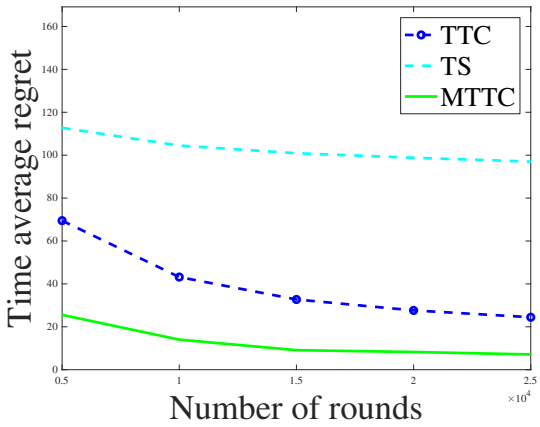
To verify our conjecture, we conduct additional experiment for the $R = 1$ case with larger T with $p = 2$ and $p = 4$. When $p = 2$, we vary T from 5×10^4 to 10^5 with a step of 10^4; when $p = 4$, we vary T from 10^8 to 5×10^8 with a step of 10^8. The plots in Fig. 3-5 clearly show that when T becomes large enough, TTC algorithm and MTTC algorithm finish with lower regrets than TS algorithm.



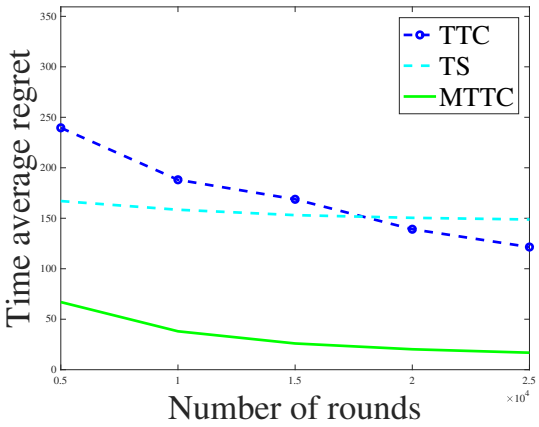
(a) $p = 2, R = 0.1$



(b) $p = 4, R = 0.1$

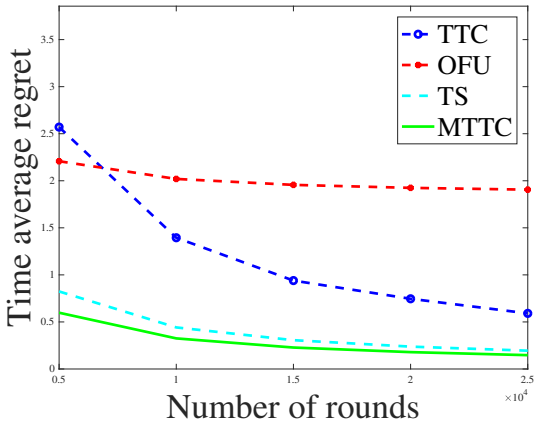


(c) $p = 6, R = 0.1$

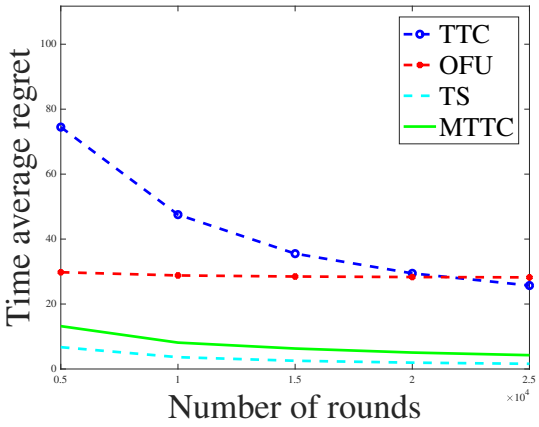


(d) $p = 8, R = 0.1$

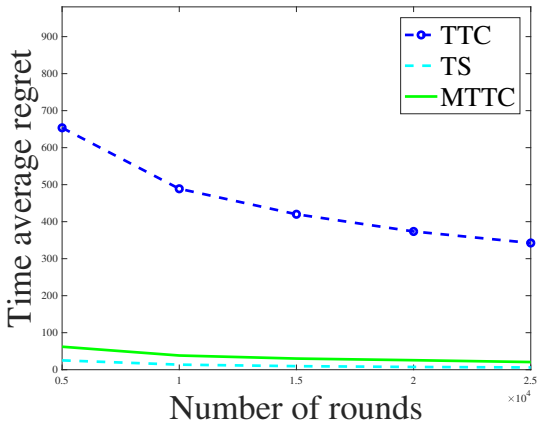
Figure 3-3: Plots of results when $R = 0.1$



