Title

Advancing Reinforcement Learning: Multi-Agent Optimization, Opportunistic Exploration, and Causal Interpretation

Author

Wang, Xiaoxiao

Publication Date

2024

Peer reviewed|Thesis/dissertation

Advancing Reinforcement Learning: Multi-Agent Optimization, Opportunistic Exploration, and Causal Interpretation

By

XIAOXIAO WANG

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Xin Liu, Chair

Zhi Ding

Prasant Mohapatra

Committee in Charge

2024

Abstract

Reinforcement learning (RL), a critical subfield of machine learning, effectively models sequential decision-making scenarios for agents operating in various environments. Despite its extensive applications, RL encounters significant challenges in real world settings, particularly regarding limited data availability, the exploration-exploitation trade-off, and the lack of explainability. In this dissertation, I explore these issues through three distinct lenses. Firstly, I improve data efficiency in situations involving multiple agents or tasks [**1**, **2**, **3**]. Secondly, I propose opportunistic learning algorithms in environments with varying exploration costs [**4**, **5**]. Thirdly, I interpret the agent's learned policy through causal explanations [**6**, **7**].

The following sections outline the contributions. Initially, I study online global optimization in multi-agent situations. Cellular network configuration is a suitable application area experiencing these challenges, including the scarcity of diverse historical data, constrained experimental budgets imposed by network operators, and highly complex and unknown network performance functions. To overcome these challenges, I introduce an online-learning-based joint-optimization algorithm combining neural network regression with Gibbs sampling, which considerably outperforms distributed Q-learning in overall performance and ramp-up time. By leveraging similarities among tasks/base stations, I propose a kernel-based multi-task contextual bandit algorithm with the similarity estimated via conditional kernel embedding. These algorithms notably outperform the default cellular network configuration and the respective baseline algorithms.

Next, I focus on opportunistic learning, where the exploration cost in RL varies based on different environmental conditions. Given that exploration cost directly impacts the regret of selecting a sub-optimal action, I design the learning strategy to explore more when the cost is low and exploit when the cost is high. I propose an AdaLinUCB algorithm for opportunistic contextual bandits to balance the exploration-exploitation trade-off adaptively. My algorithm significantly

outperforms existing contextual bandit algorithms in scenarios with large exploration cost fluctuations. I further develop two algorithms OppUCRL2 and OppPSRL for the finite-horizon episodic Markov decision process, demonstrating the benefits of opportunistic RL. My algorithms balance the exploration-exploitation trade-off dynamically through a variation factor dependent optimism, leading to superior performance. These results are supported by theoretical regret bound analyses ensuring their performance.

Lastly, I aim to enhance RL's interpretability by providing causal explanations. My approach quantifies the causal influence of states on actions and their temporal impact, thereby surpassing associative methods in RL policy explanation. I propose a mechanism to quantify the individual-level causal counterfactual path-specific importance score for a structural causal model. This mechanism effectively evaluates causal influence in decision chains, allowing us to comprehend better how a specific decision variable influences an outcome variable.

Acknowledgments

This dissertation and the work invested in it would not have been possible without the support and nurturing of many individuals.

First and foremost, I would like to express my deepest appreciation to my advisor, Prof. Xin Liu. Her insightful inspiration and kind encouragement motivated me to overcome challenges and keep moving forward. I am truly grateful for her help in selecting research directions, discussing technical details, revising manuscripts, and providing financial support. Beyond her academic advising, she has also offered considerable assistance and support in my personal life.

My sincere appreciation also extends to Prof. Zhaodan Kong and Prof. Xin Chen (Georgia Institute of Technology) for their valuable feedback on my research, enriching my work with diverse perspectives. I am equally thankful to my dissertation and qualification committee members, Prof. Zhi Ding, Prof. Prasant Mohapatra (University of South Florida), Prof. Ian Davidson and Prof. Lifeng Lai, for their insightful comments and guidance throughout my qualifying exam and dissertation.

I would like to express my heartfelt thanks to my family members, Huimin Wang, Huifang Wang, and Xing Wei. Their unwavering support and encouragement have been the cornerstone of my journey. Their presence and belief in me have been a source of strength and motivation, enabling me to persevere through both challenges and triumphs.

Lastly, I would like to thank all my friends and colleagues at the University of California, Davis, especially Xueying Guo, Jiaxin Ding, Husen Wu, Fanyu Meng, Shahbaz Rezaeii, Jeonghoon Kim, Rex Liu, Yongshuai Liu, Chao Huang, Taeyeong Choi, Albara Ramli, Ziwen Kan, Songyang Zhang, and Xingcan Li. Their companionship and support have significantly enriched my academic journey.

Contents

Abstract ii

Acknowledgments iv

Chapter 1. Introduction 1

1.1. Contributions 2

1.2. Related Work 10

Chapter 2. Multi-Agent Learning-Based Joint Optimization: A Cellular Network Configuration Application 15

2.1. Introduction 15

2.2. Related Work 17

2.3. System Model and Problem Formulation 19

2.4. Learning Utility Function 22

2.5. Online-Learning-Based Joint Optimization 27

2.6. Numerical Results 36

2.7. Conclusion 42

2.8. Appendix 42

Chapter 3. Kernel-Based Multi-Task Contextual Bandits 45

3.1. Introduction 45

3.2. Related Work 48

3.3. System Model and Problem Formulation 49

3.4. Methodology 52

3.5. Theoretical Analysis 60

3.6. Evaluation 63

3.7. Conclusion 69

Chapter 4. Opportunistic Learning for Contextual Bandits 70
4.1. Introduction 70
4.2. Related Work 72
4.3. System Model 73
4.4. Adaptive LinUCB 75
4.5. Performance Analysis 77
4.6. Numerical Results 83
4.7. Conclusions 86
4.8. Appendix 86

Chapter 5. Opportunistic Learning for Episodic Reinforcement Learning 107
5.1. Introduction 107
5.2. Related Work 109
5.3. Problem Formulation 109
5.4. Opportunistic Reinforcement Learning Algorithm 111
5.5. Regret Analysis for OppUCRL2 114
5.6. Experimental Evaluation 117
5.7. Discussion 120
5.8. Conclusion 120

Chapter 6. Causal Explanation for Reinforcement Learning: State and Temporal Importance 121
6.1. Introduction 121
6.2. Related Work 123
6.3. Preliminaries 124
6.4. Problem Formulation 126
6.5. Explanation 127
6.6. Evaluation 132
6.7. Discussions 139
6.8. Conclusion 141

6.9. Appendix 142

Chapter 7. Causal Path-Specific Importance in SCM 153

7.1. Introduction 153

7.2. Preliminaries 155

7.3. Causal Effect along Different Pathways 157

7.4. Properties of Path-Specific Counterfactual Importance Score 159

7.5. Efficient Algorithm for the Most Important Path 161

7.6. Evaluation 163

7.7. Related Works 167

7.8. Discussions 169

7.9. Conclusion 170

7.10. Appendix 170

Chapter 8. Conclusion and Future Work 176

8.1. Conclusion 176

8.2. Future Work and Limitations 177

Bibliography 179

List of Figures

1.1 The structure of the dissertation. 1

1.2 Cellular network. 3

1.3 An Online-Learning-Based Joint-Optimization Algorithm. 4

1.4 A Kernel-Based Multi-Task Contextual Bandits Algorithm. 5

1.5 Causal graph between the state and action. 8

2.1 An illustration of cellular network configuration. 15

2.2 Online prediction accuracy for the cell utility function. 37

2.3 Total network throughput for different configuration algorithms. 40

2.4 Performance of configuration algorithms in a different setting. 41

3.1 Cellular network. 46

3.2 Multi-task online learning. 47

3.3 Similarity v.s MSE in 2-task regression. 65

3.4 Multi-task learning in synthetic data. 66

3.5 Similarity matrix among 105 BSs. 67

3.6 Multi-task learning v.s. Independent learning in real data. 68

4.1 Regret under Synthetic Scenarios. 84

4.2 Rewards for Yahoo! Today Module. 85

4.3 Regret under binary-valued variation factor. 101

4.4 Regret under beta distributed variation factor with a single threshold. 102

4.5 Regret under beta distributed variation factor with different values of $l^{(-)}$. 102

4.6 Regret under beta distributed variation factor with different values of $l^{(+)}$. 103

4.7 Performance Comparison with KernelUCB. 105

4.8 Performance comparison with different $l^{(-)}$ and $l^{(+)}$ values on Yahoo! Today Module. 106
4.9 Normalized variation factor demonstration. 106

5.1 Regret under binary variation factor scenarios. 115
5.2 RL environment: River Swim. 117
5.3 RL environment: Cliff Walking. 118
5.4 RL environment: Frozen Lake. 118
5.5 Regret under Beta variation factor scenarios. 119

6.1 Causal graph of the crop irrigation problem. 122
6.2 Example causal graph between the state and action. 128
6.3 Example of a one-step MDP. 131
6.4 The importance vector for the crop irrigation problem. 134
6.5 The collision avoidance problem and its corresponding SCM skeleton. 134
6.6 Trajectory and importance on the collision avoidance problem. 135
6.7 The skeleton of the Blackjack SCM. 136
6.8 A trajectory of a blackjack game and the result from running our mechanism. 137
6.9 The Q-value-based temporal importance on A_4. 138
6.10 Importance vector for state in crop irrigation problem. 143
6.11 The policy we use for the blackjack game. 144
6.12 The skeleton of the cascading SCM for a 5-step blackjack game. 144
6.13 The causal structure of lunar lander that includes previous state and actions. 146
6.14 A lunar lander trajectory instance. 147
6.15 The importance vector on lunar lander. 148
6.16 Difference between our method and the saliency map method for current-step features. 148
6.17 The importance vector of $\mathbf{S}^{(1)}$. 149
6.18 Sensitivity analysis on the collision avoidance problem. 150
6.19 Sensitivity analysis on the lunar lander environment. 151
6.20 Sensitivity analysis on the Blackjack environment. 152
6.21 The skeleton of SCM of the one step MDP. 152

7.1 The causal graph of job hiring for the physical demanding job. 154
7.2 The causal counterfactual path-specific importance score in job hiring problem. 165
7.3 Job hiring problem with different values of w_C 166
7.4 The causal graph of smoking effect on blood pressure. 166
7.5 Average effect for each path at the population level. 167
7.6 The causal counterfactual path-specific importance score in smoking impact problem. 168

List of Tables

2.1 Features of Sample Data. 24

2.2 Coefficient of Determination for Different Selections of Features. 25

2.3 Coefficient of Determination for Regression Methods. 26

2.4 Coefficient of Determination for Hyper-parameters. 26

2.5 Prediction accuracy for existing solutions. 38

3.1 Sample Data. 66

6.1 Importance vector on the environment in Fig. 6.3 132

List of Abbreviations

Abbreviations	Definition
3GPP	3rd Generation Partnership Project
BS	Base Station
CKE	Conditional Kernel Embedding
CQI	Channel Quality Indicator
DAG	Directed Acyclic Graph
DQN	Deep Q-Network
GIS	Geographic Information System
GPS	Global Positioning System
GPC-UCB	Gaussian Process Upper Confidence Bound
HCN	Hyper-Cellular Network
HLCS	High-Level Critical Situations
IoT	Internet of Things
i.i.d.	Independently and Identically Distributed
LTE	Long-Term Evolution
MABs	Multi-armed Bandits
MCMC	Markov Chain Monte Carlo
MDP	Markov Decision Process
MIMO-NOMA	Multiple Input Multiple Output - Non-Orthogonal Multiple Access
MSE	Mean Square Error
NUM	Network Utility Maximization
OFU	Optimism in the Face of Uncertainty
PCIs	Physical Cell Identifiers
PSRL	Posterior Sampling for Reinforcement Learning
QoS	Quality of Service
RKHS	Reproducing Kernel Hilbert Space

Abbreviations	Definition
RL	Reinforcement Learning
RLUs	Rectified Linear Units
RSRP	Reference Signal Received Power
RSRQ	Reference Signal Received Quality
SCM	Structural Causal Model
SDU	Service Data Unit
SINR	Signal-to-Interference-plus-Noise Ratio
SVR	Support Vector Regression
TCA	Transfer Component Analysis
TS	Thompson Sampling
UCB	Upper Confidence Bound
WCDMA	Wideband Code Division Multiple Access
XAI	Explainable Artificial Intelligence
XRL	Explainable Reinforcement Learning

CHAPTER 1

Introduction

Reinforcement learning (RL), a subfield of machine learning, has revolutionized the way we approach sequential decision-making problems [**8**]. The paradigm of RL, where agents continually learn optimal actions through interactions with their environment and resultant rewards or penalties, has been successfully utilized in a variety of applications, including game playing, such as AlphaGo from Google's DeepMind [**9**], robotics [**10**], and power system controls [**11**].

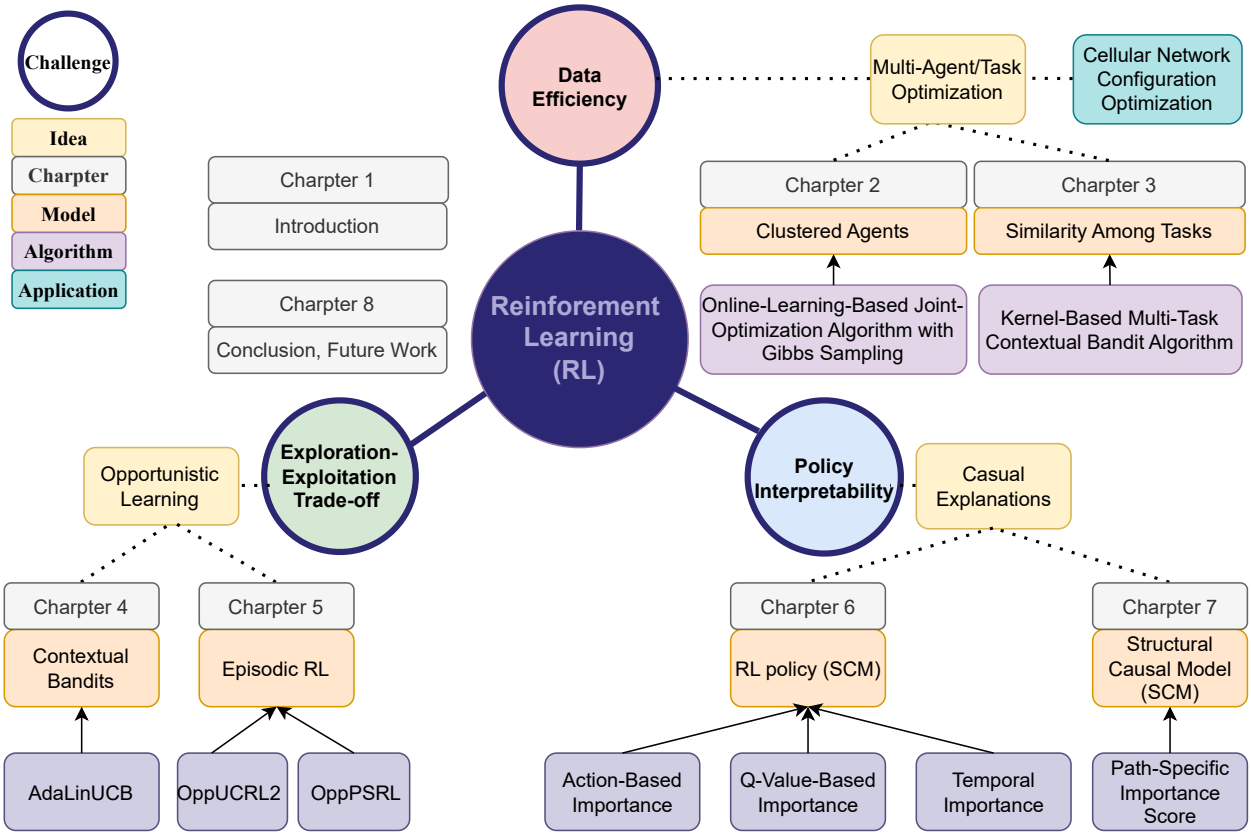


FIGURE 1.1. The structure of the dissertation.

However, real-world RL applications pose several formidable challenges, such as training with limited offline data, managing high-dimensional spaces, adhering to safety constraints, and providing real-time, explainable decision-making strategies, and accounting for multi-agent or multi-task scenarios [**12**]. This dissertation aims to address these challenges through three unique perspectives: bolstering data efficiency, implementing opportunistic exploration, and offering causal explanations as show in Fig. 1.1. Specifically, we enhance data utilization in multi-agent/task optimization by clustering agents and leveraging similarity among tasks. Our methods demonstrated effectively in optimizing cellular network configurations. Next, we propose opportunistic learning to balance exploration and exploitation in environments with fluctuating exploration costs. Finally, our work intends to shed light on RL policy decisions through causal explanations, quantifying the temporal causal influence of states on actions. Each of these components will be discussed in greater detail in the subsequent sections.

1.1. Contributions

1.1.1. Multi-Agent/Task Optimization with Application in Cellular Network Configuration. In contrast to deep reinforcement learning research conducted in simulated environments [**13**, **14**], real-world systems do not provide distinct training and evaluation environments. Training data is solely procured from the actual system, thereby eliminating the possibility for a separate exploration policy during this process. Given that all exploratory actions bear their own costs, it is imperative for the agent to act both satisfactorily and safely throughout its learning trajectory. For a systems, this necessitates limited exploration, leading to the generation of low-variance data - data encompassing a minimal state space. Many real-world systems are slow-moving, fragile, or expensive enough to render the data they produce valuable, thus demanding data-efficient policy learning.

In multi-agent or multi-task scenarios, these instances may be correlated and similar, thereby providing the potential for utilizing data or features from other agents or tasks to improve data efficiency. In this study, we adapt and apply learning algorithms to real network applications, specifically addressing the network configuration problem - the task of optimizing network performance by configuring the parameters of base stations, given the dynamics of the network.

Implementing online learning-based cellular network configuration in real-world systems presents distinct challenges, as illustrated in Figure 1.2. The traditional manual adjustment approach, guided by engineers' experience, often results in static configurations and fails to leverage the varying environments and conditions of different base stations. The pursuit of automation and optimization through online learning-based algorithms is complicated by the intricate, dynamic relationship between network configuration and performance, and the distinct characteristics of each base station. Moreover, operators must grapple with the exploitation-exploration trade-off; balancing the use of the best-known configurations for immediate gains against the need to experiment with novel configurations that may offer long-term performance improvements. This is further complicated by the disruptive nature of testing in cellular networks, where suboptimal configurations can lead to poor user experiences, aligning with the real-world reinforcement learning systems' demand for limited exploration and data-efficient policy learning. To address the challenges inherent in real-world system configurations, we introduce two online learning-based approaches that cater BSs as multiple agents or tasks:

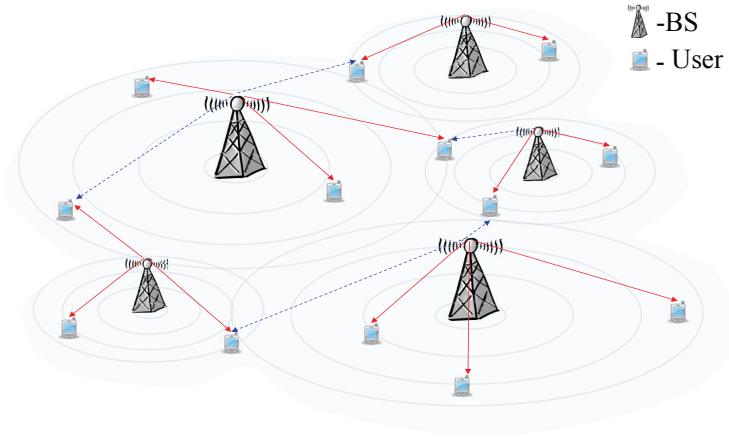


FIGURE 1.2. Cellular network.

(1) An Online-Learning-Based Joint-Optimization Algorithm This algorithm harnesses the capabilities of neural network regression and Markov Chain Monte Carlo (MCMC) methods, more specifically, Gibbs-sampling—an MCMC algorithm well-suited to the structure of cellular networks (Figure 1.3). The algorithm strives to maximize the overall network utility over time, acknowledging the impact of neighboring base stations (BSs). Recognizing the interaction among a

BS and its neighboring BSs, the utility function of a BS is defined based on its network state vector and network control within the same cluster, to enhance data efficiency. As maximizing overall network utility over time necessitates sequential decisions, the problem naturally falls within the domain of reinforcement learning. However, conventional reinforcement learning algorithms grapple with the broad state space and action set inherent in the cellular network configuration problem. To address this, we developed an online framework that integrates neural network regression with Gibbs-sampling theory, demonstrating prompt convergence to a local optimum. Evaluation on an industrial-grade cellular network simulator encompassing 87 BSs revealed that our proposed algorithm outperforms distributed Q-learning methods.