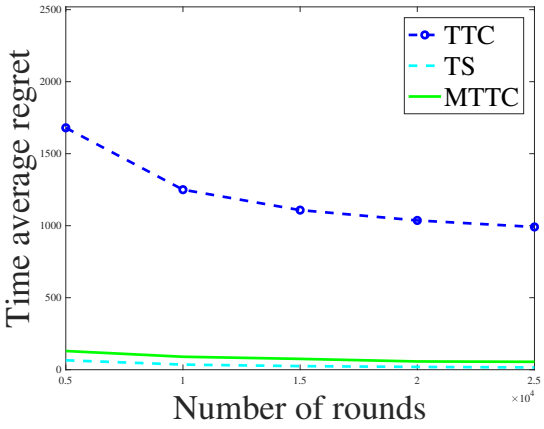
(a) $p = 2, R = 1$



(b) $p = 4, R = 1$

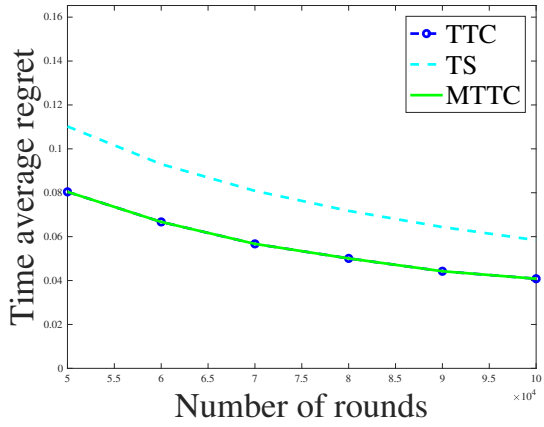


(c) $p = 6, R = 1$

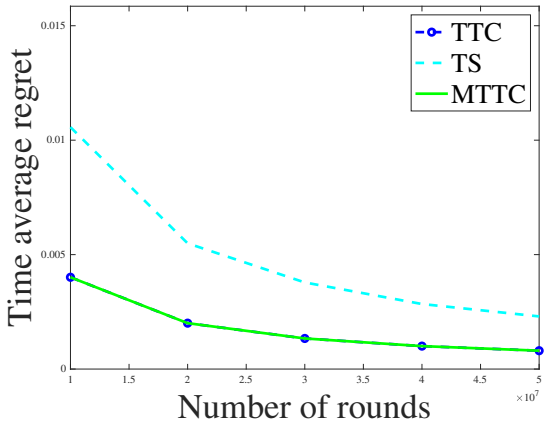


(d) $p = 8, R = 1$

Figure 3-4: Plots of results when $R = 1$



(a) $p = 2, R = 1$



(b) $p = 4, R = 1$

Figure 3-5: Additional results for grid networks

Chapter 4

Conclusion

In this thesis, we developed efficient algorithms with nearly optimal regrets for the problem of stochastic online shortest path routing with end-to-end feedback. Starting from identifiable networks, we have introduced EC algorithm to obtain nearly optimal instance-dependent and sub-optimal worst case regrets efficiently. We have then presented the TTC algorithm and the MTTC algorithm, a series of efficient algorithms that achieve nearly optimal instance-dependent and worst case regrets. Afterwards, we extended our results to general networks. Finally, we conducted extensive numerical experiments to demonstrate the superior regret performances and computation efficiency of our proposed algorithms.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix A

Proofs

A.1 Proof of Lemma 3

Following the proof in [27], denote $\boldsymbol{w} = \boldsymbol{r}_m - \mathscr{D}\boldsymbol{\mu}$, as the vector of noise, and $\boldsymbol{z} = \mathscr{D}_m^\top \boldsymbol{w}$, we have

$$\begin{aligned}
\hat{\boldsymbol{\mu}}_m - \boldsymbol{\mu} =& \left(D_m^\top \mathscr{D}_m\right)^{-1} \mathscr{D}_m^\top \boldsymbol{r}_m - \boldsymbol{\mu} \\
=& \left(D_m^\top \mathscr{D}_m\right)^{-1} \mathscr{D}_m^\top (\boldsymbol{w} + \mathscr{D}_m \boldsymbol{\mu}) - \boldsymbol{\mu} \\
=& \mathscr{V}_m^{-1} \mathscr{D}_m^\top \boldsymbol{w} \\
=& \mathscr{V}_m^{-1} \boldsymbol{z}. \qquad \text{(A.1)}
\end{aligned}$$

Therefore, $\|\hat{\boldsymbol{\mu}}_m - \boldsymbol{\mu}\|_{\mathscr{V}_m}^2 = \|\boldsymbol{z}\|_{\mathscr{V}_m^{-1}}^2$. In order to get a tail bound on the quantity $\|\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}\|_{\mathscr{V}_m}$, we can instead work on $\|\boldsymbol{z}\|_{\mathscr{V}_m^{-1}}$. Consider $\mathbb{E}\left[\exp\left(\boldsymbol{\alpha}^\top \boldsymbol{z}\right)\right]$, the moment generating function of $\boldsymbol{z}$ with respect to $\boldsymbol{\alpha} \in \Re^d$, by the R-sub-Gaussian property of $\eta_1, \ldots, \eta_{nd}$,

$$\begin{aligned}
& \mathbb{E}\left[\exp\left(\boldsymbol{\alpha}^\top \boldsymbol{z}_t\right)\right] \\
=& \mathbb{E}\left[\exp\left(\sum_{s=1}^{md}\left(\boldsymbol{\alpha}^\top \boldsymbol{a}_{I_s}\right)\eta_s\right)\right] \\
=& \mathbb{E}\left[\mathbb{E}_{\eta_{md}}\left[\exp\left(\sum_{s=1}^{md}\left(\boldsymbol{\alpha}^\top \boldsymbol{a}_{I_s}\right)\eta_s\right)\middle| \boldsymbol{a}_{I_1}, \ldots, \boldsymbol{a}_{I_{md}}, \eta_1, \ldots, \eta_{md-1}\right]\right]
\end{aligned}$$

$$
\begin{aligned}
&\leq \mathbb{E}\left[\exp\left(\sum_{s=1}^{md-1}\left(\boldsymbol{\alpha}^\top \boldsymbol{a}_{I_s}\right)\eta_s\right)\right]\exp\left(\frac{\left(\boldsymbol{\alpha}^\top \boldsymbol{a}_{I_{md}}R\right)^2}{2}\right)\\
&\quad\vdots\\
&\leq \exp\left(\sum_{s=1}^{md}\frac{\left(\boldsymbol{\alpha}^\top \boldsymbol{a}_{I_s}R\right)^2}{2}\right)\\
&= \exp\left(\frac{R^2}{2}\boldsymbol{\alpha}^\top \mathscr{V}_m\boldsymbol{\alpha}\right). \qquad \text{(A.2)}
\end{aligned}
$$