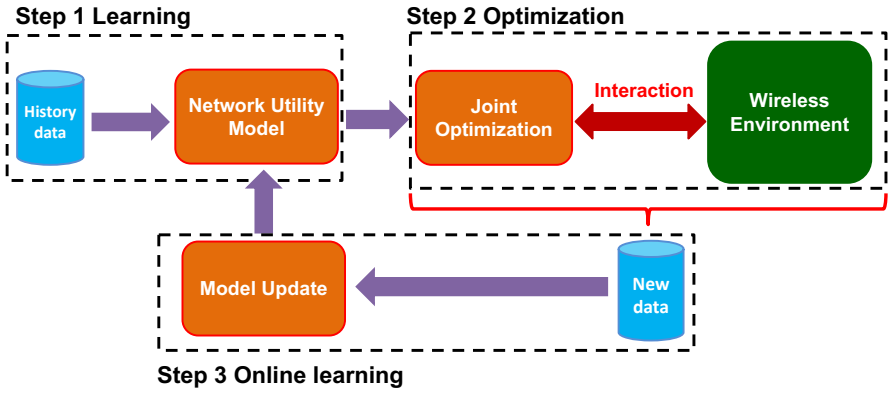


FIGURE 1.3. An Online-Learning-Based Joint-Optimization Algorithm in Cellular Network Configuration

(2) A Kernel-Based Multi-Task Contextual Bandits Algorithm This algorithm, illustrated in Figure 1.4, jointly considers multiple BSs and formulates the associated configuration problem within a multi-task online learning framework. The primary aim is to use information from multiple BSs to conjointly learn a model that maps the network state and configuration to performance. This model is then adapted to each BS based on its unique characteristics. The similarity among BSs is gauged by conditional kernel embedding. Additionally, the model enables BSs to balance the trade-off between the exploration and exploitation of different configurations. The algorithm's effectiveness was evaluated using both synthetic data and real industrial trace data comprising 105 BSs, outperforming bandits algorithms that do not employ multi-task learning by up to 70.8% and 64.8%, respectively. In related research [**15**], we proposed a collaborative

contextual bandits algorithm and demonstrated that transfer learning can significantly reduce prediction errors. Leveraging this theoretical result, we developed a practical algorithm that divides the learning model into a common homogeneous model, learned using all BSs' data, and a BS-specific feature that encapsulates the heterogeneous behavior of each individual BS. A live field test conducted in a metropolitan cellular network with over 1700 cells yielded a notable network performance improvement exceeding 20%.

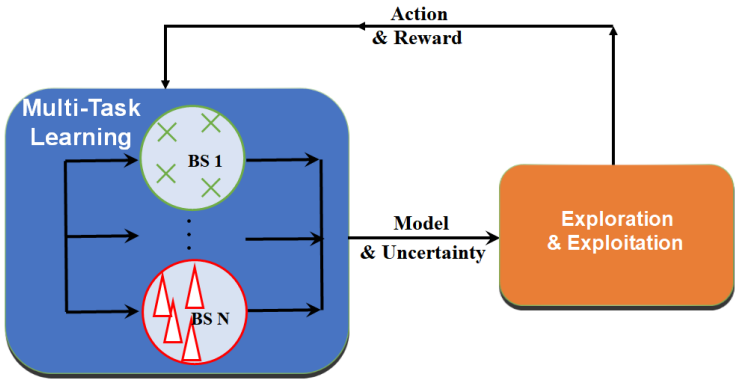


FIGURE 1.4. A Kernel-Based Multi-Task Contextual Bandits Algorithm.

1.1.2. Opportunistic Learning. Sequential decision-making problems such as contextual bandits (special case of RL) [**16**,**17**,**18**,**19**] and reinforcement learning (RL) [**20**,**21**] present inherent challenges, including the delicate trade-off between exploration (discovering new information) and exploitation (utilizing current knowledge). These issues become even more complex in opportunistic learning scenarios, where the exploration cost is not static, but rather, it varies over time and is situation-dependent.

To illustrate this, consider the case of a sequential recommender system in e-commerce. This system continually proposes a sequence of candidate products to users based on their preferences and browsing history, aiming to maximize the total click-through rate (i.e., the probability that a user will accept the recommendation). It is noteworthy that the actual monetary return from a successful recommendation can vary due to numerous factors, such as the differing purchasing power of users or their loyalty status (for instance, a diamond vs. a silver status). In this scenario, the ultimate goal is to maximize the overall monetary reward. It follows intuitively that when the potential monetary return from a recommendation is low, the monetary regret resulting from

suggesting a suboptimal product is likewise low, leading to a lower exploration cost. Conversely, high returns lead to high regret and high exploration cost. Given these unique dynamics, we proposed two novel learning frameworks.

(1) Opportunistic Contextual Bandits Opportunistic Contextual Bandits operate within a context where the exploration cost (or regret) fluctuates due to an external, time-varying factor, referred to as the variation factor. This factor is revealed before an arm is pulled, allowing the learning agent to adapt its approach accordingly. While this can be viewed as a special case of contextual bandits—with the variation factor forming part of the context—the general contextual bandit algorithms do not take advantage of the opportunistic nature of this problem, potentially leading to suboptimal performance.

As suggested by its name, in opportunistic contextual bandits, the variation of this external variation factor can be leveraged to reduce the actual regret. Further, besides the previous two examples, opportunistic contextual bandit algorithms can be applied to other scenarios that share these characteristics. We propose an Adaptive Upper-Confidence-Bound algorithm for opportunistic contextual bandits with Linear payoffs (AdaLinUCB). The algorithm is designed to dynamically balance the exploration-exploitation trade-off in opportunistic contextual bandits. To be best of our knowledge, this is the first work to study opportunistic learning for contextual bandits. We focus on the problem-dependent bound analysis here, which is a setting that allows a better bound to be achieved under stronger assumptions. To the best of our knowledge, such a bound does not exist for LinUCB in the existing literature. We prove problem-dependent bounds for both the proposed AdaLinUCB and the traditional LinUCB algorithms. Both algorithms have a regret upper bound of $O\left((\log T)^2\right)$, and the coefficient of the AdaLinUCB bound is smaller than that of LinUCB. Furthermore, using both synthetic and real-world large-scale dataset, we show that AdaLinUCB significantly outperforms other contextual bandit algorithms, under large exploration cost fluctuations.

(2) Opportunistic Reinforcement Learning We propose and study *opportunistic reinforcement learning*, a new paradigm of reinforcement learning problems where the regret of executing a suboptimal action depends on a varying cost referred to as *variation factor*, associated with the

environmental conditions. When the variation factor is low, so is the cost/regret of picking a suboptimal action and vice versa. Therefore, intuitively, we should explore more when the variation factor is low and exploit more when the variation factor is high. As its name suggests, in opportunistic RL, we leverage the opportunities of variation factor's dynamic to achieve a lower regret. we propose and study opportunistic reinforcement learning - a new variant of reinforcement learning problems where the regret of selecting a suboptimal action varies under an external environmental condition known as variation factor. When the variation factor is low, so is the regret of selecting a suboptimal action and vice versa. Our intuition is to emphasize exploitation when the variation factor is high, and explore more when the variation factor is low. We demonstrate the benefits of this novel framework for finite-horizon episodic MDPs by designing and evaluating OppUCRL2 and OppPSRL algorithms. Our algorithms dynamically balance the exploration-exploitation trade-off for reinforcement learning by introducing variation factor-dependent optimism to guide exploration. The OppUCRL2 can significantly outperform the UCRL2 [**22**] in the simulation and have the same theoretical guarantee with respect to the regret. The opportunistic RL concept is also easy to generalize for other reinforcement algorithms. To demonstrate it, we design the OppPSRL algorithm based on PSRL [**23**]. It also achieves better performance compared with the original version in the simulation. To the best of our knowledge, this is the first work proposing and studying the concept of opportunistic reinforcement learning. We believe this work will serve as a foundation for the opportunistic reinforcement learning concept and help further address the exploration-exploitation trade-off.

1.1.3. Causal Interpretability of Reinforcement Learning. The growing use of RL in sectors such as healthcare, transportation, and finance, which have significant societal and safety implications, has led to raised concerns around trust, bias, and explainability. The main concerns are about understanding how an RL agent reaches a decision and why a particular policy performs well. These challenges emerge primarily due to two factors: the intricacy and opaqueness of deep neural network-based RL algorithms, and the learning procedure of RL, which entails an agent's repeated interaction with its environment to ascertain a policy that ensures maximal long-term reward. The inherent temporality in RL, marked by crucial state relationships at disparate time instances, introduces an additional level of complexity surpassing that of supervised learning.

Existing explainable RL (XRL) strategies typically derive from methods such as the saliency map, a tool employed in supervised learning to elucidate image classifiers by highlighting relevant pixels for image classification. Analogously, these XRL methods strive to explain an RL agent's decisions by generating maps that spotlight "important" state features. However, there are at least two significant limitations with the current XRL methods. Primarily, they mostly espouse an associational viewpoint, gauging the significance of a feature by computing its correlation with an action. This approach, however, overlooks the fundamental difference between correlation and causation, which could potentially yield misleading explanations, thereby engendering user skepticism and potentially leading to the rejection of the RL system. Secondly, these techniques typically overlook temporal information, such as the interaction between states and actions over time - a vital aspect of RL. Our proposed solution integrates causality into the interpretability challenge of RL.

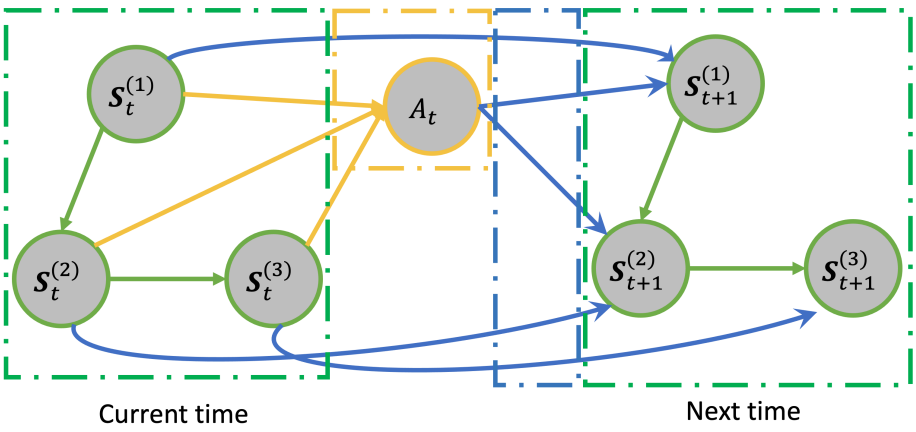


FIGURE 1.5. Example causal graph between the state and action. $\mathbf{S}_t^{(i)}$ is the i-th dimension of the interested state $\mathbf{S}$ at time t.

(1) Causal Explanation for Reinforcement Learning: State and Temporal Importance We formulated Markov Decision Process (MDP) as a Structural Causal Model (SCM), as depicted in Fig. 1.5. Our proposed XRL mechanism circumvents the limitations mentioned above in the following ways. First, our method is capable of generating fundamentally causal explanations. Second, our method is designed to quantify the temporal relationship between actions and states. In contrast, associational methods, including saliency maps, are unable to quantify how prior state

features might influence the current action because these models solely formulate the association between state and action at a single time-step, disregarding temporal correlations. Our XRL mechanism represents the inaugural effort to explain RL policies by causally elucidating their actions based on the state's causal and temporal importance. The proposed XRL mechanism can quantify the causal impact of states on actions, including their importance over time. Through a series of simulation studies - encompassing scenarios like crop irrigation, Blackjack, collision avoidance, and lunar lander - we illustrate how our mechanism outperforms existing state-of-the-art associational methods in explaining RL policies. Our XRL mechanism represents a pioneering effort in explaining RL policies by causally interpreting their actions based on both state and temporal importance.

(2) Causal Path-Specific Importance in Structural Causal Model To further leverage causality for improved explainability of learning models, we investigate individual-level path-specific effect analysis within a general SCM. Our goal is to develop a method that enables a more nuanced understanding of how a specific decision variable impacts an outcome variable. We introduce a novel definition of the causal counterfactual path-specific importance score. This proposed definition bears three key advantages. First, it can quantify individual-level impact through a specific pathway from the source vertex to the target vertex. Second, in cases of paths with multiple edges, the effect can be broken down into individual edges, making it more comprehensible for humans. Lastly, compared to the computationally intensive classical path-specific effect calculation, which necessitates independent evaluation of each path, the computation of our proposed definition is efficient. The classical approach becomes particularly burdensome in complex causal graphs, where the number of pathways increases exponentially with the number of edges. Evaluating all pathways from the source vertex to the target vertex in these contexts can prove costly. Our score possesses several desirable attributes, including adherence to the chain rule and consistency Capitalizing on these properties, we also present an effective algorithm that identifies the top-k most significant paths with the highest importance scores in a causal graph. In summary, our contributions encompass the introduction of a new metric for causal inference at the individual level, showcasing its mathematical robustness, and proposing an efficient algorithm for its computation and application in identifying impactful paths in a causal graph.

1.2. Related Work

Cellular Network Configuration: Various aspects of network parameter configuration have been studied in the literature, such as pilot power configuration, spectrum, handoff threshold, etc. Traditional approaches derive analytical relationship between network configuration and its performance based on communication theory, such as [**24**, **25**, **26**, **27**]. Such approaches are often prohibitively complex, involve various approximations, and require a significant amount of input information (such as the number of users, the location of each user, etc.). For instance, in [**24**], the authors have studied the problem of minimizing pilot power consumption while maintaining service coverage. The problem is formulated as integer programming and solved via a Dantzig-Wolfe decomposition method. Further, even with a known analytical model, most studies on pilot power configuration such as [**25**, **26**] focus on the design of heuristic policies. Recently, learning-based methods are proposed [**1**, **28**, **29**, **30**]. In [**28**], the authors propose a tailored form of reinforcement learning to adaptively select the optimal antenna configuration in a time-varying environment. In [**30**], the authors use Q-learning with compact state representation for traffic offloading. In [**29**], the authors design a generalized global bandit algorithm to control the transmit power in the cellular coverage optimization problem. In all these papers, BS similarities are not considered, and thus require more exploration. In [**31**], the authors formulate the anti-jamming power control problem of secondary users in cooperative cognitive radio networks as a Stackelberg game, and introduce RL to the source and relay node. In [**32**], the authors study into the power allocation problem of a BS in a MIMO-NOMA system. The problem is formulated as a zero-sum game against a smart jammer, and the Q-learning based power allocation scheme is used to adapt to the dynamic NOMA transmission game. Most of the existing work in RL-based network configuration treats all relevant factors of the whole network as state or action in the RL algorithms. However, in a large-scale network, general RL algorithms can suffer from large state spaces and action sets, which leads to extremely bad prediction accuracy and convergence rate. To address this scalability problem in RL-based network configuration, multiagent solutions are proposed. In [**30**], the authors employ Q-learning with compact state representation for a Hyper-Cellular Network (HCN) to decide traffic offloading strategy to improve network energy-efficiency, and further propose a distributed learning solution where the macro BSs are designed to make local decisions but learn in a cooperative way.

In [**33**], the authors propose a distributed RL algorithm to configure the power allocation of all BSs with the objective of improving network performance. In essence, this line of work uses multiagent reinforcement learning [**34**, **35**].

Multi-Armed Bandits: Multi-armed bandits (MABs) [**36**] is a sequential decision problem, which is a special case of reinforcement learning, where the reward of a decision is immediately observed. Well-known algorithms include UCB [**37**] and Thompson Sampling [**38**]. Further, with side information provided before decision making, the classic MAB is generalized to contextual bandit problems [**17**, **39**]. In wireless networks, bandit algorithms have been applied in several application scenarios. In downlink scheduling problems, the bandit algorithms, especially Whittle Index policy, have been employed to deal with the curse of dimension of Markov Decision Process [**40**, **41**, **42**]. In vehicular networks, bandits have been applied to design learning-based task offloading [**43**, **44**]. Further, emerging literature demonstrates the potential of contextual bandit in network configuration problems [**1**].

Contextual Bandits: Contextual bandit [**19**] is an extension of classic multi-armed bandit (MAB) problem [**45**]. In contrast to the classic K-arm bandit problem [**46**, **47**], side information called context is provided in contextual bandit problem before arm selection [**16**, **17**, **18**, **19**]. The contextual bandits with linear payoffs was first introduced in [**16**]. In [**48**], LinUCB algorithm is introduced based on the "optimism in the face of Uncertainty" principal for linear bandits. The LinUCB algorithm and its variances are reported to be effective in real application scenarios [**48**, **49**, **50**, **51**, **52**]. Compared to the classic K-armed bandits, the contextual bandits achieves superior performance in various application scenarios such as news article recommendation [**53**], clinical trials [**54**]. Although LinUCB is effective and widely applied, its analysis is challenging. In the initial analysis effort [**17**], instead of analyzing LinUCB, it presents an $O(\sqrt{T \ln^3(T)})$ regret bound for a modified version of LinUCB. The modification is needed to satisfy the independent requirement by applying Azuma/Hoeffding inequality. In another line of analysis effort, the authors in [**18**] design another algorithm for contextual bandits with linear payoffs and provide its regret analysis without independent requirement. Although the algorithm proposed in [**18**] is different from LinUCB and suffers from a higher computational complexity, the analysis techniques are helpful.

Transfer Learning and Multi-Task Learning: Transfer learning and multi-task learning are powerful methods that improve the learning efficiency by using the data samples from multiple sources [**55**, **56**, **57**]. They have been successfully applied to many fields, such as text sentiment classification [**58**] and image classification [**59**]. In [**60**], the authors re-weight the instances in multi-source to address both marginal and conditional distribution differences between the source and target domains. In [**61**], the authors consider transfer learning via dimensionality reduction. They learn a low-dimensional latent feature space where the distributions between the source domain data and the target domain data are the same or close to each other. Similarly, [**62**] proposes a feature transformation approach for domain adaptation called Transfer Component Analysis (TCA) to discover common latent features that have the same marginal distribution across the source and target domains while maintaining the intrinsic structure of the original domain data. A common way is using a kernel function to define the similarity among tasks, e.g., in [**63**,**64**,**65**]. In [**65**], the authors design an algorithm that can transfer information among arms in the contextual bandit.

Opportunistic Learning: Optimism in the face of uncertainty (OFU) is a popular paradigm for the exploration-exploitation trade-off in RL. Here, each pair of states and actions is assigned an optimism bonus. The agent then chooses a policy that is optimal under this "optimistic" model of the environment. To learn efficiently, the agent maintains control over its uncertainty by assigning a larger optimism bonus to potentially informative states and actions. This bonus can stimulate and guide the exploration process. Most OFU algorithms provide strong theoretical guarantees [**22**,**66**,**67**,**68**,**69**]. An alternate approach is inspired by Thompson sampling (TS) [**70**]. In RL, TS approaches [**71**] maintain a posterior distribution over the reward function and the transition kernel, then compute the optimal policy for a randomly sampled MDP from the posterior. One of the well-known TS algorithms in literature is the Posterior Sampling for Reinforcement Learning (PSRL) [**23**, **72**].

The opportunistic learning has been introduced in [**73**] for classic K-armed bandits. However, we note that opportunistic learning exists for any sequential decision making problem. In [**74**], the authors study into contextual bandits with HLCS (High-Level Critical Situations) set, and proposes a contextual-ϵ-greedy policy, a policy that has an opportunistic nature since the ϵ (exploration level) is adaptively adjusted based on the similarity to HLCSs (importance level).However,

it only introduces a heuristic algorithm, and does not present a clearly formulation of opportunistic learning. Furthermore, the policy design in [**74**] implicitly makes the assumption that the contexts in HLCS have already been explored sufficiently beforehand, which is not a cold-start problem.

Explainable RL (XRL): Based on how an XRL algorithm generates its explanation, we can categorize existing XRL methods into state-based, reward-based, and global surrogate explanations [**75**, **76**, **77**]. State-based methods explain an action by highlighting state features that are important in terms of generating the action [**78**, **79**]. Reward-based methods generally apply reward decomposition and identify the sub-rewards that contribute the most to decision making [**80**]. Global surrogate methods generally approximate the original RL policy with a simpler and transparent (also called intrinsically explainable) surrogate model, such as decision trees, and then generate explanations with the surrogate model [**81**]. In the context of state-based methods, there are generally two ways to quantify feature importance: (i) gradient-based methods, such as simple gradient [**82**] and integrated gradients [**83**], and (ii) sensitivity-based methods, such as LIME [**84**] and SHAP [**85**].

Causal Explanation: Causality has already been utilized in XAI, mainly in supervised learning settings. Most existing studies quantify feature importance by either using Granger causality [**86**] and average or individual causal effect metric [**87**] or by applying random valued interventions [**88**]. Two recent studies [**89**] and [**90**] are both focused on causal explanations in an RL setting. Compared with [**89**], the main difference is that we provide a different type of explanation. Our method involves finding an importance vector that quantifies the impact of each state feature, while [**89**] provides a causal chain starting from the action. We also demonstrate the ability of our approach to provide temporal importance explanations that can capture the impact of a state feature or action on the future state or action.

Path-Specific Analysis: Path-specific analysis has seen significant attention in academic research. Most existing work focuses on non-parametric settings, identifiability, or informational decomposition of SCMs [**91**, **92**, **93**]. These approaches are anchored in a well-defined SCM model. Related research proposed causal effect definitions such as incremental causal effects and marginal treatment effects [**94**, **95**, **96**]. These definitions employ a comparable constraint format for the

perturbation value δ. Their research predominantly concentrates on total effects and population-level analysis There exists a significant body of literature focused on population-level causal effects. The studies by Janzing et al. [**97**, **98**] and Wang et al. [**99**] define and measure causal strength (effect) using Shapley values. Janzing et al. assess causal influence based on the distributional change that results from the removal of a "causal arrow". Further, their concept of intrinsic causal contribution [**97**] allows separating intrinsic information added by each vertex from information drawn from its ancestors. Meanwhile, Wang et al. [**99**] attribute credit to edges, providing an interpretation of the entire causal graph.

CHAPTER 2

Multi-Agent Learning-Based Joint Optimization: A Cellular Network Configuration Application

2.1. Introduction

In this work, we study cellular network configuration. As illustrated in Fig. 2.1, a cellular network consists of a number of base stations (BSs), each covering a certain geographic area, called a cell. In each cell, the BS has a large number of parameters to configure, including those of spectrum band, power configuration, antenna setting, user handover, etc. The configuration of these BS parameters has a significant impact on the overall performance, and thus network configuration is critical. Current practice is mostly based on field experience and manual adjustment. The process is labor-intensive, error-prone, and far from optimal. In this work, we propose a learning-based joint-optimization approach to automate and optimize cellular network configuration.

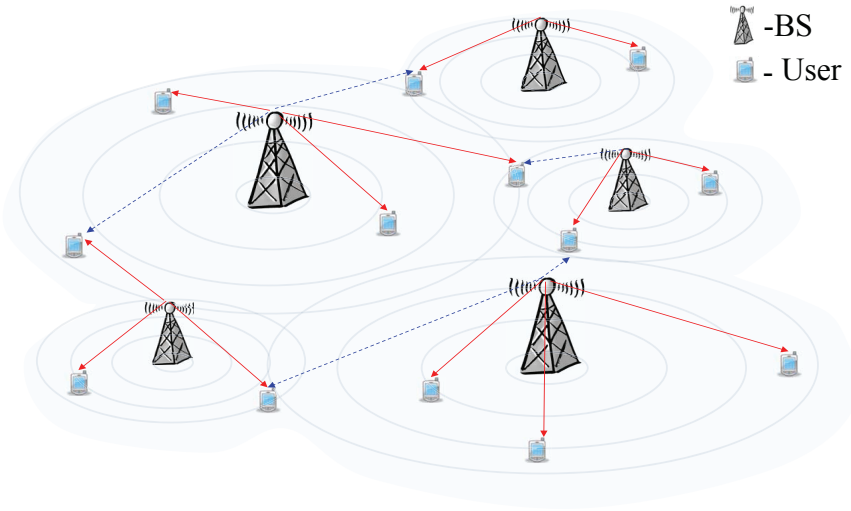


FIGURE 2.1. An illustration of cellular network configuration. A red solid line means that a mobile user is associated with the BS (i.e., the BS serves the user), and a blue dash line means that the mobile user can receive the BS's signal, but is not served by it.

The goal of network configuration is to optimize network utility that measures service quality, such as network throughput and user delay. Network utility is typically defined as the sum of the utility of all cells. Because of the geographic contiguity of cells, the utility of one cell is affected not only by the configuration of its own BS, but also by those of its neighboring BSs. Thus, to optimize its network utility, the network should jointly configure all BSs.

The extensive literature on traditional network utility maximization (NUM) has focused on abstract closed-form utility functions, which often exhibit nice mathematical properties. However, in network configuration, the impact of a control parameter on network utility can be highly complex and thus the utility function is difficult to derive from analysis. Therefore, a natural step is to learn an appropriate utility function based on available network data, and then to optimize over the learned function. This approach faces a few specific challenges:

- Limited data availability: Because current network operation does not often change network configuration parameters unless performance problems occur, each cell contains only a limited amount of data for learning its utility function.
- Convoluted sample data: The sample data collected in networks is reported in the format of network status, BS configuration, and the corresponding utility value. The caveat is that the network status depends not only on hidden environmental states that we do not directly observe, but also on the current configuration. Thus, we need to carefully extract appropriate features from the network status to learn the utility model.
- Joint optimization among cells: The utility of a cell depends not only on the configuration of its BS, but also on those of its neighboring BSs. Therefore, to maximize the network utility, we should consider all cell configurations jointly. This joint configuration is difficult because of the complexity of the learned utility function and the high dimensionality of the control variables.
- Time-varying network dynamics: Because of inherent network dynamics (e.g., traffic variation and user mobility), the utility function is time-varying, as discussed in Sec. 2.3.3. Therefore, it is essential to design an online algorithm that adapts promptly to network dynamics.

To address these challenges, we propose an approach based on online learning and joint optimization for cellular network configuration. Our contributions are multi-fold:

- To learn the appropriate utility function, we develop a neural-network-based model that addresses the convoluted sample data issue and achieves good accuracy. The learned utility function is used to formulate a global network configuration optimization problem.
- To solve this high-dimensional non-concave maximization problem, we design a Gibbs-sampling-based algorithm that converges to the optimal solution when a temperature parameter is small enough. However, when the temperature parameter is small, the convergence time of the algorithm increases dramatically. To address this challenge, in particular, in the context of network dynamics, we further design an online algorithm that converges to a local optimal promptly.
- To illustrate the idea, we use the case study of pilot power configuration. Numerical results show the effectiveness of the proposed approach. The proposed approach can be applied to similar cellular network configuration problems.

The rest of the paper is organized as follows. The related work is in Sec. 2.2. We present the system model and problem formulation in Sec. 2.3. The learning-based utility prediction is conducted in Sec. 2.4. Based on the model learned, an online-learning-based joint-optimization method for cellular network configuration is proposed in Sec. 2.5. We demonstrate the numerical results in Sec. 2.6, and conclude in Sec. 3.7.

2.2. Related Work

Most existing studies on pilot power configuration assume known models based on communication theories [**24**, **25**, **26**, **27**, **100**]. For instance, in [**24**], the authors have studied the problem of minimizing pilot power consumption while maintaining service coverage. The problem is formulated as integer programming and solved via a Dantzig-Wolfe decomposition method. The main limitation of this approach is that the modeling requires a significant amount of real-time information, such as user locations and wireless channel conditions, at any time. Further, even with a known analytical model, most studies on pilot power configuration such as [**25**, **26**] focus on the design of heuristic policies.

Data-driven wireless communication has been drawing increasing attention recently [**101**, **102**, **103**]. In [**101**], the authors provide an overview of mobile big data, and highlight the great potential of mobile big data mining for diverse applications including network resource planning. In [**102**], the authors introduce the features, sources, and applications of mobile big data in a comprehensive manner and discuss several distinctive characteristics of mobile big data such as spatio-temporal features. The authors in [**103**] consider mobile big data as an unprecedented opportunity to re-design wireless networking for potential performance gains. They also discuss the difficulties in big-data-aware network design such as scalability and complexity.

We note that our problem falls into the general scope of reinforcement learning (RL). We discuss the existing work on RL-based network configuration next. In [**28**], the authors propose a tailored form of RL to configure the antenna parameters in a single femto-cell so as to minimize the transmission power. In [**31**], the authors formulate the anti-jamming power control problem of secondary users in cooperative cognitive radio networks as a Stackelberg game, and introduce RL to the source and relay node. In [**32**], the authors study into the power allocation problem of a BS in a MIMO-NOMA system. The problem is formulated as a zero-sum game against a smart jammer, and the Q-learning based power allocation scheme is used to adapt to the dynamic NOMA transmission game. Most of the existing work in RL-based network configuration treats all relevant factors of the whole network as state or action in the RL algorithms. However, in a large-scale network, general RL algorithms can suffer from large state spaces and action sets, which leads to extremely bad prediction accuracy and convergence rate. We illustrate this issue numerically in Sec. 2.6.1.

To address this scalability problem in RL-based network configuration, multiagent solutions are proposed. In [**30**], the authors employ Q-learning with compact state representation for a Hyper-Cellular Network (HCN) to decide traffic offloading strategy to improve network energy-efficiency, and further propose a distributed learning solution where the macro BSs are designed to make local decisions but learn in a cooperative way. In [**33**], the authors propose a distributed RL algorithm to configure the power allocation of all BSs with the objective of improving network performance. In essence, this line of work uses multiagent reinforcement learning [**34**, **35**]. Along

this line, we have also designed a distributed Q-learning algorithm, where each BS runs a local Q-learning algorithm with a cooperative reward capturing the dependencies among neighboring BSs. In it, we have carefully designed local state and action to be used, as well as function approximation model, activation function, and learning rate. However, distributed RL algorithms are non-trivial in design and usually rely on heuristic. In addition, when the state space and action set are large, as in our problem, they can still suffer from slow convergence. As a result, we treat the distributed Q-learning algorithm as a comparison baseline with the details provided in Appendix 2.8.2. As shown in Sec. 2.6.2, our online-learning-based joint-optimization approach outperforms the distributed Q-learning algorithm in both the overall performance and the ramp-up time.

2.3. System Model and Problem Formulation

2.3.1. Pilot Power Configuration. In cellular networks, pilot power is one of the most important parameters to be configured. Specifically, pilot power signal is used by mobile devices to estimate channel quality, to select cells to associate with, and to select neighboring cells to handover to. Therefore, the strength of the pilot power signals determines the coverage area of the cell and impacts the network throughput and quality of service. In this work, the goal of the pilot power configuration is to maximize network throughput, which is the sum of the throughput of all cells. Specifically, pilot power affects network throughput in two ways.

First, in general, when the pilot power of a BS increases, the received signal strength of mobile devices in its vicinity increases, and thus these mobile devices are more likely to associate with this BS or handover to this BS. Therefore, the number of users associated with this BS increases, and the number of users associated with its neighboring BSs decreases. In other words, pilot power affects the total throughput by affecting the number of users in its cell and that in its neighboring cells. In general, if a cell has too few users, its service capacity cannot be fully utilized, which results in a relatively low throughput. On the other hand, if a cell has too many users, its service capacity cannot satisfy the demand of all users; meanwhile, since its neighboring cells may have under-utilized capacity, the overall system performance is not optimized.

Second, the total power, consisting of pilot power and transmission power, is fixed at a BS. When the pilot power increases, the transmission power for actual data transmission decreases.

The capacity of a cell is determined by the transmission power. Therefore, when the pilot power increases, the throughput of a cell first increases when the demand is smaller than the capacity, and then decreases when the demand is higher than the available capacity supported by the transmission power.

We note that pilot power configuration also affects network coverage; e.g., if pilot power is small, certain areas may not be covered by any BSs. Coverage issue is typically considered in the network planning stage, and addressed by deciding appropriate BS locations and a proper range of pilot power. Here the pilot power is chosen between 30dBm to 36dBm. In discussion of network configuration, we assume that the coverage requirement is satisfied and thus we do not further consider it in this paper.

2.3.2. Problem Formulation. We formulate the problem next. We use bold font to represent vectors. Consider a cellular network of N BSs. Each BS provides service for a corresponding cell. The time is discretized. For each time slot t, the network state vector is denoted by $\mathbf{S}^{(t)} = (S_1^{(t)}, S_2^{(t)}, \cdots, S_N^{(t)})$, where $S_n^{(t)}$ is the state for BS n in slot t. Examples of network state include user density, and uplink/downlink traffic load level, etc. The network control is denoted by $\mathbf{A}^{(t)} = (A_1^{(t)}, A_2^{(t)}, \cdots, A_N^{(t)})$ where $A_n^{(t)}$ represents the control variables for BS n in slot t, and its value is chosen from the BS action set $\mathcal{A}$. Pilot power is the control variable we consider in this work, which takes discrete values from 30dBm to 36dBm with a granularity of 0.5dBm.

Let $R_n^{(t)}$ denote the utility of cell n in slot t and define $r_n(\cdot)$ as the cell utility function of cell n that satisfies,

$$R_n^{(t)} = r_n(\mathbf{S}^{(t)}, \mathbf{A}^{(t)}). \tag{2.1}$$

That is, the utility of each cell (i.e., throughput of each cell) is determined by the network states and control variables. In the pilot power configuration problem, the utility is the throughput, and the object is to maximize the network throughput by choosing appropriate control variables for all BSs at each time slot. That is,

$$\max_{\mathbf{A}^{(t)} \in \mathcal{A}^N} \sum_{t=1}^{T} \sum_{n=1}^{N} r_n(\mathbf{S}^{(t)}, \mathbf{A}^{(t)}). \tag{2.2}$$

Note that a more rigorous explanation of Eq. (2.2) and the cell utility function in (2.1) is as follows. In general, in networks, there are "hidden" factors that are not recorded as network state, either because they are not readily available or because there are too many of them. For example, networks do not track detailed mobility of each user nor the application types of each flow. Because of these hidden factors, given $\mathbf{S}^{(t)}$ and $\mathbf{A}^{(t)}$, $R_n^{(t)}$ can be random. Therefore, the definition of cell utility function in (2.1) is indeed $r_n(\mathbf{S}^{(t)}, \mathbf{A}^{(t)}) = \mathrm{E}[R_n^{(t)} |\ \mathbf{S}^{(t)}, \mathbf{A}^{(t)}]$, where the expectation is taken over the "hidden" factors. Similarly, the optimization problem defined in Eq. (2.2) is the summation of expected cell utilities accordingly. For presentation simplicity, in the following, we use the earlier equation in the rest of the paper.

In addition, note that in Eq. (2.1), the utility value depends on all states and control variables. The complexity of this function is thus high. To make the problem more tractable, we utilize the inherent property of wireless transmission that signal strength attenuates significantly over distance. Therefore, the performance of a BS is mainly affected by the network states and control variables of itself and its neighbors. Thus, we make the following approximation:

$$r_n(\mathbf{S}^{(t)}, \mathbf{A}^{(t)}) \approx r_n(\mathbf{S}^{(t)}_{\mathcal{N}(n)}, \mathbf{A}^{(t)}_{\mathcal{N}(n)}), \tag{2.3}$$

where $\mathcal{N}(n) = \{n\} \cup \{i|$ BS i is BS n's neighbor$\}$ includes BS n and its neighbors; $\mathbf{S}^{(t)}_{\mathcal{N}(n)}$ is a vector which includes all $S_i^{(t)}$ such that $i \in \mathcal{N}(n)$; $\mathbf{A}^{(t)}_{\mathcal{N}(n)}$ is a vector which includes all $A_i^{(t)}$ such that $i \in \mathcal{N}(n)$.

For example, in the pilot power configuration problem, we choose the neighbors of BS n as the cells that have the most impact on cell n's utility. Note that the neighborhood relationship may not be symmetric. That is, $i \in \mathcal{N}(n)$ does not necessarily imply $n \in \mathcal{N}(i)$.

To further simplify the problem, we make the assumption that network state vector $\mathbf{S}^{(t)}$ does not depend on historical actions $\mathbf{A}^{(t-1)}, \cdots, \mathbf{A}^{(1)}$. To satisfy the assumption, we need to choose network states appropriately, that is, network states should only depend on latent environment features such as user density, user location, traffic type, and mobility pattern, which do not depend on actions. Because of this assumption, the problem defined in (2.2) can be simplified: at each time slot t, we can choose the action that maximizes the current network utility $\sum_n r_n(\mathbf{S}^{(t)}, \mathbf{A}^{(t)})$.

As a result, substituting (2.3) into (2.2), we have the following optimization problem at each slot t,

$$\max_{\mathbf{A}^{(t)} \in \mathcal{A}^N} \sum_{n=1}^{N} r_n(\mathbf{S}^{(t)}_{\mathcal{N}(n)}, \mathbf{A}^{(t)}_{\mathcal{N}(n)}), \text{ given network state } \mathbf{S}^{(t)}. \tag{2.4}$$

Note that, in contrast, our implemented baseline algorithm, a distributed Q-learning algorithm, still uses the aggregated throughput as the reward as in Eq. (2.2).

The network operates as follows: At the beginning of each slot t, network state vector $\mathbf{S}^{(t)}$ is observed, then based on $\mathbf{S}^{(t)}$, the network control $\mathbf{A}^{(t)}$ at time t is decided by a configuration policy to maximize the sum of the utilities of all cells. The process goes on for each time slot in a centralized manner. Note that the configuration policy can be updated based on historical information: $S_n^{(\tau)}$, $A_n^{(\tau)}$, $R_n^{(\tau)}$, for $\forall n$, $\forall \tau < t$.

2.3.3. Network Dynamics. We note that networks exhibit time-varying characteristics, which implies that we need online algorithms to capture such dynamics. Network utility is affected by many network factors, such as traffic type and volume, arrival and departure pattern, and user mobility. As discussed in Sec. 2.3.2, some factors are "hidden" because a network typically does not and cannot collect all information. Therefore, when such "hidden" factors change in the network, the expression of cell utility functions, i.e., $r_n(\cdot)$, can change over time. To capture such dynamics, we need an online learning algorithm to train and update the model for the utility function, as well as a fast converging algorithm to optimize the configuration.

2.4. Learning Utility Function

In this section, we choose an appropriate regression model to predict the utility of each cell based on the network state vector and network control in the neighborhood of the cell. That is to say, we choose an appropriate regression model to learn the cell utility function $r_n(\mathbf{S}^{(t)}_{\mathcal{N}(n)}, \mathbf{A}^{(t)}_{\mathcal{N}(n)})$ by a data-driven approach, so as to provide a good foundation to solve the problem in (2.4). Note that based on the regression model chosen here, we can build a framework to learn and control in an online manner, discussed in Sec. 2.5. To learn a good utility function for the network configuration problem, we need to address two specific challenges: limited data availability and convoluted sample data, discussed as follows.

2.4.1. Data Aggregation. One obstacle in estimating the utility function is the limited data availability in cellular networks. This may sound surprising at first glance given that networks collect a large amount of data during operation. The caveat is that most of the data is collected under a fixed network configuration parameter. Current practice is highly conservative - one typically does not change network configuration unless performance issues occur, such as insufficient coverage or a high call-drop rate. Current network operators are also reluctant to conduct experiments in the field because a poor configuration can negatively affect performance. As a result, we may have a large number of samples for a given configuration, but few for others, in a given cell.

To fully utilize the limited data, we notice that a cellular network typically consists of cells with similar structures. Thus, we assume that the cell utility functions $r_n(\cdot)$ of different cells have the same expression, denoted by $r(\cdot)$. Thus, the problem defined in (2.4) can be approximated by the following, at each time slot t,

$$\max_{\mathbf{A}^{(t)} \in \mathcal{A}^N} \sum_{n=1}^{N} r(\mathbf{S}^{(t)}_{\mathcal{N}(n)}, \mathbf{A}^{(t)}_{\mathcal{N}(n)}), \text{ given network state } \mathbf{S}^{(t)}. \tag{2.5}$$

With this unified cell utility function, we can aggregate data from all cells to learn the utility function. This approach alleviates the data scarcity issue, especially when the algorithm needs to adapt to network dynamics.