One can rewrite this to

$$\mathbb{E}\left[\exp\left(\boldsymbol{\alpha}^\top \boldsymbol{z}-\frac{R^2}{2}\boldsymbol{\alpha}^\top \mathscr{V}_m\boldsymbol{\alpha}\right)\right]\leq 1.$$

We further define

$$M_{\boldsymbol{\alpha}} = \exp\left(\boldsymbol{\alpha}^\top \boldsymbol{z}-\frac{R^2}{2}\boldsymbol{\alpha}^\top \mathscr{V}_m\boldsymbol{\alpha}\right)$$

and

$$\overline{M} = \int M_{\boldsymbol{\alpha}} h(\boldsymbol{\alpha})d\boldsymbol{\alpha},$$

where $h(\cdot)$ is the density of the $\mathscr{N}\left(0, \mathscr{V}_m^{-1}/R^2\right)$. Now we have

$$
\begin{aligned}
\overline{M} =& \frac{R^d}{\sqrt{(2\pi)^d \det \mathscr{V}_m^{-1}}}\int \exp\left(\boldsymbol{\alpha}^\top \boldsymbol{z}-R^2\boldsymbol{\alpha}^\top \mathscr{V}_m\boldsymbol{\alpha}\right)d\boldsymbol{\alpha}\\
=& \frac{R^d}{\sqrt{(2\pi)^d \det \mathscr{V}_m^{-1}}}\int \exp\left(\frac{\boldsymbol{z}\mathscr{V}_m^{-1}\boldsymbol{z}}{4R^2}-R^2\left\|\boldsymbol{\alpha}-\frac{\mathscr{V}_m^{-1}\boldsymbol{z}}{2R^2}\right\|_{\mathscr{V}}^2\right)d\boldsymbol{\alpha}\\
=& \frac{1}{2^{d/2}}\exp\left(\frac{\|\boldsymbol{z}\|_{\mathscr{V}_m^{-1}}^2}{4R^2}\right). \qquad \text{(A.3)}
\end{aligned}
$$

Note that

$$\mathbb{E}[\overline{M}] = \int \mathbb{E}[M_{\boldsymbol{\alpha}}]h(\boldsymbol{\alpha})d\boldsymbol{\alpha}\leq 1,$$

we have

$$\mathbb{E}\left[\exp\left(\frac{\|\boldsymbol{z}\|_{\mathscr{V}_m^{-1}}^2}{4R^2}\right)\right]\leq 2^{d/2},$$

and thus

$$\Pr\left(\|\boldsymbol{z}\|^2_{\mathscr{V}_m^{-1}} \geq R^2\left(2\ln(2)d + 4\ln\delta^{-1}\right)\right) \leq \delta \tag{A.4}$$

by Chernoff Bound. Therefore, we have shown that

$$\Pr\left(\|\hat{\boldsymbol{\mu}} - \hat{\boldsymbol{\mu}}\|_{\mathscr{V}_m} \geq R\sqrt{2\ln(2)d + 4\ln\delta^{-1}}\right) \leq \delta. \tag{A.5}$$

□

A.2 Proof of Lemma 4

From Lemma 3, we have the probability that the difference between $\hat{\boldsymbol{\mu}}_m$ and $\boldsymbol{\mu}$ under the $\mathscr{V}_m$ norm is not less than $R\sqrt{2\ln(2)d+4\ln\delta^{-1}}$ is at most δ, *i.e.*,

$$\Pr\left(\|\hat{\boldsymbol{\mu}}_m-\boldsymbol{\mu}\|_{\mathscr{V}_m}\geq R\sqrt{2\ln(2)d+4\ln\delta^{-1}}\right)\leq\delta.$$

Equivalently, we have with probability at least $1-\delta$.

$$(\hat{\boldsymbol{\mu}}_m-\boldsymbol{\mu})^\top\mathscr{V}_m(\hat{\boldsymbol{\mu}}_m-\boldsymbol{\mu})\leq\gamma^2, \tag{A.6}$$

where

$$\gamma=R\sqrt{2\ln(2)d+4\ln\delta^{-1}}.$$

By definition of $\mathscr{B}$ and $\mathscr{V}_m$, we know that $\mathscr{V}_m=m\mathscr{B}\mathscr{B}^\top$. Denoting $\boldsymbol{x}=\mathscr{B}^\top(\hat{\boldsymbol{\mu}}_m-\boldsymbol{\mu})$, (A.6) indicates that with probability at least $1-\delta$,

$$\|\boldsymbol{x}\|^2\leq\frac{\gamma^2}{m}. \tag{A.7}$$

As $\mathscr{B}$ is the 2-approximate barycentric spanners of $\mathscr{A}$, for any $\boldsymbol{a}\in\mathscr{A}$, there exists some $\boldsymbol{\nu}_{\boldsymbol{a}}\in[-2,2]^d$, such that $\mathscr{B}\boldsymbol{\nu}_{\boldsymbol{a}}=\boldsymbol{a}$. Therefore,