REMARK 2.4.1. *To consider the impact of the difference between cells and the impacts of neighboring cells as well as dynamic environmental factors, the input features of the unified cell utility function should be carefully designed, so as to include these impacts to some degree. There is clearly a balance between data availability and the impact of the difference between cells. The feature selection procedure is discussed in Sec. 2.4.2.*

REMARK 2.4.2. *Note that in scenarios when data is abundant, each cell can train a separate utility function $r_n(\cdot)$ based on its own data. Then, the joint network configuration algorithms in Sec. 2.5 including Module 1 and Module 2 still apply using these separate cell utility functions.*

2.4.2. Feature Selection with Convoluted Sample Data. The sample data is collected from an industrial-grade cellular network simulator that is described in Sec. 2.6, based on a random policy for pilot power configuration in this section. Sample data is illustrated in Table 2.1, where

"Cell" is the cell ID; "Pilot" is the current pilot power of the BS in this cell; "Utility" is the throughput of the corresponding cell with the unit of Kbit; "Users" is the number of users in the cell, and "Cluster" includes the cell IDs of the 5 neighboring cells that have the most impact on this cell's throughput.

TABLE 2.1. Features of Sample Data.

Cell	Pilot	Load	Utility	Users	Cluster
101	30	0.5934	616.3	19	233;184;202;185;171;
102	30	0.1458	63.2	2	101;177;180;184;172;
107	33	0.8991	357.8	19	211;115;114;219;113;

An important problem in feature selection is that sample data is convoluted as shown in Table 2.1. Specifically, the values of the load and the number of users in the sample data depend not only on latent environmental states that we do not directly observe (such as user density), but also on the current control variables, i.e. the pilot powers.

Note that our final goal is not the prediction itself, but to optimize network configuration based on the learned cell utility function. Consequently, because of this convolution, we cannot naively use the items in the sample data as features to train the utility function. Otherwise, the resulting utility function is not useful in the control problem defined in Eq. (2.4) because $\mathbf{S}^{(t)}_{\mathcal{N}(n)}$is not known until $\mathbf{A}^{(t)}_{\mathcal{N}(n)}$is decided.

As a result, we need to carefully select state features so that they 1) extract latent environment information as much as possible, and 2) are as independent as possible from the control variables.

We note that another consideration is that we have limited data availability, and thus the number of features cannot be too large.

Considering these two factors, we have tested the prediction accuracy of several sets of features, as summarized in Table 2.2, based on a neural network regression model discussed in Sec. 2.4.3. We measure prediction accuracy by the coefficient of determination R^2 as,

$$R^2 = 1 - \frac{\eta_{\text{ss}}}{\eta_{\text{var}} M_{\text{sp}}}, \tag{2.6}$$

where η_{ss} is the sum of squared prediction errors, η_{var} is the variance of the target, and M_{sp} is the total number of samples. The larger the value of R^2, the better the model can capture the observed outcomes. Note that $R^2 \leq 1$. Here, we compare the prediction accuracy on both training data

TABLE 2.2. Coefficient of Determination for Different Selections of Features.

Features	Training Data	Cross Validation
Pilot Powers, Load, Number of Users	0.729630	0.728747
Pilot Powers, Time-Average Load, Time-Average Number of Users	0.758423	0.717136
Pilot Powers, Cluster-Average Load, Cluster-Average Number of Users	0.127370	0.113907

and cross validation set, and use the latter one as the criterion to choose an appropriate model, since the training data prediction accuracy can be overly optimistic, as discussed in [**104**].

The first feature set includes the pilot powers of the target BS and its neighbors, the current load of the target BS, and the current number of users in the target BS, as shown in the first row of Table 2.2. This directly uses the items from the sample data. The prediction accuracy of this feature set is about 0.73. However, as stated above, the load and the number of users depend on the current pilot powers, and thus this model is not useful in pilot power configuration. It only serves as a benchmark for prediction accuracy.

To address this convolution issue, we consider two other feature sets, as shown in the second and third rows of Table 2.2. The second feature set includes time-average load and time-average number of users in the target cell, and the third one includes cluster-average load and cluster-average number of users, instead of the load and number of users as in the first one. Here, the cluster average means that we average the load or the number of users among the target cell and its neighboring cells. These two new feature sets alleviate the dependency of state features on the control variables by averaging over time slots or over mutually interfering cells. As shown in Table 2.2, the prediction accuracy of the second feature set is similar to that of the first feature set while the prediction accuracy of the third one is low. Therefore, we choose the second feature set, i.e., we choose time-average load and time-average number of users as the state features.

Note that in the beginning of each slot t, we only know the historical data until slot $t-1$. Also, to keep the data updated, we only use the most recent data through an exponential moving average with a parameter of $1/l_{\text{cache}}$, where l_{cache} is chosen to ensure stability and to allow prompt update. We choose $l_{\text{cache}} = 300$ in our numerical evaluations in Sec. 2.6.

2.4.3. Model Selection. We compare several learning methods summarized in Table 2.3, including the 2^{nd} degree polynomial regression, and epsilon-support vector regression (SVR). It can

be seen that the neural network model outperforms the other methods. Note that SVR achieves the second best performance, however SVR is much slower in online training and prediction, which is a severe limitation in our case.

TABLE 2.3. Coefficient of Determination for Regression Methods.

Regression Method	Training Data	Cross Validation
Linear Regression	0.542157	0.529733
Lasso Regression	0.541470	0.530069
Polynomial Regression	0.625649	0.610161
Epsilon-SVR	0.671278	0.657081
Neural Network Regression	0.730046	0.693995

For the neural network model, it is also important to select appropriate hyper-parameters. We choose only one hidden layer because there are only 8 input features (i.e., 6 pilot powers from the target cell and its 5 neighbors, the time-average load, and time-average number of users, as explained in Sec. 2.4.2.) Also, we compare the performance of different number of neurons in the hidden layer, as shown in Table 2.4. We choose 25 for its best performance.

TABLE 2.4. Coefficient of Determination for Hyper-parameters.

Number of Neurons	Training Data	Cross Validation
10	0.690083	0.684574
15	0.689096	0.687062
20	0.691792	0.684915
25	0.702833	0.690766
30	0.734662	0.672190
35	0.797004	0.611949

2.4.4. Summary of Cell Utility Function Regression Model. Here, in summary, a neural-network-based regression model is chosen to predict the throughput of a cell based on the states and control variables of itself and its neighbors. To address the limited data availability problem as discussed in Sec. 2.4.1, a unified cell utility function is chosen for all cells utilizing structure similarity of cells. The learned model predicts the throughput of the cell. The 8 input features of the learning model are 1) the pilot power of the BS in the target cell, 2) the pilot powers of the BSs in 5 neighboring cells that have the most impact on the target cell, 3) time-average load, and 4) time-average number of users in the target cell. The input features are carefully selected to

address the convoluted sample data problem, as discussed in Sec. 2.4.2. The time-average features are estimated using exponential moving average method. In addition, based on the experiment results, a neural network model with one hidden layer and 25 neurons are selected for its best performance among different regression models.

2.5. Online-Learning-Based Joint Optimization

After estimating the cell utility function $r(\cdot)$, we proceed to solve the optimization problem formulated in Eq. (2.4). This step is challenging because of the huge state and action spaces for network configuration. Let the number of possible values for the control variable of BS n be $|\mathcal{A}|$. Then, the cardinality of network action space is $|\mathcal{A}|^N$, which increases exponentially with the number of BSs. When a cellular network has hundreds of BSs to configure, the action space is immense. Even when we limit the impact of a BS to its neighborhood, the optimization space is still exponential. Specifically, the action of a BS affects the performance of itself and its neighbors, which affects the action of the neighbors, and so on. This ripple effect fuses the optimization of the entire network together. In this section, we design an online algorithm based on Gibbs sampling to solve this joint optimization problem.

2.5.1. Gibbs Sampling Primer. We first introduce the intuition to apply Gibbs sampling and then provide Gibbs sampling primer.

By considering the values of the pilot powers of the whole network as a random field and regarding the utility of the whole network as the global energy function of a possible configuration, we want this random field (i.e., the pilot powers of the whole network) to follow a specific joint distribution such that the random field value (i.e., a specific pilot power configuration of the whole network) of the maximal global energy function (i.e., utility of the whole network) has a dominant probability. To achieve this goal, a joint distribution of the random field based on the value of network utility can be designed. Meanwhile, Markov Chain Monte Carlo (MCMC) can be applied to sample the resulting high dimensional multi-variate probability distribution. Note that by MCMC, a Markov chain of random field is constructed (i.e., the state of the Markov chain is the value of the random field), which has a steady state distribution equivalent to the targeting joint distribution of the random field.

Among MCMC algorithms, Gibbs sampling is widely used. In Gibbs sampling, the constructed Markov chain transits based on conditional distributions. In addition, if the targeting joint distribution of Gibbs sampling satisfies a certain property, the transition probabilities of the designed Markov chain can be purely based on local specification, which potentially leads to update by simple local interactions in some special cases. Specifically, if the targeting joint distribution is based a global energy function that is derivable from a potential, i.e., a sum over cliques on the random field with a neighborhood structure, the Markov chain in Gibbs sampling can be updated by local specification. This requirement is restrictive and if the original neighborhood structure of the cellular network is used, the network utility (which is the global energy function here) cannot satisfy it. However, with a judiciously designed two-hop neighborhood, the cellular network configuration can satisfy this constraint and further the network utility function can be maximized by simple local interacts.

To make the paper self-contained, here we also briefly introduce Gibbs sampling. Consider a random field $\hat{\mathbf{A}}$, which can be viewed as a vector of dimension N defined on an undirected graph. That is, the n-th element of $\hat{\mathbf{A}}$ corresponds to the n-th node on the undirected graph. Then, $\hat{\mathbf{A}}$ is said to have a Gibbs distribution if its probability distribution can be written as:

$$\Pr\{\hat{\mathbf{A}} = \mathbf{a}\} \propto \exp\left[\frac{1}{\nu} F(\mathbf{a})\right], \forall \mathbf{a}, \tag{2.7}$$

where $F(\mathbf{a})$ satisfies the condition that there exists a function $V_C(\mathbf{a}_C)$ for each clique C of the undirected graph such that,

$$F(\mathbf{a}) = \sum_C V_C(\mathbf{a}_C), \tag{2.8}$$

where $\mathbf{a}_C$ is a vector including all a_i (which is the i-th element of vector $\mathbf{a}$) with $i \in C$ (i.e., the i-th node of the undirected graph is in clique C).

To relate to our problem, $\hat{\mathbf{A}}$ represents the pilot power configuration of the whole network. In Eq. (2.7), $F(\mathbf{a})$ can be viewed as a network utility function of configuration $\mathbf{a}$ for the random field $\hat{\mathbf{A}}$. The $V_C(\mathbf{a}_C)$ is the utility function of a clique C in the undirected graph with configuration $\mathbf{a}$. That is, by Eq. (2.8), the utility function $F(\mathbf{a})$ contains functions on the cliques of the undirected graph.

The key insight is that when ν, $\nu > 0$, is small enough, the configuration $\mathbf{a}$ with the highest utility function can have a dominant probability. Furthermore, we can obtain this probability by Gibbs sampling, discussed next.

By Gibbs-Markov equivalence, it can be shown that when a random field has a Gibbs distribution, it is a Markov random field with a well-designed local specification [**105**], and the distribution of the random field is equivalent to the steady state distribution of a Markov chain on the Markov random field. Further, by reversible Markov chain theory, algorithms such as Gibbs sampler can design transition probabilities for the Markov chain, which relates to the local specializations. When this Markov chain approaches its steady state distribution, the Gibbs distribution is obtained. The readers can refer to the Chapter 7 of [**105**] for details.

However, this theory is limited to the case when function $F(\cdot)$ is a sum of functions on cliques of the undirected graph, as in (2.8). A recent study [**106**] considering locally coupled system extends the results to the case when $F(\cdot)$ consists of functions on neighborhoods of each node. We adopt this approach in our algorithmic design.

2.5.2. Gibbs-Sampling-Based Network Configuration. In this subsection, we focus on a single time slot t, and solve the problem defined in (2.4) for a given network state $\mathcal{S}^{(t)}$. We omit the superscript (t) in this section when no confusion is incurred.

We next define $F(\cdot)$ and $\hat{A}$ in our problem context. We define $F(\cdot)$ as the network utility function:

$$F(\mathbf{a}) = \sum_{n=1}^{N} r_n(\mathbf{S}_{\mathcal{N}(n)}, \mathbf{a}_{\mathcal{N}(n)}), \forall \mathbf{a} \in \mathcal{A}^N. \tag{2.9}$$

Define $\hat{\mathbf{A}}$ as a random vector on the action space satisfying the probability distribution $\Pr[\hat{\mathbf{A}} = \mathbf{a}] = \pi_\nu(\mathbf{a})$ with,

$$\pi_\nu(\mathbf{a}) = \frac{\exp\left[\frac{F(\mathbf{a})}{\nu}\right]}{\sum_{\mathbf{z}\in\mathcal{A}^N} \exp\left[\frac{F(\mathbf{z})}{\nu}\right]}, \forall \mathbf{a} \in \mathcal{A}^N, \tag{2.10}$$

where $\nu > 0$ in (2.10) is called the temperature parameter.

We first introduce the following neighborhood concepts. Note that the neighborhood $\mathcal{N}$ in our problem is not necessarily a mutual concept, as discussed in Sec. 2.3. Thus, we further define the

interaction neighborhood $\mathcal{N}^{(1)}$ as,

$$\mathcal{N}^{(1)}(n) = \{i | n \in \mathcal{N}(i)\} \cup \mathcal{N}(n). \tag{2.11}$$

That is, $\mathcal{N}^{(1)}(n)$ includes both the BSs that affect cell n's utility, i.e. $\mathcal{N}(n)$, and the cells that are affected by BS n. We further define two-hop neighborhood $\mathcal{N}^{(2)}$ as in [**106**],

$$\mathcal{N}^{(2)}(n) = \cup_{i \in \mathcal{N}^{(1)}(n)} \mathcal{N}^{(1)}(i).$$

That is, $\mathcal{N}^{(2)}(n)$ is the union of the interaction neighborhoods of all interaction neighbors of BS n.

Then, by a similar argument as in [**106**], we can define an undirected graph based on the neighborhood concept of $\mathcal{N}^{(2)}$ so that (2.10) is the distribution of a Markov random field on this undirected graph of neighborhood $\mathcal{N}^{(2)}$, and the local specification of the Markov random field is,

$$\begin{aligned} &\Pr\{\hat{A}_n = \mathrm{a}_n | \hat{\mathbf{A}}_{\mathcal{N}^{(2)}(n)\setminus n} = \mathbf{a}_{\mathcal{N}^{(2)}(n)\setminus n}\} \\ =&\frac{\exp\left[\frac{1}{\nu}\sum_{i\in\mathcal{N}^{(1)}(n)} r_i(\mathbf{S}_{\mathcal{N}(i)}, \mathbf{a}_{\mathcal{N}(i)})\right]}{\sum_{\lambda\in\mathcal{A}}\exp\left[\frac{1}{\nu}\sum_{i\in\mathcal{N}^{(1)}(n)} r_i(\mathbf{S}_{\mathcal{N}(i)}, \lambda, \mathbf{a}_{\mathcal{N}(i)\setminus n})\right]}, \end{aligned}$$

where $\mathcal{N}^{(2)}(n) \setminus n$ is a set including all the BSs in $\mathcal{N}^{(2)}(n)$ but excluding n.

Module 1 Gibbs-Sampling-Based Network Configuration

Input: ν, $\tilde{\mathbf{A}}^{(0)}$; N, $\mathcal{N}(\cdot)$, $\mathbf{S}^{(t)}$, $\mathcal{A}$ and $r_n(\cdot)$ for each n.
Output: $\tilde{\mathbf{A}}^{(\tau)}$ for each iteration τ.

1: **Init:** Calculate $\mathcal{N}^{(1)}(n)$ for each BS n by (2.11);
2: **for** each iteration τ **do**
3: $\tilde{\mathbf{A}}^{(\tau)} = \tilde{\mathbf{A}}^{(\tau-1)}$;
4: Pick i uniformly at random from $\{1, 2, \cdots, N\}$;
5: Update $\tilde{A}_i^{(\tau)}$ by picking its value from the action set $\mathcal{A}$ according to the following probability,

$$\begin{aligned} &\forall \mu \in \mathcal{A}, \quad \Pr\{\tilde{A}_i^{(\tau)} = \mu\} \\ =&\frac{\exp\left[\frac{1}{\nu}\sum_{j\in\mathcal{N}^{(1)}(i)} r_j(\mathbf{S}^{(t)}_{\mathcal{N}(j)}, \mu, \tilde{\mathbf{A}}^{(\tau-1)}_{\mathcal{N}(j)\setminus i})\right]}{\sum_{\lambda\in\mathcal{A}}\exp\left[\frac{1}{\nu}\sum_{j\in\mathcal{N}^{(1)}(i)} r_j(\mathbf{S}^{(t)}_{\mathcal{N}(j)}, \lambda, \tilde{\mathbf{A}}^{(\tau-1)}_{\mathcal{N}(j)\setminus i})\right]}; \end{aligned} \tag{2.12}$$

6: **end for**

As a result, inspired by the Gibbs sampling algorithm, we have Module 1. Module 1 aims at solving the problem for one time slot t with a given network state $\mathbf{S}^{(t)}$ using an iterative algorithm.

In this iterative algorithm, in each iteration τ, a vector $\tilde{\mathbf{A}}^{(\tau)} = (\tilde{A}_1^{(\tau)}, \tilde{A}_2^{(\tau)}, \cdots, \tilde{A}_N^{(\tau)})$ is generated, leading to a sequence of output vectors $\tilde{\mathbf{A}}^{(\tau)}$, $\tau = 1, 2, \cdots$. In Module 1, first in Line 1, based on $\mathcal{N}(\cdot)$, the undirected interfering neighborhood $\mathcal{N}^{(1)}$ is constructed. Then, in each iteration τ, $\tilde{\mathbf{A}}^{(\tau)}$ is generated by updating $\tilde{\mathbf{A}}^{(\tau-1)}$ according to the following rules:

i) Randomly select a BS i to update $\tilde{A}_i^{(\tau)}$, as in Line 4.

ii) Update $\tilde{A}_i^{(\tau)}$ according to the probability in (2.12), as in Line 5. That is, for each value μ in $\mathcal{A}$, assign probability as in (2.12) to it, and then select a value from $\mathcal{A}$ randomly according to the corresponding probabilities.

Note that ν, $\nu > 0$, is the temperature parameter as in (2.10). The following results hold.

THEOREM 2.5.1. *The sequence $\tilde{\mathbf{A}}^{(\tau)}$ generated by Module 1 forms a time-homogeneous, irreducible, aperiodic, and reversible Markov chain. In addition, for any initial $\tilde{\mathbf{A}}^{(0)}$,*

$$\tilde{\mathbf{A}}^{(\tau)} \xrightarrow{\text{d}} \hat{\mathbf{A}},$$

and the network utility $F(\tilde{\mathbf{A}}^{(\tau)})$ satisfies,

$$\frac{1}{\tau}\sum_{\tau'=0}^{\tau-1} F(\tilde{\mathbf{A}}^{(\tau')}) \xrightarrow{\text{as}} \text{E}\left[F(\hat{\mathbf{A}})\right],$$

where $\hat{\mathbf{A}}$ is the random vector with the probability distribution in (2.10)*, and $\xrightarrow{\text{d}}$ and $\xrightarrow{\text{as}}$ denote convergence in distribution and almost sure convergence, respectively.*

PROOF. See Appendix 2.8.1. □

Now, let $\pi_\nu(\mathbf{A}_\star^{(t)})$ denote the probability that we choose one of the optimal solutions of the problem in (2.4). Then the following result holds due to the property of Gibbs distribution. (See [**105**] for details.)

THEOREM 2.5.2. *The probability $\pi_\nu(\mathbf{A}_\star^{(t)})$ is a monotonically decreasing function of parameter ν, and,*

$$\lim_{\nu \to 0} \pi_\nu(\mathbf{A}_\star^{(t)}) = 1.$$

REMARK 2.5.1. *It follows from Theorem 2.5.1 and Theorem 2.5.2 that* $\tilde{\mathbf{A}}^{(\tau)}$ *in Module 1 has a steady-state distribution that chooses the optimal solution of the problem defined in* (2.4) *with a probability arbitrarily close to one, if the temperature parameter* ν *is sufficiently small.*

2.5.3. Accelerate Convergence. While it has nice theoretical properties, Module 1 has two practical limitations. First, when implementing Module 1, if ν is too small, computational challenges exist due to numerical overflow because the exponential factors in (2.12) can be extremely large. Second, as stated in Remark 2.5.1, when the temperature parameter ν is small enough, $\tilde{\mathbf{A}}^{(\tau)}$ in Module 1 will converge to the optimal solution. However, a smaller ν leads to a longer convergence time for the corresponding Markov Chain [**106**] in Module 1. In fact, when ν is small enough, the convergence time (i.e., the mixing time of Markov chain) can be prohibitively long.

However, a fast algorithm to solve the problem in (2.4) for one slot t is critical for online network configuration. Clearly, the solution of the problem in (2.4) for each time slot t should be computed within the length of one time slot, while the length of one time slot will not be long since we want to adjust network configuration promptly according to the network state.

For this reason, in this subsection, we focus on a single slot t, and design a fast converging algorithm that converges to a local optimum of the problem in (2.4) for a given network state $\mathbf{S}^{(t)}$. Here, the local optimum is defined as follows.

DEFINITION 1. *A network control configuration* $\mathbf{a}$ *is a local optimum solution of the problem defined in* (2.4) *for a given network state* $\mathbf{S}^{(t)}$ *if* $F(\mathbf{a}) \geq F(\mathbf{a}')$ *holds for any* $\mathbf{a}'$ *such that* $\exists i$ *satisfying* $\mathbf{a}_n = \mathbf{a}'_n, \forall n \neq i$.

The algorithmic design is presented in Module 2. It is inspired by the insight from Gibbs sampling: In Module 1, a smaller temperature parameter ν leads to a larger probability of selecting a local optimum solution of the problem in (2.4), while a larger parameter ν leads to a larger probability of escaping from the trap of a local optimum.

Specifically, instead of randomly choosing one BS to update in each iteration as in Line 4 of Module 1, Module 2 updates the corresponding control variable for each BS in turn in Lines 4-6. In addition, after selecting a BS, Module 2 chooses the value of the control variable of the selected BS in a deterministic way in Line 5. This differs from Module 1 which assigns a probability for each

possible value of the control variable of the selected BS, and chooses its configuration according to these assigned probabilities in Line 5 of Module 1. However, to see the link between Module 2 and Module 1, note that the value of the control variable chosen for a selected BS in Line 5 of Module 2 is the one that maximizes the probability in (2.12) in Line 5 of Module 1 if the same BS is selected. (This can be shown by noticing that the denominator of the probability in (2.12) in Module 1 is the same for different values of $\tilde{A}_i^{(\tau)}$.) This is roughly equivalent to setting the temperature ν to 0 in Module 1[1].

Module 2 Fast Converging Network Configuration

Input: $\mathbf{a}_{\text{init}}^{(\text{old})}$; N, $\mathcal{N}(\cdot)$, $\mathbf{S}^{(t)}$, $\mathcal{A}$ and $r_n(\cdot)$ for each n.
Output: $\mathbf{A}^{(t)}$.

1: **Init:** Let $\mathbf{a}^{(\text{old})} = \mathbf{a}_{\text{init}}^{(\text{old})}$, and calculate $\mathcal{N}^{(1)}(\cdot)$ by (2.11);
2: **repeat**
3: $\quad \mathbf{a}^{(\text{new})} = \mathbf{a}^{(\text{old})}$;
4: $\quad$ **for** $n = 1$ **to** N **do**
5: $\quad\quad \mathrm{a}_n^{(\text{new})} = \arg\max_{\mathcal{A}} \sum_{i \in \mathcal{N}^{(1)}(n)} r(\mathbf{S}_{\mathcal{N}(i)}^{(t)}, \mathbf{a}_{\mathcal{N}(i)}^{(\text{new})})$;
6: $\quad$ **end for**
7: **until** $\mathbf{a}^{(\text{old})} == \mathbf{a}^{(\text{new})}$;
8: $\mathbf{A}^{(t)} = \mathbf{a}^{(\text{new})}$;

Note that in Line 5 of Module 2, ties can be broken by any deterministic rule. Further, with any deterministic tie-breaking rule employed in Line 5, the repeat-until loop in Lines 2-7 of Module 2 terminates within a finite number of iterations. To see this, we first note that upon each update for $\mathbf{a}^{(\text{new})}$ in Line 5, $F(\mathbf{a}^{(\text{new})})$ increases or remains the same. This is because the impact of a_n on $F(\mathbf{a})$ is only manifested through $r_i(\mathbf{S}_{\mathcal{N}(i)}^{(t)}, \mathbf{a}_{\mathcal{N}(i)})$, with $i \in \{j | n \in \mathcal{N}(j)\}$ (recall (2.9)). Then, in Line 5, the change for the n-th element of $\mathbf{a}^{(\text{new})}$ cannot decrease the value of $F(\mathbf{a}^{(\text{new})})$. By combining this with the deterministic tie-breaking rule, the following holds: Either $\mathbf{a}^{(\text{old})}$ equals $\mathbf{a}^{(\text{new})}$ which leads to the termination of the repeat-until loop in Line 7, or $F(\mathbf{a}^{(\text{old})}) < F(\mathbf{a}^{(\text{new})})$. That is, the values of $\mathbf{a}^{(\text{old})}$ in different iterations of the repeat-until loop are different and $F(\mathbf{a}^{(\text{old})})$ keeps increasing. Thus, the $\mathbf{a}^{(\text{old})}$ in different iterations can form a sequence of variables without repetition. Since the number of possible values for $\mathbf{a}^{(\text{old})}$ is finite, the repeat-until loop terminates in a finite number of iterations. Then, the following holds.

[1]There is a slight difference when there are two or more values of the control variable $\tilde{A}_i^{(\tau)}$ that achieve the maximum probability in (2.12), but still one of these optimal values will be chosen when ν is set to 0.

THEOREM 2.5.3. *The output* $\mathbf{A}^{(t)}$ *in Module 2 is a local optimum solution of the problem defined in* (2.4) *on one time slot* t *for a given network state* $\mathbf{S}^{(t)}$.

PROOF. We prove this by contradiction. First, we assume that $\mathbf{A}^{(t)}$ is not a local optimum solution. By Module 2, $\mathbf{A}^{(t)}$ must be the $\mathbf{a}^{(\text{old})}$ in the last repeat-until loop. As a result, the $\mathbf{a}^{(\text{old})}$ in the last iteration for the repeat-until loop in Lines 2-7 is not a local optimum solution.

Then, by Definition 1, there exists $\mathbf{a}'$ and n such that the last $\mathbf{a}^{(\text{old})}$ in the repeat-until loop satisfies, $\mathrm{a}_n^{(\text{old})} \neq \mathrm{a}'_n$, $\mathrm{a}_i^{(\text{old})} = \mathrm{a}'_i$ for $\forall i \neq n$, and that $F(\mathbf{a}^{(\text{old})}) < F(\mathbf{a}')$.

Note that the impact of a_n on $F(\mathbf{a})$ is only manifested through $r_i(\mathbf{S}^{(t)}_{\mathcal{N}(i)}, \mathbf{a}_{\mathcal{N}(i)})$ with $i \in \{j | n \in \mathcal{N}(j)\}$. As a result, combining the above with (2.11) and Line 5 in Module 2, we have $\mathbf{a}^{(\text{new})} \neq \mathbf{a}^{(\text{old})}$ in the last iteration in the repeat-until loop. However, this contradicts the termination condition of the repeat-until loop in Line 7. Thus, the conclusion holds. □

REMARK 2.5.2. *By comparing Remark 2.5.1 and Theorem 2.5.3, we note that Module 2 speeds up the optimization procedure by trading off the performance from a global optimum to a possible local optimum.*

REMARK 2.5.3. *Further, in Sec. 2.6, we find that the repeat-until loop in Module 2 is typically short, e.g., 3-4 steps.*

2.5.4. Online-Learning-Based Joint Network Configuration. Now, we combine Sec. 2.4 and Sec. 2.5.2 to design online-learning-based joint network configuration in Algo. 1. In Algo. 1, let $\mathcal{N}(t, n)$ be the neighborhood of node n in slot t. Note that the neighborhood concept is decided by the impact on one cell's utility (recall the discussion below Eq. (2.3)), and it may change over time. Then, given a network state $\mathbf{S}^{(t)}$ (which includes time-average features that are extracted as discussed in Sec. 2.4.2), we obtain control variables $\mathbf{A}^{(t)}$ by the fast converging algorithm in Module 2 for each time slot t. Note that the underlying condition for the algorithm to perform well is that the state in one time slot remains the same. Due to network dynamics, this requires an adequately short length for the time slot, i.e., the necessity to design a fast-converging configuration algorithm, as discussed in Sec. 2.5.3.

Also, in Lines 6-7 of Algo. 1, we use mini-batch to train the cell utility model $r(\cdot)$ (which is a unified model for all cells, recall Sec. 2.4.1) by stochastic gradient descent. (Specifically, the

Algorithm 1 Online-Learning-Based Joint Network Configuration

Input: $\mathbf{A}^{(0)}$, N, $B_{\max}$, initial cell utility model $r(\cdot)$, and observe $\mathcal{N}(t,n)$, $S_n^{(t)}$, $R_n^{(t)}$ for each n in each slot t.
Output: $A_n^{(t)}$ for each BS n in each slot t.
1: **Init:** Mini-batch $B = \emptyset$;
2: **for** each time slot t **do**
3: Observe neighborhood structure, let $\mathcal{N}(\cdot) = \mathcal{N}(t,\cdot)$, and observe state $\mathbf{S}^{(t)}$;
4: By Module 2, obtain $\mathbf{A}^{(t)}$, with setting $\mathbf{a}_{\text{init}}^{(\text{old})} = \mathbf{A}^{(t-1)}$ and $r_n(\cdot) = r(\cdot)$ for each n;
5: Observe cell utility $R_n^{(t)}$ for each BS n;
6: Add the tuples $(\mathbf{S}_{\mathcal{N}(n)}^{(t)}, \mathbf{A}_{\mathcal{N}(n)}^{(t)}, R_n^{(t)})$ for $n = 1, 2, \cdots, N$ to the mini-batch B; if the size of B exceeds $B_{\max}$, drop the oldest tuples in it;
7: Update the cell prediction model $r(\cdot)$ by stochastic gradient descent to minimize the loss function on the mini-batch;
8: **end for**

Adam method is used due to its strong performance [**107**].) Details about the training model and hyper-parameters are provided in Sec. 2.4.3.

REMARK 2.5.4. *Then, by Theorem 2.5.3, the output* $\mathbf{A}^{(t)}$ *in Algo. 1 is a local optimum solution of the problem defined in* (2.5) *for a given network state* $\mathbf{S}^{(t)}$ *on each slot* t.

To make it clear, we summarize the key steps of pilot power configuration process shown in Algo. 1:

- At the beginning of each time slot t, we observe sample data including raw network status and neighborhood structure[2]. Also, based on raw features, the time-average features can be obtained by moving average, as discussed in Sec. 2.4.2. Then, local network state $\mathbf{S}_{\mathcal{N}(n)}^{(t)}$ for each n is extracted. This process to extract current network state is shown in Line 3 of Algo. 1.
- Then, based on current network state $\mathbf{S}^{(t)}$, the current network control $\mathbf{A}^{(t)}$ can be determined by the joint network configuration module in Module 2. Note that to apply the joint configuration module, the up-to-date cell utility function $r(\cdot)$ is used. This process to determine the current network control based current network state and cell utility function is shown in Line 4 of Algo. 1.

[2] Note that the neighborhood is the 5 neighboring cells that have the most impact on this cell's throughput, as discussed in Sec. 2.4.2.

- Then, the cell utility $R_n^{(t)}$ for each BS n can be observed from the network, as shown in Line 5 of Algo, 1. Note that the cell utilities depend on the current network state $\mathbf{S}^{(t)}$ and the current network control $\mathbf{A}^{(t)}$. As a result, the tuples $(\mathbf{S}_{\mathcal{N}(n)}^{(t)}, \mathbf{A}_{\mathcal{N}(n)}^{(t)}, R_n^{(t)})$ for all n can serve as training data of the cell utility function $r(\cdot)$.
- Consequently, at the end of each time slot, the new training data of tuples $(\mathbf{S}_{\mathcal{N}(n)}^{(t)}, \mathbf{A}_{\mathcal{N}(n)}^{(t)}, R_n^{(t)})$ are used to update the mini-batch B, which collects historical data for online cell utility function training. Then, the cell utility function $r(\cdot)$ are updated by stochastic gradient descent on the updated mini-batch B. Note that the cell utility function $r(\cdot)$ has a neural-network–based learning model, as studied in Sec. 2.4. This procedure for online training of cell utility function is shown in Lines 6-7 of Algo. 1

REMARK 2.5.5. *Note that although we are motivated by the pilot power configuration problem, Algo. 1 can be a general method for the online cellular network configuration.*

2.6. Numerical Results

We evaluate the performance of the proposed approach on an industrial-grade cellular network simulator, Huawei U-Net. U-Net is an integrated component of Huawei's rich wireless network planning and design over years. It follows 3GPP standards and can support various scenarios. To accurately simulate wireless network, U-Net utilizes highly accurate geographic information system (GIS). It builds in high-precision propagation models, such as commonly used models for macro cellular networks in dense urban areas, common urban areas, suburban areas, and rural areas, as well as ray tracing models. It also integrates accurate models for network elements (NEs) and antennas, as well as functions such as MIMO and multi-section splitting. It can plan network resources such as neighboring cells, frequencies, scrambling codes, and PCIs (physical cell identifier). In addition, data of live networks can be imported on the U-Net.

Several methods developed using this industrial-grade network simulation tool has been applied to real networks, such as adding more than 2300 newly planned sites across China, as well as other network problems including refarming planning, network optimization by operators in countries such as China, Norway, United States and United Kingdom.

The simulation scenario here includes 87 cells, reproducing their geographical relationship of a metropolitan area. The scenario is a 3G WCDMA cellular network including typical voice and data traffic. The simulation scenario includes thousands of users. Number of users in each cell are based on data extracted from live network traffic statistics, while the location of each user in a cell is randomized. In the simulation, the total power constraint at each BS is 43dBm, i.e., the sum of the transmission power and the pilot power should be no greater than 43dBm. The pilot power of each BS is chosen from 30dBm to 36dBm with a granularity of 0.5dBm. This sophisticated simulator takes a relatively long time to run, for example, a trace of 900 time slots here can take up to three days to complete.

2.6.1. Learning the Utility Function Online. We first evaluate the prediction accuracy of our online utility training procedure. As shown in Algo. 1, the cell utility function $r(\cdot)$ is updated at the end of each time slot (recall Line 7). Using the utility function updated at the end of slot $t-1$, based on $\mathbf{S}^{(t)}_{\mathcal{N}(n)}$and $\mathbf{A}^{(t)}_{\mathcal{N}(n)}$, we predict the utility at slot t, i.e., $r(\mathbf{S}^{(t)}_{\mathcal{N}(n)}, \mathbf{A}^{(t)}_{\mathcal{N}(n)})$, and compare it with its true value $R^{(t)}_n$ obtained at the end of slot t for all cells. Here, the prediction accuracy is calculated at each time slot for all cells, and measured by the coefficient of determination, i.e., R^2 defined in (3.21). The result is shown in Fig. 2.2.

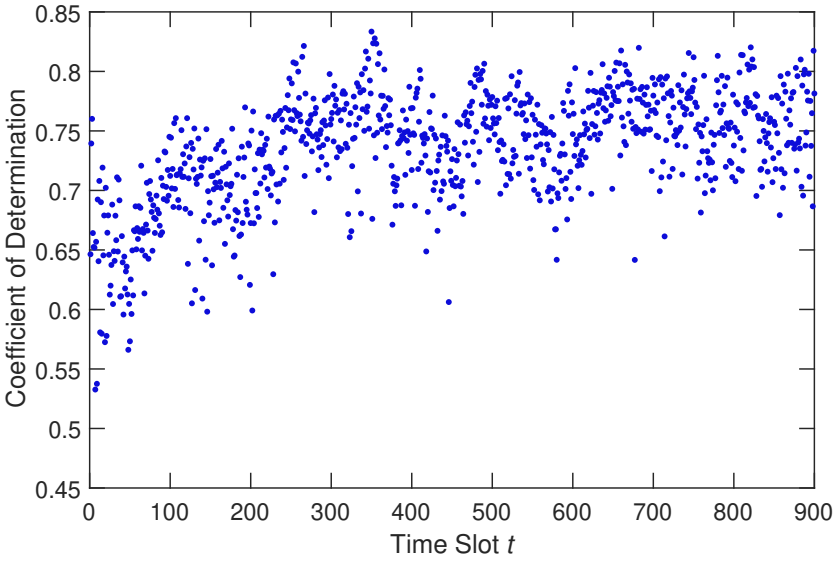


FIGURE 2.2. Online prediction accuracy for the cell utility function.

TABLE 2.5. Prediction accuracy for existing solutions.

	Joint Network State and Action
Coefficient of Determination	-57.9404

Fig. 2.2 shows that after the initial warm-up period, R^2 fluctuates around 0.75, within the range of $[0.6, 0.85]$. This is comparable to the offline prediction accuracy in Sec. 2.4 which is around 0.7. Therefore, our online learning procedure sets a good foundation to conduct the pilot power configuration.

We select the size of the mini-batch B for the online training of the cell utility function $r(\cdot)$ (recall Line 6 of Algo. 1) by comparing the prediction accuracy of the online cell utility function trained with different mini-batch sizes. In Fig. 2.2, this size, i.e. $B_{\max}$, is selected as 700. Further, with $B_{\max} = 700$, considering that there are in total 87 BSs, we note that the mini-batch can be fully updated within 9 time slots. This illustrates how a unified cell utility function $r(\cdot)$ can accelerate the data collection and thus online training.

In Fig. 2.2, we use the offline data generated by the random policy to train the initial $r(\cdot)$ in Algo. 1. At the initial slots, the coefficient of determination is still well above 0.5. This indicates that Algo. 1 can use the sample data of existing algorithms in the system to speed up the learning.

However, note that there is still a gap between the prediction accuracy in the initial slots and that in the subsequent slots. One reason for this performance gap is that the best cell utility function changes due to time-varying network dynamics, as discussed in Sec. 2.3.3, which is not captured in the offline model. Another reason is that the best cell utility function $r(\cdot)$ can be different for different network configuration policies as they generate different distributions of the input features for the model learned.

Table 2.5 compares the prediction accuracy of existing solutions. In most existing work for learning-based network configuration, such as [**28**, **31**, **32**], all relevant features of the entire system are treated as state or action. Thus, in this pilot power configuration problem with 87 cells, using the existing solutions, the network state is a 174-dimensional vector ($2 \times 87 = 174$), which includes load and number of users in each cell. The network action is a 87-dimensional vector that includes pilot powers of all cells. To illustrate the prediction accuracy of such a solution, we train a neural-network-based regression model with the 174-dimensional state vector and 87-dimensional action vector, to predict the total throughput of the network. However, in this method,

the data aggregation in Sec. 2.4.1 cannot be used, which leads to data scarcity: in this 900 iteration simulation, there are in total 900 data points, i.e., 900 combinations of state/action vectors and the corresponding training target (i.e., total throughput). To relieve the data scarcity, for this learning model with joint network state and action vectors, we only provide off-line prediction accuracy achieved in the following manner: For each iteration t, the learning model is trained based on all the history and future data points, i.e., the data collected in all slots $\tau \neq t$, and then, the total network throughput in slot t is predicted based on the network state in slot t by this off-line trained model. Last, based on the resulting prediction of all the 900 time slots, we calculated the coefficient of determination of this method[3].

However, as shown in Table 2.5, the prediction accuracy of this learning model is extremely poor, despite that the future data and current exact network state are provided. Clearly, the poor performance is due to the huge action/state space, as well as the limited data availability: For the 87-dimensional joint network action, the total action space is $13^{87} = 8.186 \times 10^{96}$, and the 174-dimensional joint network state leads to a even larger state space. Meanwhile, there are only 900 data samples available. As a consequence, it is impossible to use the existing solutions to build a reliable learning model for our network configuration problem. This result motivates the need for our proposed online-learning-based joint-optimization framework that intelligently selects state/action space and address data scarcity.

We further design a distributed Q-learning algorithm based on multiagent reinforcement learning theory [**34**,**35**] as a comparable baseline, which is discussed in next section and Appendix 2.8.2.