$$\begin{aligned}\langle\boldsymbol{a},\hat{\boldsymbol{\mu}}_m-\boldsymbol{\mu}\rangle^2=&(\hat{\boldsymbol{\mu}}_m-\boldsymbol{\mu})^\top\mathscr{B}\boldsymbol{\nu}_{\boldsymbol{a}}\boldsymbol{\nu}_{\boldsymbol{a}}^\top\mathscr{B}^\top(\hat{\boldsymbol{\mu}}_m-\boldsymbol{\mu})\\=&\boldsymbol{x}^\top\boldsymbol{\nu}_{\boldsymbol{a}}\boldsymbol{\nu}_{\boldsymbol{a}}^\top\boldsymbol{x}\\=&\left(\boldsymbol{x}^\top\boldsymbol{\nu}_{\boldsymbol{a}}\right)^2\\\leq&\|\boldsymbol{x}\|^2\|\boldsymbol{\nu}_{\boldsymbol{a}}\|^2\\\leq&\frac{4d\gamma^2}{m},\end{aligned}$$

where the second last inequality follows from Cauchy-Schwarz inequality. □

A.3 Proof of Theorem 5

We prove the instance dependent and worst case regrets separately.

1. Obtaining Instance Dependent Regret: To keep track of the regret incurred by EC algorithm in the committing stage, we consider the event $\{\boldsymbol{a}_{I_{t_0}} \neq \boldsymbol{a}_*\}$ with $t_0 = n \cdot d + 1$. This means that EC algorithm either has a large overestimate on $\langle \boldsymbol{a}_*, \boldsymbol{\mu} \rangle$, that is

$$\langle \boldsymbol{a}_*, \hat{\boldsymbol{\mu}} \rangle \geq \langle \boldsymbol{a}_*, \boldsymbol{\mu} \rangle + \frac{\Delta_{I_{t_0}}}{2}, \tag{A.8}$$

or a major underestimate on $\langle \boldsymbol{a}_{I_{t_0}}, \boldsymbol{\mu} \rangle$, that is

$$\langle \boldsymbol{a}_{I_{t_0}}, \hat{\boldsymbol{\mu}} \rangle \leq \langle \boldsymbol{a}_{I_{t_0}}, \boldsymbol{\mu} \rangle - \frac{\Delta_{I_{t_0}}}{2}. \tag{A.9}$$

From Lemma 4, we set

$$\frac{4d\gamma^2}{n} = \frac{\Delta_{\min}^2}{4}, \tag{A.10}$$

which is equivalent to

$$\delta = \exp\left(\ln(2)d - \frac{\Delta_{\min}^2 n}{64dR^2}\right). \tag{A.11}$$

This further indicates

$$\begin{aligned}
\Pr\left(\langle \boldsymbol{a}_*, \hat{\boldsymbol{\mu}} \rangle \geq \langle \boldsymbol{a}_*, \boldsymbol{\mu} \rangle + \frac{\Delta_{I_{t_0}}}{2}\right) &\leq \Pr\left(\langle \boldsymbol{a}_*, \hat{\boldsymbol{\mu}} \rangle \geq \langle \boldsymbol{a}_*, \boldsymbol{\mu} \rangle + \frac{\Delta_{\min}}{2}\right) \\
&\leq \Pr\left(\langle \boldsymbol{a}_*, \hat{\boldsymbol{\mu}} - \boldsymbol{\mu} \rangle^2 \geq \frac{\Delta_{I_{\min}}^2}{4}\right) \\
&= \Pr\left(\langle \boldsymbol{a}_*, \hat{\boldsymbol{\mu}} - \boldsymbol{\mu} \rangle^2 \geq \frac{8d\gamma^2}{n}\right) \\
&\leq \exp\left(\ln(2)d - \frac{\Delta_{\min}^2 n}{64dR^2}\right).
\end{aligned}$$

Similarly,

$$\Pr\left(\langle \boldsymbol{a}_{I_{t_0}}, \hat{\boldsymbol{\mu}}\rangle \leq \langle \boldsymbol{a}_{I_{t_0}}, \boldsymbol{\mu}\rangle - \frac{\Delta_{I_{t_0}}}{2}\right) \leq \exp\left(\ln(2)d - \frac{\Delta_{\min}^2 n}{64dR^2}\right).$$

By union bound, we have

$$\Pr\left(\left\{\boldsymbol{a}_{I_{t_0}} \neq \boldsymbol{a}_*\right\}\right) \leq 2\exp\left(\ln(2)d - \frac{\Delta_{\min}^2 n}{64dR^2}\right). \tag{A.12}$$

the expected regret of EC algorithm is upper bounded as

$$\begin{aligned}
&\mathbb{E}\left[\text{Regret(Explore-then-Commit alorithm)}\right] \\
\leq& n\cdot d\Delta_{\max} + 2(T - n\cdot d)\Delta_{\max}\exp\left(\ln(2)d - \frac{\Delta_{\min}^2 n}{64dR^2}\right) \\
\leq& n\cdot d\Delta_{\max} + 2T\Delta_{\max}\exp\left(\ln(2)d - \frac{\Delta_{\min}^2 n}{64dR^2}\right).
\end{aligned} \tag{A.13}$$

Setting

$$n = \frac{64dR^2\ln(dT) + 64\ln(2)d^2R^2}{\Delta_{\min}^2 T} \tag{A.14}$$

brings us

$$\begin{aligned}
&\mathbb{E}\left[\text{Regret}_T\text{(EC algorithm)}\right] \\
\leq& \frac{64d^2R^2\Delta_{\max}\ln(dT) + 64\ln(2)d^3R^2\Delta_{\max}}{\Delta_{\min}^2} + 2.
\end{aligned} \tag{A.15}$$

2. Obtaining Worst Case Regret:

With probability at least $1-\delta$, we have $|\langle \boldsymbol{a}_k, \hat{\boldsymbol{\mu}} - \boldsymbol{\mu}\rangle| \leq \sqrt{4d\gamma^2/n}$ holds for all $\boldsymbol{a}_k \in \mathscr{A}$. Therefore, with probability at least $1-\delta$,

$$\begin{aligned}
\langle \boldsymbol{a}_{I_{t_0}}, \boldsymbol{\mu}\rangle \leq& \langle \boldsymbol{a}_{I_{t_0}}, \hat{\boldsymbol{\mu}}\rangle + \sqrt{4d\gamma^2/n} \\
\leq& \langle \boldsymbol{a}_*, \hat{\boldsymbol{\mu}}\rangle + \sqrt{4d\gamma^2/n} \\
\leq& \langle \boldsymbol{a}_*, \boldsymbol{\mu}\rangle + 2\sqrt{4d\gamma^2/n}
\end{aligned}$$

Therefore, the expected regret of EC algorithm is upper bounded as

$$\begin{aligned}
&\mathbb{E}\left[\text{Regret(Explore-then-Commit alorithm)}\right] \\
\leq & n \cdot d\Delta_{\max} + \sqrt{\frac{4d\gamma^2 T^2}{n}} + T\delta \\
\leq & nd^2 + \sqrt{\frac{4dR(2\ln(2)d + 4\ln\delta^{-1})T^2}{n}} + T\delta \\
\leq & 6d^{\frac{5}{4}} T^{\frac{2}{3}} \sqrt{R\ln T} + 1. \qquad \text{(A.16)}
\end{aligned}$$

Here we take $n = d^{-3/4}T^{2/3}$, and $\delta = 1/T$. □

A.4 Proof of Lemma 6

On one hand, if (3.5) holds, then the detected path $\boldsymbol{a}_k$ cannot be optimal, *i.e.*, given

$$\langle \boldsymbol{a}_k, \hat{\boldsymbol{\mu}}_m \rangle - \langle \tilde{\boldsymbol{a}}_m, \hat{\boldsymbol{\mu}}_m \rangle \geq 2\tilde{\Delta}_m,$$

we have

$$\begin{aligned}
\langle \boldsymbol{a}_k, \boldsymbol{\mu} \rangle \geq & \langle \boldsymbol{a}_k, \hat{\boldsymbol{\mu}}_m \rangle - \tilde{\Delta}_m && \text{(A.17)} \\
> & \langle \tilde{\boldsymbol{a}}_m, \hat{\boldsymbol{\mu}}_m \rangle + \tilde{\Delta}_m && \text{(A.18)} \\
\geq & \langle \tilde{\boldsymbol{a}}_m, \boldsymbol{\mu} \rangle, && \text{(A.19)}
\end{aligned}$$

where inequalities (A.17) and (A.19) hold because we have conditioned on E, and inequality (A.18) follows from re-arranging the terms in (3.5). This implies that routing a packet via path $\boldsymbol{a}_k$ incurs more delay than that of $\tilde{a}_m$, and it thus cannot be optimal; On the other hand, this criterion (3.5) also promises that any path $\boldsymbol{a}_k$ with a gap

$$\Delta_k > 4\tilde{\Delta}_m$$

is detected after epoch m, *i.e.*,

$$\begin{aligned}
\langle \boldsymbol{a}_k, \hat{\boldsymbol{\mu}}_m \rangle - \tilde{\Delta}_m \geq & \langle \boldsymbol{a}_k, \boldsymbol{\mu} \rangle - 2\tilde{\Delta}_m && \text{(A.20)} \\
= & \langle \boldsymbol{a}_*, \boldsymbol{\mu} \rangle + \Delta_k - 2\tilde{\Delta}_m && \text{(A.21)} \\
> & \langle \boldsymbol{a}_*, \boldsymbol{\mu} \rangle + 2\tilde{\Delta}_m && \text{(A.22)} \\
\geq & \langle \boldsymbol{a}_*, \hat{\boldsymbol{\mu}}_m \rangle + \tilde{\Delta}_m && \text{(A.23)} \\
\geq & \langle \tilde{\boldsymbol{a}}_m, \hat{\boldsymbol{\mu}}_m \rangle + \tilde{\Delta}_m && \text{(A.24)}
\end{aligned}$$

where inequalities (A.20) and (A.23) hold because we have conditioned on E, equality (A.21) holds by definition of Δ_k, inequality (A.22) follows from the assumption that $\Delta_k >$

$4\tilde{\Delta}_m$, and inequality (A.24) follows from the optimality of $\tilde{\boldsymbol{a}}_m$. This is equivalent to

$$\langle \boldsymbol{a}_k, \hat{\boldsymbol{\mu}}_m \rangle - \langle \tilde{\boldsymbol{a}}_m, \hat{\boldsymbol{\mu}}_m \rangle > 2\tilde{\Delta}_m, \tag{A.25}$$

and $\boldsymbol{a}_k$ is detected. □

A.5 Proof of Theorem 7

We begin by decomposing the regret as following

$$\begin{aligned}
&\mathbb{E}\left[\text{Regret}_T\left(\text{TTC algorithm}\right)\right] \\
=&\mathbb{E}\left[\text{Regret}_T\left(\text{TTC algorithm}\right)|\neg E\right]\Pr(\neg E) \\
&+\mathbb{E}\left[\text{Regret}_T\left(\text{TTC algorithm}\right)|E\right]\Pr(E) \\
\leq&\mathbb{E}\left[\text{Regret}_T\left(\text{TTC algorithm}\right)|\neg E\right] \\
&+T^2\delta\mathbb{E}\left[\text{Regret}_T\left(\text{TTC algorithm}\right)|E\right],
\end{aligned}$$

and then distinguish the following two cases:

1. Analyzing $\mathbb{E}\left[\text{Regret}_T\left(\text{TTC algorithm}\right)|\neg E\right]$

Under this case, all the sub-optimal arms should be eliminate when

$$\tilde{\Delta}_m \leq \Delta_{\min}/4,$$

or

$$m = (256dR^2\ln\delta^{-1} + 128\ln(2)d^2R^2)/\Delta_{\min}^2.$$

Therefore,

$$\begin{aligned}
&\mathbb{E}\left[\text{Regret}_T\left(\text{TTC algorithm}\right)|\neg E\right] \\
\leq&\frac{256d^2R^2\Delta_{\max}\ln\delta^{-1} + 128\ln(2)d^3R^2\Delta_{\max}}{\Delta_{\min}^2}.
\end{aligned} \tag{A.26}$$

2. Analyzing $T^2\delta\mathbb{E}\left[\text{Regret}_T\left(\text{TTC algorithm}\right)|E\right]$

We know that the regret of each round is at most $\Delta_{\max}$, and the total regret can be trivially upper bounded by $T\Delta_{\max}$. Therefore,

$$T^2\delta\mathbb{E}\left[\text{Regret}_T\left(\text{TTC algorithm}\right)|E\right] \leq T^3\Delta_{\max}\delta. \tag{A.27}$$

Combining the above two cases, we can sett δ to T^{-3}, and have

$$\begin{aligned}&\mathbb{E}\left[\text{Regret}_T\left(\text{TTC algorithm}\right)\right]\\ \leq&\frac{768d^2R^2\Delta_{\max}\ln T+128\ln(2)d^3R^2\Delta_{\max}}{\Delta_{\min}^2}+1.\end{aligned}\tag{A.28}$$

□

A.6 Proof of Theorem 8

Conditioned on E, The first half of the theorem follows directly from Theorem 7. Now suppose $\Delta_{\min} \leq d^{3/2}T^{-1/4}$, then the MTTC algorithm switches to the alternative. The regret of in the first n epochs can be upper bounded as

$$\Delta_{\max} d\overline{n} \leq d^2\overline{n} = \sqrt{T}R^2\left(32\ln(2)d + 48\ln T\right) \tag{A.29}$$

The regret of running the alternative can be upper bounded as $C'(d\sqrt{T\ln T})$ with some absolute constant C', and the expected regret conditioned on $\neg E$ is constant. The conclusion then follows. □

A.7 Proof of Lemma 9

We first note that in each iteration of while-loop in Algorithm 4, either u is increased by 1 or the variable Flag is set to False. Therefore, after at most d iterations, the algorithm terminates. We then see that the statement $\mathscr{B}$ has linearly independent columns holds trivially as $\mathscr{C}_u$ has linearly independent columns by virtue of our algorithm, and $\mathscr{B}$ is just a sub-matrix of $\mathscr{C}_u$.

As an intermediate step, we show $\text{rank}(\mathscr{B}) = \text{rank}(\mathscr{A})$. Since $\mathscr{B}$ is always a subset of $\mathscr{A}$, the rank of $\mathscr{B}$ cannot exceed that of $\mathscr{A}$. Now if $\text{rank}(\mathscr{B}) < \text{rank}(\mathscr{A}) = d_0$, then there must exists an $\boldsymbol{a} \in \mathscr{A}$, such that $\boldsymbol{a}$ is linearly independent of the columns of $\mathscr{B}$. We declare that this is impossible once the algorithm terminate after u iterations. Upon termination, the matrix $(\boldsymbol{a}, \mathscr{C}_u)$ has rank d as $\mathscr{C}_u$ is full rank. By definition of $\boldsymbol{a}$, the rank of $(\boldsymbol{a}, \mathscr{B})$ is $\text{rank}(B)+1 \leq d_0 < d$, and therefore, we must be able to start from $(\boldsymbol{a}, \mathscr{B})$, and add columns from $\mathscr{C}_u(:, [u+1, d])$ to form a d-by-d full rank matrix by basis extension theorem. This is equivalent to replace a column of $\mathscr{C}_u(:, [u+1, d])$ with $\boldsymbol{a}$ while keeping resulted matrix full rank, which is exactly step (2) of the procedure. This means the algorithm should not terminate, and it is a contradiction. Therefore, $\text{rank}(\mathscr{B}) = \text{rank}(\mathscr{A})$.