2.6.2. Conducting Online-Learning-Based Configuration. Now, we compare network utility performance for different configuration algorithms. We consider two baselines. One is a random policy that selects pilot power uniformly at random; the other is a distributed Q-learning algorithm, motivated by the fact that our problem falls into the general scope of reinforcement learning. If conventional Q-learning that treats all BSs as a whole is used, the action and state space is immense: Since there are 13 possible choices for each BS's pilot power configuration and there are in total 87 BSs, the size of the action space for this network configuration problem is $13^{87} = 8.186 \times 10^{96}$, and the size of the state space can be even larger. For this reason, we use

[3]To the readers who are familiar with statistical learning, this is in essence leave-one-out cross validation.

distributed Q-learning. The idea is that each BS runs a local Q-learning algorithm based on its local information: the state includes the local state and the pilot power of the BS, while the reward is the average of the cell's reward and its neighbors' rewards. As a result, the pilot power in the next time slot of each BS is controlled by its own Q-learning algorithm. More details about our distributed Q-learning design are shown in Appendix 2.8.2.

The results of different algorithms are shown in Fig. 2.3. Our Algo. 1 outperforms distributed Q-learning, in terms of both the overall performance and the ramp-up time. It is better in overall performance since it jointly optimizes network control, while in the distributed Q-learning algorithm each cell makes decision locally. Furthermore, Algo. 1 has a much shorter ramp-up time compared to the distributed Q-learning algorithm. This benefits from that Algo. 1 uses data aggregation to learn a unified cell utility function $r(\cdot)$. This enables it to collect more data within each time slot, thus boosting the learning procedure.

In addition, when running the experiments for Fig. 2.3, we found that in each time slot, Algo. 1 typically achieves the local optimum (recall Theorem 2.5.3) within 3-4 iterations. That is, the repeat-until loop in Module 2 typically ends in 3-4 iterations. This illustrates the efficiency of Algo. 1 on quickly deciding the network configuration.

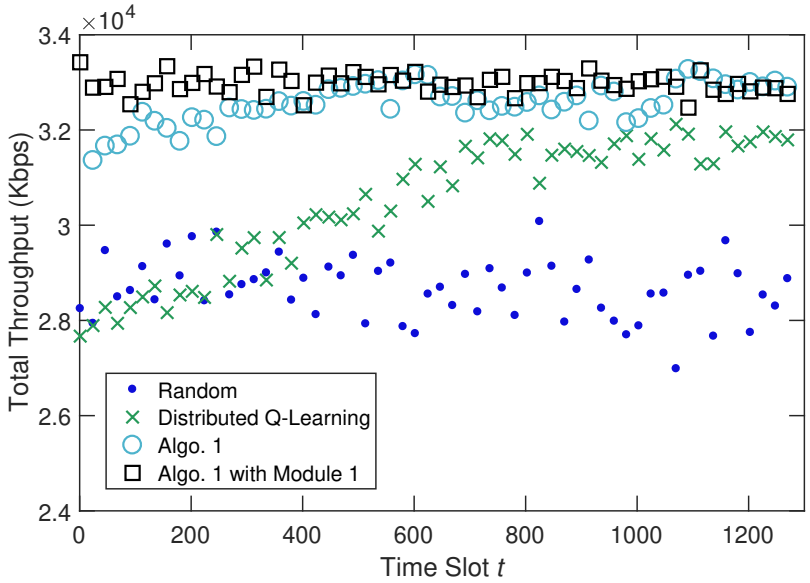


FIGURE 2.3. Total network throughput for different configuration algorithms.

We have also tested the performance of Algo. 1 with Module 1, as shown in Fig. 2.3. In this algorithm, instead of implementing Module 2 in Line 4 of Algo. 1, the Module 1 is implemented. Here, for this algorithm, we terminate the for loop in Module 1 after 87000 (i.e., 1000×87 where 87 is the total number of BSs) iterations, and use the $\tilde{\mathbf{A}}^{(\tau)}$ in the last iteration as $\mathbf{A}^{(t)}$. Further, the initial cell utility function $r(\cdot)$ in this algorithm is the model trained offline on the data generated by Algo. 1. Also, to deal with the limitations of Module 1 as discussed in Sec. 2.5.3, the temperature parameter ν is set to be the smallest one that does not cause numerical overflow. It can be seen that our Algo. 1 still approaches this Module 1-based algorithm (which is slow and with well-trained initial utility model).

To illustrate the robustness of our online-learning-based framework, we have also conducted experiments in other settings with different random user locations, as shown in Fig. 2.4. The result is similar to that in Fig. 2.3 and our Algo. 1 still outperforms distributed Q-learning in both the overall performance and the ramp-up time.

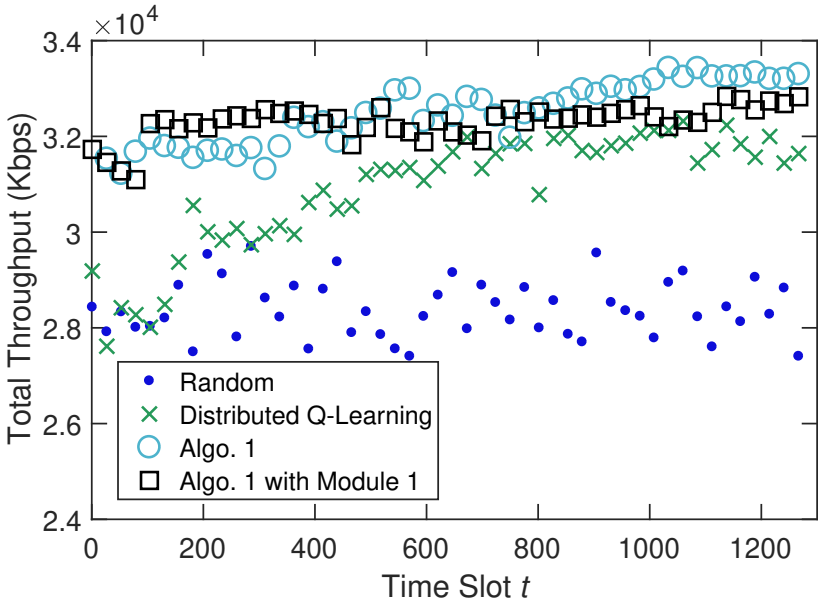


FIGURE 2.4. Performance of configuration algorithms in a different setting.

2.7. Conclusion

In the context of cellular network configuration, a learning-based approach needs to address a few specific challenges, namely, limited data availability, convoluted sample data, highly complex optimization due to interactions among neighboring cells, and the need to adapt to network dynamics. In this work, we develop an online-learning-based joint-optimization approach. In our approach, to learn an appropriate utility function, we develop a neural-network-based model that addresses the convoluted sample data issue and achieves good accuracy based on data aggregation. Based on the utility function learned, we formulate a global network configuration optimization problem. To solve this high-dimensional non-concave maximization problem, we design a Gibbs-sampling-based algorithm that converges to an optimal solution when a temperature parameter is small enough. To speed up its convergence, we further design an efficient algorithm that converges to a local optimum promptly. To adapt to network dynamics, we develop an online scheme that updates the learned utility function and solves the corresponding optimization problem as the network changes. To illustrate the idea, we use the case study of pilot power configuration. The simulation results show that our online utility model achieves a good prediction accuracy, and our online scheme outperforms a benchmark based on a distributed Q-learning algorithm.

The proposed approach has the potential to be applied to other network configuration problems with similar characteristics, such as handoff threshold configuration, antenna adjustment, and transmission power allocation. Future research also includes other feature selection mechanisms for convoluted sample data, incorporating network change detection mechanisms, and generalizing the framework to self-organizing networks (SON).

2.8. Appendix

2.8.1. Proof of Theorem 2.5.1. After we construct the interaction neighborhood $\mathcal{N}^{(1)}$, the problem is similar to that in [**106**]. Also, the proof for the homogeneous, irreducible and aperiodic property of $\tilde{\mathbf{A}}^{(\tau)}$ follows the same argument as that in [**106**], and thus is omitted. However, since our Module 1 differs in the selection of the BS to update, i.e., Line 4, the proof of reversibility is different and is provided as follows.

To prove that the Markov chain $\tilde{\mathbf{A}}^{(\tau)}$ is reversible and that its steady state distribution is (2.10), by reversible Markov chain theory [**108**], we only need to show that $\tilde{\mathbf{A}}^{(\tau)}$ satisfies,

$$\pi_\nu(\mathbf{a})\Pr\left\{\mathbf{a}'|\mathbf{a}\right\} = \pi_\nu(\mathbf{a}')\Pr\left\{\mathbf{a}|\mathbf{a}'\right\} \tag{2.13}$$

where $\pi_\nu(\mathbf{a})$ is as in (2.10), and $\Pr\{\mathbf{a}'|\mathbf{a}\}$ is the transition probability from state $\mathbf{a}$ to state $\mathbf{a}'$, i.e., $\Pr\left\{\tilde{\mathbf{A}}^{(\tau+1)} = \mathbf{a}'|\tilde{\mathbf{A}}^{(\tau)} = \mathbf{a}\right\}$. We prove (2.13) by considering the following three cases:

i) If $\mathbf{a} = \mathbf{a}'$, Eq. (2.13) follows directly.

ii) If $\mathbf{a}$ and $\mathbf{a}'$ differ in more than one element, by Module 1, $\Pr\{\mathbf{a}'|\mathbf{a}\} = \Pr\{\mathbf{a}|\mathbf{a}'\} = 0$, leading to (2.13).

iii) If $\mathbf{a}$ and $\mathbf{a}'$ differ in exactly one element, we assume it is the n-th element without loss of generality. That is, $\mathrm{a}_n \neq \mathrm{a}'_n$ and $\mathrm{a}_i = \mathrm{a}'_i, \forall i \neq n$. Then, $\Pr\{\mathbf{a}'|\mathbf{a}\} =$

$$\frac{1}{N}\cdot\frac{\exp\left[\frac{1}{\nu}\sum_{j\in\mathcal{N}^{(1)}(n)} r_j(\mathbf{S}_{\mathcal{N}(j)}, \mathbf{a}'_{\mathcal{N}(j)})\right]}{\sum_{\lambda\in\mathcal{A}}\exp\left[\frac{1}{\nu}\sum_{j\in\mathcal{N}^{(1)}(n)} r_j(\mathbf{S}_{\mathcal{N}(j)}, \lambda, \mathbf{a}'_{\mathcal{N}(j)\setminus n})\right]}, \tag{2.14}$$

where $1/N$ is the probability that node n is picked (recall Line 4 in Module 1), and the remaining multiplier in (2.14) is the probability that value a'_n is picked (recall Line 5 in Module 1). Similarly, $\Pr\{\mathbf{a}|\mathbf{a}'\} =$

$$\frac{1}{N}\cdot\frac{\exp\left[\frac{1}{\nu}\sum_{j\in\mathcal{N}^{(1)}(n)} r_j(\mathbf{S}_{\mathcal{N}(j)}, \mathbf{a}_{\mathcal{N}(j)})\right]}{\sum_{\lambda\in\mathcal{A}}\exp\left[\frac{1}{\nu}\sum_{j\in\mathcal{N}^{(1)}(n)} r_j(\mathbf{S}_{\mathcal{N}(j)}, \lambda, \mathbf{a}_{\mathcal{N}(j)\setminus n})\right]}. \tag{2.15}$$

By combining (2.14), (2.15) with (2.10), (2.9), we have Eq. (2.13).

Having proved the above properties for the Markov chain $\tilde{\mathbf{A}}^{(\tau)}$, the convergence results follow from standard Markov chain theory (see [**108**]).

2.8.2. Design of Distributed Q-learning Algorithm. For the distributed Q-learning algorithm, each cell conducts its own local Q-learning as follows:

- The local state is the cell's pilot power. (We also tried an alternative local state definition, using the cell's pilot power and the average number of users of the cell and its neighbors. The performance does not change much.)

- The local action consists in changing the local pilot power. Because we work with discrete actions, we usually allow for 3 actions: {-1, 0, +1}. Here, action -1 means a reduction by 0.5 dBm; action 0 denotes no change at all; action +1 means an increase by 0.5 dBm. (We also tried more actions, e.g., {2, -1, 0, +1, +2} to allow for a wider range of decreases or increases, which however does not affect performance in any significant way.)
- The local reward is the tricky part. If we just define the local reward to be the cell's local throughput, then we cannot capture the dependencies among neighboring cells. To alleviate this, we define the local reward as the average of the cell's reward and its neighbors' rewards.

For each local Q-learning, we use a neural network function approximation to represent the local Q-function. Since the action and state space is not large, we are using a neural network with a single hidden layer consisting of just 5 nodes. We experimentally found that the activation function of rectified liner activations units (RLUs) works the best. For the output layer, we just use a single node with a linear activation function to represent the Q-value. The input layer consists of nodes for the state and the action.

We train the neural network using stochastic gradient descent with learning rate 0.01. Finally, regarding the Q-learning we are using a discount factor $\gamma = 0.7$ and ϵ-*greedy exploration* with probability $\epsilon = 0.1$. That is, we follow the action that maximizes the Q value with probability $1 - \epsilon = 0.9$, and a random action with probability $\epsilon = 0.1$. We fine-tuned these values using grid search.

CHAPTER 3

Kernel-Based Multi-Task Contextual Bandits

3.1. Introduction

With the development of mobile Internet and the rising number of smart phones, recent years have witnessed a significant growth in mobile data traffic [**109**]. To satisfy the increasing traffic demand, cellular providers are facing increasing pressure to further optimize their networks. Along this line, one critical aspect is cellular base station (BS) configuration. In cellular networks, a BS is a piece of network equipment that provides service to mobile users in its geographical coverage area (similar to a WiFi access point, but much more complex), as shown in Figure 3.1. Each BS has a large number of parameters to configure, such as spectrum band, power configuration, antenna setting, and user hand-off threshold. These parameters have a significant impact on the overall cellular network performance, such as user throughput or delay. For instance, the transmit power of a BS determines its coverage and affects the throughput of all users it serves.

In current practice, cellular configuration needs manual adjustment and is mostly decided based on the field experience of engineers . Network configuration parameters typically remain static for a long period of time, even years, unless severe performance problems arise. This is clearly not optimal in terms of network performance: different base stations have different deployment environments (e.g., geographical areas), and the conditions of each BS (e.g., the number of users) also change over time. Therefore, as shown in Figure 3.1, setting appropriate parameters for each deployed BS based on its specific conditions could significantly help the industry to optimize its networks. A natural way of achieving this goal is to apply online-learning-based algorithms in order to automate and optimize network configuration.

Online-learning-based cellular BS configuration faces multiple challenges. First, the mapping between network configuration and performance is highly complex. Since different BSs have different deployment environments, they have different mappings between network configuration and

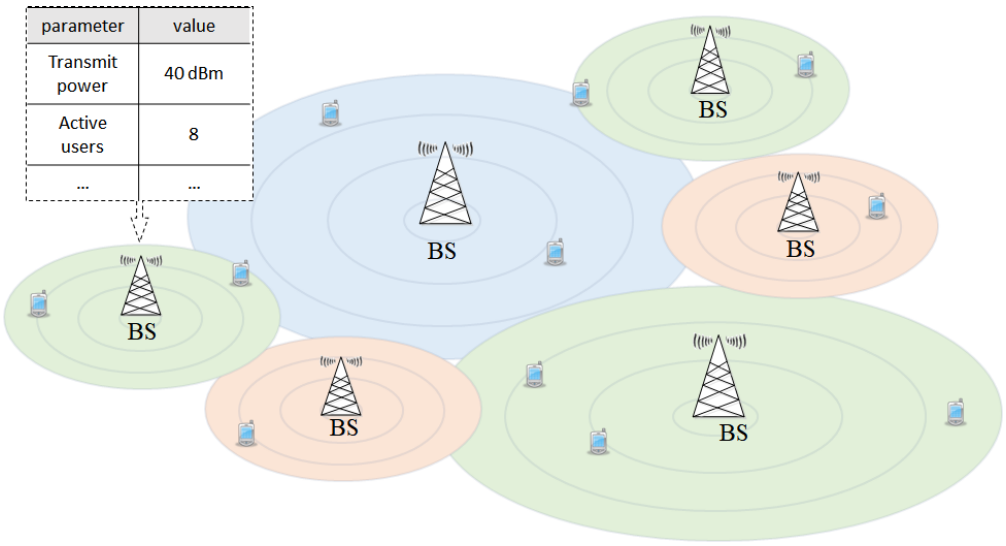


FIGURE 3.1. Cellular network.

performance, given a BS condition. Furthermore, for a given BS, its condition also changes over time due to network dynamics, leading to different optimal configurations at different points in time. In addition, for a given BS and given condition, the impact of network configuration on performance is too complicated to model using white-box analysis due to the complexity and dynamics of network environment, user diversity, traffic demand, mobility, etc. Second, to learn this mapping and to optimize the network performance over a period of time, operators face a fundamental exploitation-exploration tradeoff: in this case, exploitation means to use the best known configuration that benefits immediate performance but may overlook better configurations that are unknown; and exploration means to experiment with unknown or uncertain configurations which may have a better performance in the long run, at the risk of a potentially lower immediate performance. Furthermore, running experiments in cellular networks is disruptive - users suffer poor performance under poor configurations. Thus, providers are often conservative when running experiments and would prefer to reduce the number of explorations needed in each BS. Fortunately, in a cellular network, BSs usually have similarities, even though they are not identical. Therefore, it would be desirable to effectively leverage data from different BSs by exploiting such similarities.

To address these challenges, we consider multiple BSs jointly and formulate the corresponding configuration problem as a multi-task on-line learning framework as shown in Figure 3.2. The key

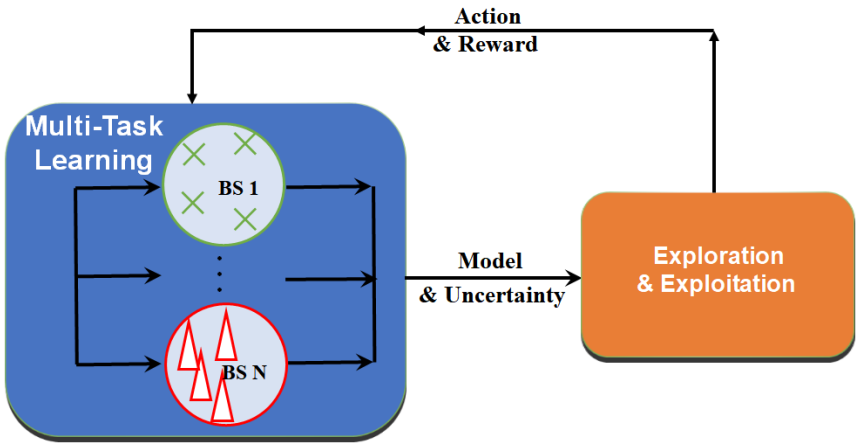


FIGURE 3.2. Multi-task online learning.

idea is to leverage information from multiple BSs to jointly learn a model that maps the network state and its configuration to performance. The model is then customized to each BS based on its characteristics. Furthermore, the model also allows the BSs to balance the tradeoff between the exploration and exploitation of the different configuration. Specifically, we propose a kernel-based multi-BS contextual bandits algorithm that can leverage similarity among BSs to automate and optimize cellular network configuration of multiple BSs simultaneously. Our contributions are multi-fold:

- We develop a kernel-based multi-task contextual bandits algorithm to optimize cellular network configuration. The key idea is to explore similarities among BSs to make intelligent decisions about networks configurations in a sequential manner.
- We propose a method to estimate the similarity among the BSs based on the conditional kernel embedding.
- We present theoretical guarantees for the proposed algorithm in terms of regret and multi-task-learning efficiency.
- We evaluate our algorithm both in synthetic data and real traces data and outperforms bandits algorithms not using multi-task learning by respectively up to 70.8% and 64.8% .

The rest of the paper is organized as follows. The related work is in Sec. 3.2. We introduce the system model and problem formulation in Sec. 3.3. We present a kernel-based multi-BS contextual bandit algorithm in Sec. 3.4. The theoretical analysis of the algorithm is in Sec.3.5. We demonstrate the numerical results in Sec. 3.6, and conclude in Sec. 3.7.

3.2. Related Work

Cellular Network Configuration Various aspects of network parameter configuration have been studied in the literature, such as pilot power configuration, spectrum, handoff threshold, etc. Traditional approaches derive analytical relationship between network configuration and its performance based on communication theory, such as [**24**, **25**, **27**, **110**]. Such approaches are often prohibitively complex, involve various approximations, and require a significant amount of input information (such as the number of users, the location of each user, etc.).