Now for any $\boldsymbol{a} \in \mathscr{A}$, if $\boldsymbol{a}$ cannot be expressed as linearly combination of columns of $\mathscr{B}$, then adding $\boldsymbol{a}$ to $\mathscr{B}$ has rank $d_0 + 1$. As $(\mathscr{B}, \boldsymbol{a})$ is a sub-matrix of $\mathscr{A}$, we have

$$d_0 + 1 = \text{rank}\,(\mathscr{B}, \boldsymbol{a}) \leq \text{rank}(\mathscr{A}) = d_0,$$

which is a contradiction. □

A.8 Proof of Lemma 10

We first show that the noise vector $\boldsymbol{w}_t = (\eta_1, \ldots, \eta_t)$ is R-sub-Gaussian for all $t \in [T]$. It is easy to see that $\mathbb{E}[\boldsymbol{w}_t] = 0$, and for any $\boldsymbol{\alpha} \in \Re^t$, we have

$$\begin{aligned}
&\mathbb{E}\left[\exp\left(\boldsymbol{\alpha}^\top \boldsymbol{w}_t\right)\right] \\
=&\mathbb{E}\left[\mathbb{E}\left[\exp\left(\sum_{s=1}^{t-1}\alpha_s\eta_s + \alpha_t\eta_t\right)\middle| \boldsymbol{a}_{I_1}, \ldots, \boldsymbol{a}_{I_t}, \eta_1, \ldots, \eta_{t-1}\right]\right] \\
\leq&\mathbb{E}\left[\mathbb{E}\left[\exp\left(\frac{\alpha_t^2 R^2}{2}\right)\exp\left(\sum_{s=1}^{t-1}\alpha_s\eta_s\right)\middle| \boldsymbol{a}_{I_1}, \ldots, \boldsymbol{a}_{I_t}, \eta_1, \ldots, \eta_{t-1}\right]\right] \\
=&\exp\left(\frac{\alpha_t^2 R^2}{2}\right)\mathbb{E}\left[\exp\left(\sum_{s=1}^{t-1}\alpha_s\eta_s\right)\right] \\
&\vdots \\
=&\exp\left(\frac{\|\boldsymbol{\alpha}\|^2 R^2}{2}\right). \qquad \text{(A.30)}
\end{aligned}$$

Now from Theorem 2.2 of [30], we have the probability that the difference between $\hat{\boldsymbol{\mu}}_m$ and $\boldsymbol{\mu}$ under the $\mathscr{V}_m$ norm is not less than $R\sqrt{32\ln(6)d_0 + 32\ln\delta^{-1}}$ is at most δ, *i.e.*,

$$\Pr\left(\|\hat{\boldsymbol{\mu}}_m - \boldsymbol{\mu}\|_{\mathscr{V}_m} \geq R\sqrt{32\ln(6)d_0 + 32\ln\delta^{-1}}\right) \leq \delta.$$

Equivalently, we have

$$(\hat{\boldsymbol{\mu}}_m - \boldsymbol{\mu})^\top \mathscr{V}_m (\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}) \leq \gamma^2, \qquad \text{(A.31)}$$

where

$$\gamma = R\sqrt{32\ln(6)d_0 + 32\ln\delta^{-1}}.$$

By definition of $\mathscr{B}$ and $\mathscr{V}_m$, we know that $\mathscr{V}_m = m\mathscr{B}\mathscr{B}^\top$. Denoting

$$\boldsymbol{x} = \mathscr{B}^\top(\hat{\boldsymbol{\mu}}_m - \boldsymbol{\mu}),$$

(A.31) indicates with probability at least $1-\delta$,

$$\|\boldsymbol{x}\|^2 \leq \frac{\gamma^2}{m}. \tag{A.32}$$

As $\mathscr{B}$ is the basis of $\mathscr{A}$, for any $\boldsymbol{a} \in \mathscr{A}$ there exists some $\boldsymbol{\nu_a}$, such that $\mathscr{B}\boldsymbol{\nu_a} = \boldsymbol{a}$. Therefore,

$$\begin{aligned}
\langle \boldsymbol{a}, \hat{\boldsymbol{\mu}}_m - \boldsymbol{\mu} \rangle^2 =& (\hat{\boldsymbol{\mu}}_m - \boldsymbol{\mu})^\top \mathscr{B}\boldsymbol{\nu_a}\boldsymbol{\nu_a}^\top \mathscr{B}^\top (\hat{\boldsymbol{\mu}}_m - \boldsymbol{\mu}) \\
=& \boldsymbol{x}^\top \boldsymbol{\nu_a}\boldsymbol{\nu_a}^\top \boldsymbol{x} \\
=& \left(\boldsymbol{x}^\top \boldsymbol{\nu_a}\right)^2 \\
\leq& \|\boldsymbol{x}\|^2 \|\boldsymbol{\nu_a}\|^2 \\
\leq& \frac{S\gamma^2}{m},
\end{aligned}$$

where the second last inequality follows from Cauchy-Schwarz inequality. □

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix B

Algorithm for Finding Barycentric Spanners

In this section, we briefly state the algorithm from [12] to illustrate how to construct barycentric spanners when the network is identifiable. The detail is shown in Algorithm 5. Here, $\mathscr{B}(:,-j)$ is the matrix $\mathscr{B}$ with the j^{th} column removed. Each iteration of the for- and while-loop requires two quantities, *i.e.*,

$$\arg\max_{\boldsymbol{a}\in\mathscr{A}} \det(\boldsymbol{a},\mathscr{B}(:,-j)), \tag{B.1}$$

$$\arg\max_{\boldsymbol{a}\in\mathscr{A}} -\det(\boldsymbol{a},\mathscr{B}(:,-j)), \tag{B.2}$$

to compute $\arg\max_{\boldsymbol{a}\in\mathscr{A}} \det|(\boldsymbol{a},\mathscr{B}(:,-j))|$, This can be done by two calls to the longest path algorithm for directed acyclic graphs. Specifically, the each coefficient of the linear function $\det(\boldsymbol{a},\mathscr{B}(:,-j)$ is the determinant of a $(d-1)$-by-$(d-1)$ sub-matrix of $\mathscr{B}(:,j)$ and can therefore be computed efficiently. Afterwards, we can set the each link's delay to the corresponding coefficient in G, and run the longest path algorithm over this network to find $\boldsymbol{a}$.

Algorithm 5 Basis Identification for Identifiable Networks [12]

Input: A set of paths $\mathscr{A}$.
Initialization: $\mathscr{B} \leftarrow I$.
for $j = 1, \ldots, d$ **do**
    $B(:, j) \leftarrow \arg\max_{\boldsymbol{a} \in \mathscr{A}} \det |\, (\boldsymbol{a}, \mathscr{B}(:, -j)) \,|$.
end for
while $\exists \boldsymbol{a} \in \mathscr{A}, j \in [d]$ *s.t.*, $\det |\, (\boldsymbol{a}, \mathscr{B}(:, -j)) \,| > 2 \det |\mathscr{B}|$ **do**
    $\mathscr{B}(:, j) \leftarrow \boldsymbol{a}$.
end whilereturn $\mathscr{B}$.

Bibliography

[1] Yasin Abbasi-Yadkori, Andras Antos, and Csaba Szepesvári. Forced-exploration based algorithms for playing in stochastic linear bandits. In *COLT Workshop on Online Learning with Limited Feedback*, 2009.

[2] Yasin Abbasi-Yadkori, David Pál, and Csaba. Szepesvári. Improved algorithms for linear stochastic bandits. In *NIPS*, 2011.

[3] Marc Abeille and Alessandro Lazaric. Linear thompson sampling revisited. In *AISTAT*, 2017.

[4] Jacob Abernethy, Elad Hazan, and Alexander Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *COLT*, 2009.

[5] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *ICML*, 2013.

[6] Jean-Yves Audibert, Sebastien Bubeck, and Gabor Lugosi. Minimax policies for combinatorial prediction games. In *COLT*, 2011.

[7] Jean-Yves Audibert, Sebastien Bubeck, and Gabor. Lugosi. Regret in online combinatorial optimization. In *Mathematics of Operations Research*, 2012.

[8] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. In *JMLR*, 2002.

[9] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *The Journal of Machine Learning Research*, 3:397–422, 2003.

[10] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 2002.

[11] Peter Auer and Ronald Ortner. Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem. In *Periodica Mathematica Hungarica*, 2010.

[12] Baruch Awerbuch and Robert Kleinberg. Online linear optimization and adaptive routing. In *STOC*, 2004.

[13] Richard Bellman. On a routing problem. In *Quart. Appl. Math.*, 1958.

[14] Sebastien Bubeck. Bandit theory, part ii. In *https://blogs.princeton.edu/imabandit/2016/05/13/bandit-theory-part-ii/*, 2016.

[15] Sebastien Bubeck and Ronen Eldan. The entropic barrier: a simple and optimal universal self-concordant barrier. In *COLT*, 2015.

[16] Wei Chen, Yajun Want, and Yang Yuan. Combinatorial multi-armed bandit: General framework, results and applications. In *ICML*, 2013.

[17] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *AISTATS*, 2011.

[18] Alon Cohen, Tamir Hazan, and Tomer Koren. Tight bounds for bandit combinatorial optimization. In *COLT*, 2017.

[19] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to algorithms. In *MIT Press*, 2009.

[20] Varsha Dani, Thomas Hayes, and Sham Kakade. Stochastic linear optimization under bandit feedback. In *COLT*, 2008.

[21] Edsger Dijkstra. A note on two problems in connexion with graphs. In *Num. Math.*, 1959.

[22] Yi Gai, Bhaskar Krishnamachari, and Rahul Jain. Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations. In *IEEE/ACM Transactions on Networking*, 2012.

[23] John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

[24] Elad Hazan, Zohar Karnin, and Raghu Meka. Volumetric spanners: An efficient exploration basis for learning. In *Journal of Machine Learning Research*, 2016.

[25] Branislav Kveton, Zheng Wen, Azin Ashkan, and Csaba Szepesvári. Tight regret bounds for stochastic combinatorial semi-bandits. In *AISTATS*, 2015.

[26] T. L. Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.

[27] Tor Lattimore and Csaba Szepesvari. Bandit algorithms. In *Cambridge University Press, Available at: http://banditalgs.com*, 2018.

[28] Keqin Liu and Qing Zhao. Adaptive shortest-path routing under unknown and stochastically varying link states. In *WiOpt*, 2012.

[29] Liang Ma, Ting He, Kin K. Leung, Ananthram Swami, and Don Towsley. Identifiability of link metrics based on end-to-end path measurements. In *IMC*, 2013.

[30] Philippe Rigollet and J. Hutter. High dimensional statistics. In *http://www-math.mit.edu/ rigollet/PDFs/RigNotes17.pdf*, 2017.

[31] Herbert Robbins. Some aspects of the sequential design of experiments. *Bull. Amer. Math. Soc.*, 58(5):527–535, 1952.

[32] Paat Rusmevichientong and John Tsitsiklis. Linearly parameterized bandits. In *Mathematics of Operations Research*, 2010.

[33] Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. In *Mathematics of Operations Research*, 2014.

[34] Aleksandrs Slivkins. Introduction to multi-armed bandits. In *http://slivkins.com/work/MAB-book.pdf*, 2017.

[35] Mohammad Sadegh Talebi, Zhenhua Zou, Richard Combes, Alexandre Proutiere, and Mikael Johansson. Stochastic online shortest path routing: The value of feedback. In *IEEE Transactions on Automatic Control*, 2018.