Recently, learning-based methods are proposed [**1**, **28**, **29**, **30**]. In [**28**], the authors propose a tailored form of reinforcement learning to adaptively select the optimal antenna configuration in a time-varying environment. In [**30**], the authors use Q-learning with compact state representation for traffic offloading. In [**29**], the authors design a generalized global bandit algorithm to control the transmit power in the cellular coverage optimization problem. In all these papers, BS similarities are not considered, and thus require more exploration. In [**1**], the authors study the pilot power configuration problem and design a Gibbs-sampling-based online learning algorithm so as to maximize the throughput of users. In comparison, they make the assumption that all BSs are equal while we allow different BSs to learn different mappings.

Contextual Bandits Contextual bandit [**19**] is an extension of classic multi-armed bandit (MAB) problem [**45**]. One type of algorithm is based on upper confident bound(UCB), such as Lin-UCB [**53**], Kernel-UCB [**49**], in which they assume the reward is a function of the context and trade off between the exploitation and exploration based on upper confident bound of the estimation [**111**]. The contextual bandit is also widely used in many application areas, such as news article recommendation [**53**], clinical trials [**54**].

Multi-task Learning Multi-task learning has been extensively studied in many machine learning lectures [**57**]. A common way is using a kernel function to define the similarity among tasks, e.g., in [**63**, **64**, **65**]. In [**65**], the authors design an algorithm that can **transfer information among arms** in the contextual bandit. Compared with [**65**], in our problem, we define an individual contextual bandit problem for each BS and consider the **multi-task learning among different contextual bandit problems**.

3.3. System Model and Problem Formulation

In this section, firstly, we describe the detail of the multi-BS configuration problem. Then we formulate the problem as a multi-task contextual bandits model.

3.3.1. Multi-BS Configuration. We focus on the multi-BS network configuration problem. Specifically, we consider a set of BSs $\mathcal{M} := \{1, \cdots, M\}$ in a network. The time of the system is discretized, over a time horizon of T slots. At time slot t, $\forall t \in \mathcal{T} := \{1, \cdots, T\}$, for each BS $m \in \mathcal{M}$, its state is represented by a vector $s_t^{(m)}$ in state space $\mathcal{S} \subset \mathbb{R}^d$. The state may include the number of users in a BS, user mobility, traffic demand, and neighboring BS configuration. At the beginning of each time slot t, the BS observes its state $s_t^{(m)}$, and chooses its configuration $c_t^{(m)} \in \mathcal{C}_{action} \subset \mathbb{R}$ using a network configuration algorithm, where C_{action} is a finite set. At the end of time slot t, the BS receives a resulting reward $r_{c_t,t}^{(m)} \in \mathbb{R}$, which is a measure of network performance. We note that $c_t^{(m)}$ can depend on all historical information of all BSs and the current state $s_t^{(m)}$.

In practice, the configuration parameters can include pilot power, antenna direction, handoff threshold, etc. The reward can be metrics of network performance, such as uplink throughput, downlink throughput, and quality-of-service scores. Time granularity of the system is decided by network operators. In the current practice, configurations can be updated daily during midnight maintenance hours. To further improve network performance, network operators are moving towards more frequent network configuration updates, e.g., on an hourly basis, based on network states.

The goal of the problem is to find the configuration $c_t^{(m)}$, for all t and m that maximizes the total cumulative reward over time, i.e.,

$$\max_{c_t^{(m)} \in \mathcal{C}_{action}, \forall t} \sum_{m=1}^{M} \sum_{t=1}^{T} r_{c_t,t}^{(m)} \tag{3.1}$$

In this problem, for a given BS and a given state, we do not have *a prior* knowledge of the reward of its action. We need to learn such a mapping during the time horizon. In other words, when choosing $c_t^{(m)}$, one should consider the historical information of all BSs and current state, i.e., $s_\tau^{(m)}, c_\tau^{(m)}, r_{c_\tau,\tau}^{(m)}, \forall \tau < t, \forall m$ and $s_t^{(m)}$. The choice of $c_t^{(m)}$ and the corresponding reward also affect

future actions. Therefore, there exists a fundamental exploitation-exploration tradeoff: exploitation is to use the best learned configuration that benefits the immediate reward but may overlook better configurations that are unknown; and exploration is to experiment unknown or uncertain configurations which may have a better reward in the long run, at the risk of a potentially lower immediate reward.

Furthermore, we note that the action of one BS can be affected by the information of other BSs. Therefore, the information from multiple BSs should be leveraged jointly to optimize the problem in (3.1). Also, note that the BSs are similar but not identical. Therefore, the similarity of BSs need to be explored and leveraged to optimize the network configuration.

In summary, the goal of multi-BS configuration problem is to choose appropriate actions for all time slot and all BSs to maximize the the problem defined in Eq. (3.1).

3.3.2. Multi-Task Contextual Bandit. MAB [**45**] is a powerful tool in a sequential decision making scenario where at each time step, a learning task pulls one of the arms and observes an instantaneous reward that is independently and identically (i.i.d.) drawn from a fixed but unknown distribution. The task's objective is to maximize its cumulative reward by balancing the exploitation of those arms that have yielded high rewards in the past and the exploration of new arms that have not been tried. The contextual bandit model [**19**] is an extension of the MAB in which each arm is associated with side information, called the context. The distribution of rewards for each arm is related to the associated context. The task is to learn the arm selection strategy by leveraging the contexts to predict the expected reward of each arm. Specifically, in the contextual bandit, over a time horizon of T slots, at each time t, environment reveals context $x_{a_t,t} \in \mathcal{X}$ from set $\mathcal{X}$ of contexts for each arm $a \in \mathcal{A}$ from arms set $\mathcal{A} := \{1, 2, \cdots, N\}$, the leaner required to select one arm a_t and then receives a reward $r_{a_t,t}$ from environment. At the end of the time slot t, learner improves arm selection strategy based on new observation $\{x_{a_t,t}, r_{a_t,t}\}$. At time t, the best arm is defined as $a_t^* = \arg\max_{a \in \mathcal{A}} \mathbb{E}(r_{a_t,t}|x_{a_t,t})$ and the corresponding reward is $r_{a_t^*,t}$. The regret at time T is defined as the sum of the gap between the real reward and the optimal reward through the T time slots in Eq. (3.2). The goal of maximization of the accumulative reward $\sum_{t=1}^{T} r_{a_t,t}$ is

equivalent to minimizing the regret[1].

$$R(T) = \sum_{t=1}^{T} (r_{a_t^*,t} - r_{a_t,t}) \tag{3.2}$$

Based on the classical contextual bandit problem, we propose a multi-task contextual bandit model. Consider a set of tasks $\mathcal{M} := \{1, \cdots, M\}$, each task $m \in \mathcal{M}$ can be seen as a standard contextual bandit problem. More specifically, in task m, at each time t, for each arm $a \in \mathcal{A}$, there is an associated context vector $x_{a,t}^{(m)} \in \mathbb{R}^p$. If the arm $a_t^{(m)}$ is pulled as time t for task m, it receives a reward $r_{a_t,t}^{(m)}$. The detail is shown in Problem 1.

Problem 2 Multi-Task Contextual Bandit.

for $t = 1$ to T **do**
  Environment reveals context $x_{a,t}^{(m)} \in \mathcal{X}$ for each arm $a \in \mathcal{A}$ and each task $m \in \mathcal{M}$
  for $\forall m \in \mathcal{M}$ **do**
    Selects and pulls an arm $a_t^{(m)} \in \mathcal{A}$
    Environment reveals a reward $r_{a_t,t}^{(m)} \in [0, 1]$
  end for
  Improves arm selection based on new observations $\{(x_{a_t,t}^{(m)}, r_{a_t,t}^{(m)}) | m \in \mathcal{M}\}$
end for

We also define the best arm as $a_t^{(m)*} = \arg\max_{a \in \mathcal{A}} \mathbb{E}(r_{a_t,t}^{(m)} | x_{a_t,t}^{(m)})$ and the corresponding reward is $r_{a_t^*,t}^{(m)}$. The regret over time horizon T is defined as the sum of the gap between the real reward and the optimal reward through the T time slot among all M tasks in Eq. (3.3). The goal of the problem is to minimize the regret.

$$R(T) = \sum_{m=1}^{M} \sum_{t=1}^{T} \left(r_{a_t^*,t}^{(m)} - r_{a_t,t}^{(m)} \right) \tag{3.3}$$

We can formulate the multi-BSs configuration problem as multi-task contextual bandit. **We regard the configuration optimization problem for one BS as one task.** Specifically, for each BS m, at time t, the context space $\mathcal{X}$ can be represented by a product of state space $\mathcal{S}$ and action space $\mathcal{C}_{action}$. And we index the finite set $\mathcal{C}_{action}$ by arms set $\mathcal{A}$, i.e., use $c_{a,t}$ to represent each possible configuration in time t. Then we define **context associated with arm a is the combination of the state and the action, i.e., $x_{a,t}^{(m)} = (s_t^{(m)}, c_{a,t}^{(m)})$, where $s_t^{(m)} \in \mathcal{S}$ and**

[1]This is pseudo regret [**112**].

$c_{a,t}^{(m)} \in \mathcal{C}_{action}$. Then the goal of finding the best arms which can maximize the total accumulative reward in Eq. (3.1) is equivalent to minimizing the regret defined in Eq. (3.3).

3.4. Methodology

Most existing work on the contextual bandit problems, such as LinUCB [**53**], KernelUCB [**49**] assume the reward is a function of the context, i.e., $r_{a_t,t} = f(x_{a_t,t})$. At each time slot t, these algorithms use the estimated function $\hat{f}(\cdot)$ to predict the reward of each arm according to the context at time t, i.e.,$\{x_{a_t,t}\}_{a\in\mathcal{A}}$. Based on the value and uncertainty of the prediction, they calculate the upper confident bound (UCB) of each arm. Then they select the arm a_t that has the maximum UCB value and then obtains a reward $r_{a_t,t}$. Last, they update the estimated function $\hat{f}(\cdot)$ by the new observation $(x_{a_t}, r_{a_t,t})$.

In our multi-BS configuration problem defined in Eq. (3.1), if we model every BS as an independent classical contextual bandit problem and use the existing algorithm to make its own decision, it would lose information across BSs and thus is not efficient. Specifically, in the training process, it would learn a group of function $\{f^{(m)}|m \in \mathcal{M}\}$ independently and ignore the similarity among them. In practice, the BSs that are configured simultaneously have lots of similar characteristics, such as geographical location, leading to similar reward functions. Furthermore, in the real case, since the configuration parameters have a large impact on the network performance, the cost of experience is expensive. We need an approach to use the data effectively. So, motivated by this observation, we design the kernel-based multi-BS contextual bandits that can leverage the similarity information and share experiences among BSs, i.e., tasks.

In this section, we propose a framework to solve the problem in Eq. (3.3). We start with the regression model. Then we describe how to incorporate it with multi-task learning. Next, we propose kernel-based multi-BS contextual bandits algorithm in Sec.3.4.3. In the last, we discuss the details of task similarity for real data in Sec. 3.4.4.

3.4.1. Kernel Ridge Regression. For the network configure problem, we need to learn a model from historical data that can predict the reward $r_{a_t,t}$ from the context $x_{a_t,t}$. There are two challenges. First, the learned model should capture the non-linear relation between the configuration parameters, state (context) and the network utility (reward) in complex scenarios. Second,

since the learned model is used in the contextual bandit model, it needs to not only offer the mean estimate value of the prediction but also a confidence interval of the estimation that can describe the uncertainty of the prediction. This important feature is used later to trade off exploration and exploration in the bandit model.

To address these two challenges, we use kernel ridge regression to learn the prediction model that can capture non-linear relation and provide an explicit form of the uncertainty of the prediction. Furthermore, intuitively, the kernel function can be regarded as a measure of similarity among data points. which makes it suitable for the multi-task learning into it in Sec. 3.4.2. Let us briefly describe the kernel regression model.

Kernel ridge regression is a powerful tool in supervised learning to characterize the non-linear relation between the target and feature. For a training data set $\{(x_i, y_i)\}_{i=1}^n$, kernel method assumes that there exists a feature mapping $\phi(x) : \mathcal{X} \to \mathcal{H}$ which can map data into a feature space in which a linear relationship $y = \phi(x)^T\theta$ between $\phi(x)$ and y can be observed, where θ is the parameter need to be trained. The kernel function is defined as the inner product of two data vectors in the feature space. $k(x, x') = \phi(x)^T\phi(x'), \forall x, x' \in \mathcal{X}$.

The feature space $\mathcal{H}$ is a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$ with inner product $k < \cdot, \cdot >$. It can be called as the associated reproducing kernel Hilbert space (RKHS) of k, notated by $\mathcal{H}_k$. The goal of kernel ridge regression is to find a function f in the RKHS $\mathcal{H}$ that can minimize the mean squared error of all training data, as shown in Eq. (3.4).

$$\hat{f} = \arg\min_{f \in \mathcal{H}_k} \sum_{i=1}^{n} (f(x_i) - y_i)^2 + \lambda ||f||^2_{\mathcal{H}_k} \tag{3.4}$$

The solution of Eq. (3.4) is (detail derivation in [**113**])

$$f(x) = \boldsymbol{k}_x^T (K + \lambda I)^{-1} \boldsymbol{y} \tag{3.5}$$

where $\boldsymbol{y} = (y_1, \cdots, y_n)^T$, K is the Gram matrix [**114**], i.e., $K_{ij} = k(x_i, x_j)$, $\boldsymbol{k}_x = (k(x, x_1), \cdots, k(x, x_n))^T$ is the vector of the kernel value between all historical data X and the new data, x.

This provides basis for our bandit algorithms. The uncertainty of prediction of the kernel ridge regression is discussed in Sec.3.4.3.

3.4.2. Multi-Task Learning. We next introduce how to integrate kernel ridge regression into multi-task learning which allows us to use similarities information among BSs.

In multi-task learning, the main question is how to efficiently use data from one task to another task. Borrowing the idea from [**57**, **65**], we define the regression functions in the followings:

$$f : \tilde{\mathcal{X}} \to \mathcal{Y} \tag{3.6}$$

where $\tilde{\mathcal{X}} = \mathcal{Z} \times \mathcal{X}$, $\mathcal{X}$ is the original context space, $\mathcal{Z}$ is the task similarity space, $\mathcal{Y}$ is the reward space. For each context $x_{a_t,t}^{(m)}$ of BS m, we can associate it with the task/BS descriptor $z_m \in \mathcal{Z}$, and define $\tilde{x}_{a_t,t}^{(m)} = (z_m, x_{a_t,t}^{(m)})$ to be the augmented context. We define the following kernel function $\tilde{k}$ in (3.7) to capture the relation among tasks.

$$\tilde{k}((z,x),(z',x')) = k_{\mathcal{Z}}(z,z')k_{\mathcal{X}}(x,x') \tag{3.7}$$

where $k_{\mathcal{X}}$ is the kernel defined in original context, and $k_{\mathcal{Z}}$ is the kernel defined in tasks that measures the similarity among tasks/BSs. Then we define the task/BS similarity matrix K_Z as $(K_Z)_{ij} = k_{\mathcal{Z}}(z_i, z_j)$. We discuss the training of this similarity kernel and similarity matrix in Sec.3.4.4.

In the multi-tasks contextual bandit model, at time t, we need to train an arm selection strategy based on the history data we experienced, i.e.,$\{(x_\tau^{(m)}, r_{a_\tau,\tau}^{(m)}) | m \in \mathcal{M}, \tau < t\}$. We formulate a regression problem in Eq. (3.8)

$$\hat{f}_t = \arg\min_{f \in \mathcal{H}_{\tilde{k}}} \sum_{m=1}^{M} \sum_{\tau=1}^{t-1} (f(\tilde{x}_{a_\tau,\tau}^{(m)}) - r_{a,\tau}^{(m)})^2 + \lambda ||f||_{\mathcal{H}_{\tilde{k}}}^2 \tag{3.8}$$

where $\tilde{x}_{a_\tau,\tau}^{(m)}$ is the augmented context of the arm a_τ for task m, which is defined as the combination of the task descriptor z_m and original context $x_{a_t,t}^{(m)}$, i.e., $\tilde{x}_{a_t,t}^{(m)} = (z_m, x_{a_t,t}^{(m)})$.

Then we can get a similar result as Eq. (3.5) in Eq. (3.9) . The only difference is that we use the augmented context $\tilde{x}$ and new kernel $\tilde{k}$ instead of the x and k.

$$\hat{f}_t(\tilde{x}) = \tilde{\boldsymbol{k}}_{t-1}^T(\tilde{x})(\tilde{K}_{t-1} + \lambda I)^{-1}\boldsymbol{y}_{t-1} \tag{3.9}$$

where $\tilde{K}_{t-1}$ is Gram matrix [**114**] of $[\tilde{x}^{(m)}_{a_\tau,\tau}]_{\tau<t,m\in\mathcal{M}}$,
$\tilde{\boldsymbol{k}}_{t-1}(\tilde{x}) = [\tilde{k}(\tilde{x}, \tilde{x}^{(m)}_{a_\tau,\tau})]_{\tau<t,m\in\mathcal{M}}$, and $\boldsymbol{y}_{t-1} = [r^{(m)}_{a_\tau,\tau}]_{\tau<t,m\in\mathcal{M}}$. For the hyper parameter of kernel $k_{\mathcal{X}}$ and the regularization parameter λ, we can use maximum likelihood method to train them. Then we can use Eq.(3.9) to predict the network utility (reward) based on the configured parameter and network state (augmented context).

3.4.3. Kernel-based Multi-BS Contextual Bandits. Next, we introduce how to measure the uncertainty of the prediction in Eq. (3.9). At time T, for a specific task, i.e., BS, $m \in \mathcal{M}$, for a given augmented context $\tilde{x}^{(m)}_{a_T,T}$ of an arm, in order to estimate the uncertainty of the prediction $\hat{f}_T(\tilde{x}^{(m)}_{a_T,T})$, we need to make an assumption that the reward at time T, i.e., $r^{(m)}_{a_T,T}$ and all historical reward data, i.e., $\{r^{(m)}_{a_\tau,\tau})|m \in \mathcal{M}, \tau < T\}$ are all independent random variables. Then we can use McDiarmid's inequality to get an upper confident bound of the predicted value. Since the mathematical derivation of this step is the same as Lemma 1 in [**65**], we only make a minor modification to obtain Theorem 3.4.1.

THEOREM 3.4.1. *For task $\forall m \in \mathcal{M}$, suppose the rewards $r^{(m)}_{a_T,T}$ at time T and the history reward $\{r^{(m)}_{a_\tau,\tau})|m \in \mathcal{M}, \tau < T\}$ are independent random variables with means $E[r^{(m)}_{a_\tau,\tau}|\tilde{x}^{(m)}_{a_\tau,\tau}] = f^*(\tilde{x}^{(m)}_{a_\tau,\tau})$, where $f^* \in \mathcal{H}_{\tilde{k}}$ and $||f^*||_{\mathcal{H}_{\tilde{k}}} \leq c$. Let $\alpha = \sqrt{\frac{\log(2((T-1)MN+1)/\delta)}{2}}$ and $\delta > 0$. With probability at least $1 - \frac{\delta}{T}$, we have that $\forall a \in \mathcal{A}$*

$$|\hat{f}_t(\tilde{x}^{(m)}_{a,t}) - f^*(\tilde{x}^{(m)}_{a,t})| \leq (\alpha + c\sqrt{\lambda})\sigma^{(m)}_{a,t} \tag{3.10}$$

where the width is

$$\sigma^{(m)}_{a,t} = \sqrt{\tilde{k}(\tilde{x}^{(m)}_{a,t}, \tilde{x}^{(m)}_{a,t}) - \tilde{\boldsymbol{k}}^T_{t-1}(\tilde{x}^{(m)}_{a,t})(\tilde{K}_{t-1} + \lambda I)^{-1}\tilde{\boldsymbol{k}}_{t-1}(\tilde{x}^{(m)}_{a,t})} \tag{3.11}$$

Based on Theorem 3.4.1, we define the upper confident bound UCB for each arm for each task in Eq. (3.12), where $\hat{f}_t$ is obtained from Eq. (3.9), and β is a hyper parameter.

$$\text{UCB}^{(m)}_{a,t} = \hat{f}_t(\tilde{x}^{(m)}_{a,t}) + \beta\sigma^{(m)}_{a,t} \tag{3.12}$$

Then we propose Algorithm 1 to solve the multi-BS configuration problem.

Algorithm 1 Kernel-based multi-BS configuration.

1: **for** $t = 1$ to T **do**
2: Update the Gram matrix $\tilde{K}_{t-1}$
3: **for** all BS $m \in \mathcal{M}$ **do**
4: Observe system state at time t for BS m: $s_t^{(m)}$ and determine the context feature $x_{a,t}^{(m)}$ for each $a \in \mathcal{A}$
5: Determine the task/BS descriptor z_m and get the augmented context $\tilde{x}_{a,t}^{(m)}$
6: **for** all arm a in $\mathcal{A}$ at time t **do**
7: $ucb_{a,t}^{(m)} = \hat{f}(x_{a,t}^{(m)}) + \beta\sigma_{a,t}^{(m)}$
8: **end for**
9: For BS m, choose arm $a_t^{(m)} = \arg\max ucb_{a,t}^{(m)}$
10: Observe reward $r_{a_t,t}^{(m)}$
11: **end for**
12: Update y_t by $\{r_{a_t,t}^{(m)} | m \in \mathcal{M}\}$
13: **end for**

In Algorithm 1, at each time t, it updates the prediction model $\hat{f}_t$. Then for each task $m \in \mathcal{M}$, it uses the model to obtain the UCB of each arm $a \in \mathcal{A}$. Next it selects the arm that has the maximum UCB. Algorithm 1 can trade off between the exploitation and exploration in the multi-BS configuration problem. The intuition behind it is as following: if one configuration is only tried for few times or even yet tried, its corresponding arm's width defined in Eq.(3.11) is larger, which makes its UCB value larger, then this configuration will be tried in following time with high probability.

Independent Assumption Note that the independent assumption of Theorem 3.4.1 is not true in Algorithm 1, because the previous rewards influence the arm selection strategy (prediction function), then influence the following reward. To address it, we select a subset of them to make this assumption hold true in Sec. 3.5.

High Dimensionality In Algorithm 1, it updates $\tilde{K}_{t-1}$ in line 2 and recalculates $(\tilde{K}_{t-1}+\lambda I)^{-1}$ in line 7 based on Eq. (3.9). Since at time t, the dimension of $\tilde{K}_{t-1}$ is $M(t-1)$ and the computation complexity of inverse it is $O(M^3t^3)$. It increases dramatically over time. To address this issue, we use the Schur complement [**115**] as following to simplify it.

THEOREM 3.4.2. *For a matrix* $M = \begin{bmatrix} A & U \\ V & C \end{bmatrix}$, *define Schur complement of block* C *as* $S := A - UC^{-1}V$. *Then we can get*

$$M^{-1} = \begin{bmatrix} A & U \\ V & C \end{bmatrix}^{-1} = \begin{bmatrix} S^{-1} & -S^{-1}UC^{-1} \\ -C^{-1}VS^{-1} & C^{-1}VS^{-1}UC^{-1} + C^{-1} \end{bmatrix} \tag{3.13}$$

Based on it, we can update $(\tilde{K}_t + \lambda I)^{-1}$ by $(\tilde{K}_{t-1} + \lambda I)^{-1}$. It decreases the computation complexity to $O(Mt^2)$.

For the issue of dealing with large dimension of Gram matrix K has been much studied in Chapter 8 of [**116**]. Most of them are designed for the supervise learning cases. In our problem, based on thr feature of online learning, Schur complement method is more suitable and efficiency.

3.4.4. Similarity. The kernel $k_{\mathcal{Z}}(z, z')$ that defines the similarities among the tasks/BSs plays a significant role in Algorithm 1. When $k_{\mathcal{Z}}(z, z') = \mathbb{1}(m = m')$, where $\mathbb{1}$ is the characteristic function, Algorithm 1 is equivalent to running the contextual bandit independently for each BS. In this section, we discuss how to measure the similarity in real data if it is not provided.

Suppose the ground truth function for task i (i.e., BS i) is $y = f_i(x)$, we need to define the similarity among different BSs based on the ground truth functions $f_i(x)$. From a Bayesian view, $y = f_i(x)$ is equivalent to the conditional distribution $P(Y_i|\mathcal{X}_i)$. Therefore, we can use the conditional kernel embedding to map the conditional distributions to operators in a high-dimensional space, and then define the similarity based on it. Let us start with the definition of kernel embedding and conditional kernel embedding.

3.4.4.1. ***Conditional kernel embedding.*** Kernel embedding is a method in which a probability is mapped to an element of a potentially infinite dimensional feature spaces, i.e., a reproducing kernel Hilbert space (RKHS) [**117**]. For a random variable in domain $\mathcal{X}$ with distribution $P(X)$, suppose $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the positive definite kernels with corresponding RKHS $\mathcal{H}_{\mathcal{X}}$, the kernel embedding of a kernel k for X is defined as

$$\nu_x = \mathbb{E}_X[k(\cdot, x)] = \int k(\cdot, x) dP(x) \tag{3.14}$$

It is an element in $\mathcal{H}_{\mathcal{X}}$.

For two random variable X and Y, suppose $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ are respectively the positive definite kernels with corresponding RKHS $\mathcal{H}_\mathcal{X}$ and $\mathcal{H}_\mathcal{Y}$. The kernel embedding for the marginal distribution $P(Y|X = x)$ is:

$$\nu_{Y|x} = \mathbb{E}_Y[l(\cdot, y)|x] = \int l(\cdot, y) dP(y|x) \tag{3.15}$$

It is an element in $\mathcal{H}_\mathcal{Y}$. Then for the conditional probability $P(Y|X)$, the kernel embedding is defined as a conditional operator $O_{Y|X} : \mathcal{H}_\mathcal{X} \to \mathcal{H}_\mathcal{Y}$ that satisfies Eq. (3.16)

$$\nu_{Y|x} = O_{Y|X} k(x, \cdot) \tag{3.16}$$

If we have a data set $\{x_i, y_i\}_{i=1}^n$, which are i.i.d drawn from $P(X, Y)$, the conditional kernel embedding operator can be estimated by:

$$\hat{O}_{Y|X} = \Psi(K + \lambda I)^{-1} \Phi \tag{3.17}$$

where $\Psi = (l(y_1, \cdot), \cdots, l(y_n, \cdot))$ and $\Phi = (k(x_1, \cdot), \cdots, k(x_n, \cdot))$ are implicitly formed feature matrix, K is the Gram matrix of x, i.e., $(K)_{ij} = k(x_i, x_j)$

The definition of conditional kernel embedding provides a way to measure probability $P(Y|X)$ as an operator between the spaces $\mathcal{H}_\mathcal{Y}$ and $\mathcal{H}_\mathcal{X}$.

3.4.4.2. ***Similarity Calculation***. In this section, we use the conditional kernel embedding to define the similarity space $\mathcal{Z}$ and augmented context kernel $k_\mathcal{Z}$ in Eq. (3.7).

We define the task/BS similarity space as $\mathcal{Z} = P_{\mathcal{Y}|\mathcal{X}}$, the set of all conditional probability distributions of Y given X. Then for task/BS m, given a context $x_{a,t}^{(m)}$ for arm a at t, we define the augmented context $\tilde{x}_{a,t}^{(m)}$ as $(P_{\mathcal{Y}_m|\mathcal{X}_m}, x_{a,t}^{(m)})$.

Then we use the Gaussian-form kernel based on the conditional kernel embedding to define $k_\mathcal{Z}$:

$$k_\mathcal{Z}(P_{\mathcal{Y}_m|\mathcal{X}_m}, P_{\mathcal{Y}_{m'}|\mathcal{X}_{m'}}) = \exp(-||O_{Y|X}^{(m)} - O_{Y|X}^{(m')})||^2 / 2\sigma_Z^2) \tag{3.18}$$

where $||\cdot||$ is Frobenius norm, $O_{Y|X}$ is the conditional kernel embedding defined in Eq. (3.16) and can be estimated by Eq. (3.17). The hyper parameter σ_Z can be heuristically estimated by the median of Frobenius norm of all dataset. In Eq. (3.17), it can only be used in explicit kernels. Next, we use the kernel trick to derive a form that does not include explicit features.

Given a group of data sets $D_m = \{(x_i, y_i)\}_{i=1}^{n_m}$, and k and l are respectively two positive definite kernels with RKHS $\mathcal{H}_\mathcal{X}$ and $\mathcal{H}_\mathcal{Y}$, for data set D_m, we define $\Psi_m = (l(y_1, \cdot), \cdots, l(y_{n_m}, \cdot))$ and $\Phi_m = (k(x_1, \cdot), \cdots, k(x_{n_m}, \cdot))$ are implicitly formed feature matrix of y and x. $K_m = \Phi_m^T \Phi_m$ and $L_m = \Psi_m^T \Psi_m$ are Gram matrix of all x and y. and $\mathcal{O}_{Y|X}^{(m)}$ for conditional kernel embedding. According to Eq. (3.17), we have

$$\mathcal{O}_{Y|X}^{(m)} = \Psi_m (K_m + \lambda I)^{-1} \Phi_m^T$$

$$\begin{aligned} ||\mathcal{O}_{Y|X}^{(m)} - \mathcal{O}_{Y|X}^{(m')}||^2 =& tr(\mathcal{O}_{Y|X}^{(m)T} \mathcal{O}_{Y|X}^{(m)}) - 2tr(\mathcal{O}_{Y|X}^{(m)T} \mathcal{O}_{Y|X}^{(m')}) \\ &+ tr(\mathcal{O}_{Y|X}^{(m')T} \mathcal{O}_{Y|X}^{(m)}) \end{aligned} \tag{3.19}$$

Define matrix $K_{mm'}$ and $L_{mm'}$ by $(K_{mm'})_{ij} = k(x_i, x_j)$ and $(L_{mm'})_{ij} = l(y_i, y_j)$, where (x_i, y_i) is the i-th data in $\mathcal{D}_m$ and (x_j, y_j) is the j-th data in $\mathcal{D}_{m'}$, so as $K_{mm'}$ and $L_{m'm}$. Then for the second term in Eq. (3.19),

$$\begin{aligned} tr(\mathcal{O}_{Y|X}^{(m)T} \mathcal{O}_{Y|X}^{(m')}) &= tr(\Psi_m (K_m + \lambda I)^{-1} \Phi_m^T \Phi_{m'} (K_{m'} + \lambda I)^{-1} \Psi_{m'}^T) \\ &= tr((K_m + \lambda I)^{-1} \Phi_m^T \Phi_m (K_{m'} + \lambda I)^{-1} \Psi_{m'}^T \Psi_m) \\ &= tr((K_m + \lambda I)^{-1} K_{mm'} (K_{m'} + \lambda I)^{-1} L_{m'm}) \end{aligned}$$

After using the same trick for other terms, Eq. (3.19) can be written as

$$\begin{aligned} ||\mathcal{O}_{Y|X}^{(m)} - \mathcal{O}_{Y|X}^{(m')}||^2 &= tr((K_m + \lambda I)^{-1} K_m (K_m + \lambda I)^{-m} L_m) \\ &- 2 * tr((K_m + \lambda I)^{-1} K_{mm'} (K_{m'} + \lambda I)^{-1} L_{m'm} \\ &+ tr((K_{m'} + \lambda I)^{-1} K_{m'} (K_{m'} + \lambda I)^{-1} L_{m'}) \end{aligned} \tag{3.20}$$

Then we can use Eq. (3.20) in Eq. (3.18) to measure the similarity between tasks. We denote the conditional kernel embedding metric for measure similarity as CKE.

3.4.4.3. ***Other similarity metrics.*** In the above, the similarity is defined based on $P(Y|X)$ among tasks through conditional kernel embedding. In the practice, there are several ways to define similarity.

The average $\mathbf{R^2}$ method: For example, for data set $\mathcal{D}_1$ and $\mathcal{D}_2$, we can train a regression model on $\mathcal{D}_1$ and test it on $\mathcal{D}_2$, then measure the similarity using the prediction accuracy.

Specifically, in the test set, we can measure prediction through the coefficient of determination R^2 as,

$$R^2 = 1 - \frac{\eta_{\text{ss}}}{\eta_{\text{var}} M_{\text{sp}}}, \tag{3.21}$$

where η_{ss} is the sum of squared prediction errors, η_{var} is the variance of the target, and M_{sp} is the total number of samples. The larger the value of R^2, the better the model can capture the observed outcomes. Switch the training and testing data, we obtain another R^2. Then we can define the similarity base on the average of these two R^2.If the value is smaller than a negative threshold, we can define the similarity as 0.

The hyper parameter method In work [**63**], it regards the similarity as a covariance matrix among tasks, they train them with other hyper parameter in the model based on maximum likelihood metric. This method needs more computation resource.

In practice, using different similarity definitions may have different results. The selection of the method to define similarity is in general heuristic. In this problem, we have conducted experiments using different similarity definitions in the evaluation section . It turns out that conditional kernel embedding (CKE) has the best performance in this kernel-based multi-BS configuration problem, and thus described in detail in this section.

3.5. Theoretical Analysis

In this section, we provide theoretical analysis of Algorithm 1 based on the classical bandit analysis. The first part is about regret analysis and the second part is about the multi-task-learning efficiency.

3.5.1. Regret Analysis. In Algorithm 1, at each time slot t, it uses the trained model to make a decision for all BSs in parallel. This is not in the same form of classical bandit model. In order to simply the analysis, we make an sequential version in Algorithm 2, in which at each time t, it receives the context (state and action) of one BS with its BS ID, denoted by V_t, that is used to identify the BS index m. Then algorithm 2 obtains the augment context using V_t and then makes a decision for the BS. In this manner, Algorithm 2 makes a decision for all BSs sequentially. The performance of parallel and sequential methods are similar when the number of BSs is moderate and

all BSs come in order, as in our case, since the difference of number of updates for the model in the parallel and sequential cases is small. It is also shown from the simulation that their performances are similar.

Algorithm 2 Sequential multi-BS configuration.

for $t = 1$ to T **do**
  Update the Gram matrix $\tilde{K}_{t-1}$
  Observe the BS ID V_t and the corresponding context features at time t: $x_{a,t}$ for each $a \in \mathcal{A}$
  Determine the BS descriptor z_m based on V_t and get the augmented context $\tilde{x}_{a,t}$
  for all arm a in $\mathcal{A}$ at time t **do**
    $ucb_{a,t} = \hat{f}(\tilde{x}_{a,t}) + \beta\sigma_{a,t}$
  end for
  Choose arm $a_t = \arg\max ucb_{a,t}$ for BS V_t
  Observe reward $r_{a_t,t}$
  Update y_t by $r_{a_t,t}$
end for

The regret of Algorithm 2 is defined by

$$R(T) = \sum_{m=1}^{M} \sum_{t=1}^{T} (r_{a_t^*,t}^{(m)} - r_{a_t,t}^{(m)}) \mathbb{1}(V_t = m) \tag{3.22}$$

In Algorithm 2, the estimated reward $\hat{r}_{a_t,t}$ at time t can be regarded as the sum of variables in history $[r_{a_\tau,\tau}]_{\tau<t}$ that are dependent random variables. It does not meet the assumption in Theorem 3.4.1, thus we are unable to analysis the uncertainty of the prediction.

To address this issue, as in [**45**, **111**], we design the base version (Algorithm 3) and super version (Algorithm 4) of Algorithm 2 in order to meet the requirement of Theorem 3.4.1.These algorithms are only designed too help theoretical analysis. In Algorithm 4, it constructs special, mutually exclusive subsets $\{\Psi(s)\}_S$ of ts the elapsed time to guarantee the event $\{t \in \Psi_{t+1}^{(s)}\}$ is independent of the rewards observed at times in $\Psi_t^{(s)}$. On each of these sets, it uses Algorithm 3 as subroutine to obtain the estimated reward and width of the upper confident bound which is the same as Algorithm 2.

The construction of Algorithm 3 and Algorithm 4 follow similar strategy of that in the proof of KernelUCB (see Theorem 1 in [**49**] or Theorem 1 in [**65**]). Then we can get the following theorem 3.

Algorithm 3 Base sequential multi-BS configuration.

Input: $\beta, \Psi \subset \{1, \cdots, t-1\}$
Calculate Gram matrix $\tilde{K}_\Psi$ and get $y_\Psi = [r_{a_\tau,\tau}]_{\tau \in \Psi}$
Observe the BS ID V_t and corresponding context features at time t: $x_{a,t}$ for each $a \in \mathcal{A}$
Determine the BS descriptor z_m and get the augmented context $\tilde{x}_{a,t}$
for all arm a in $\mathcal{A}$ at time t **do**
  $\sigma_{a,t} = \sqrt{\tilde{k}(\tilde{x}_{a,t}, \tilde{x}_{a,t}) - \tilde{k}_{a,\Psi}^T (\tilde{K}_\Psi + \lambda I)\tilde{k}_{a,\Psi}}$
  $ucb_{a,t} = \hat{f}(x_{a,t}) + \beta\sigma_{a,t}$
end for

Algorithm 4 Super sequential multi-BS configuration.

Input: $\beta, T \in \mathbb{N}$
Initialize $S \leftarrow \log\lceil T \rceil$ and $\Psi_1^{(s)} \leftarrow \emptyset$ for all $s \in S$
for $t = 1$ to T **do**
  $s \leftarrow 1$ and $\hat{\mathcal{A}}_1 \leftarrow \mathcal{A}$
  repeat
    $\sigma_{a,t}, ucb_{a,t}$ for all $a \in \hat{\mathcal{A}}_{(s)} \leftarrow \text{BaseAlg}(\Psi_t^{(s)}, \beta)$
    $\omega_{a,t} = \beta\sigma_{a,t}$
    if $\omega_{a,t} \leq \frac{1}{\sqrt{T}}$ for all $a \in \hat{\mathcal{A}}_{(s)}$ **then**
      Choose $a_t = \arg\max_{a \in \hat{\mathcal{A}}_{(s)}} ucb_{a,t}$
      $\Phi_{t+1}^{(s)} \leftarrow \Phi_t^{(s)}$ for all $s \in S$
    else if $\omega_{a,t} \leq 2^{-s}$ for all $a \in \hat{\mathcal{A}}_{(s)}$ **then**
      $\hat{\mathcal{A}}_{s+1} \leftarrow \{a \in \hat{\mathcal{A}}_s | ucb_{a,t} \geq \max_{a' \in \hat{\mathcal{A}}_s} ucb_{a',t} - 2^{1-q}\}$
      $s \leftarrow s + 1$
    else
      Choose $a_t \in \hat{\mathcal{A}}_s$ s.t. $\omega_{a_t,t} > 2^{-q}$
      $\Phi_{t+1}^{(s)} \leftarrow \Phi_t^{(s)} \cup \{t\}$ and $\forall s' \neq s, \Phi_{t+1}^{(s)} \leftarrow \Phi_t^{(s)}$
    end if
  until a_t is found
  Observe reward $r_{a_t,t}$
end for

THEOREM 3.5.1. *Assume that $r_{a,t} \in [0,1], \forall a \in \mathcal{A}, T \geq 1, ||f^*||_{\mathcal{H}_{\tilde{k}}} \leq c_{\tilde{k}}, \forall \tilde{x} \in \tilde{X}$ and tasks similarity matrix K_Z is known. With probability $1-\delta$, the regret of Algorithm 4 satisfies,*

$$\begin{aligned} R(T) &\leq 2\sqrt{T} + 10\left(\sqrt{\frac{\log(2TN(\log(T)+1)/\delta))}{2}} + c\sqrt{\lambda}\right) \\ &\sqrt{2d\log(g([T])}\sqrt{T\lceil\log(T)\rceil} \\ &= O(\sqrt{T\log(g([T]))}) \end{aligned} \tag{3.23}$$

where $g([T]) = \frac{det(\tilde{K}_{T+1}+\lambda I)}{\lambda^{T+1}}$ and $d = max(1, \frac{c_{\tilde{k}}}{\lambda})$

3.5.2. Multi-task-learning Efficiency. In this section, we discuss the benefits of multi-task learning from the theoretical view point.

In the sequential setting, i.e., Algorithm 2 and Algorithm 4, because all BSs/tasks come in order, at time t, each task happens $n = \frac{t}{M}$ times. Let K_{X_t} be Gram matrix of $[x_{a_\tau,\tau}^{(m)}]_{\tau \leq t, m \in \mathcal{M}}$ i.e., original context, K_Z be the similarity matrix. Then, following Theorem 2 in [**65**], the following results hold,

THEOREM 3.5.2. *Define the rank of matrix $K_{X_{T+1}}$ as r_x and the rank of matrix K_Z as r_z. Then*

$$\log(g([T])) \leq r_z r_x \log\left(\frac{(T+1)c_k^{\sim} + \lambda}{\lambda}\right)$$

According to Eq. (3.23), if the rank of similarity matrix is lower, which means all BSs/tasks have higher inter-task similarity, the regret bound is tighter.

We make the further assumption that all distinct tasks are similar to each other with task similarity equal to μ. Define $g_\mu([T])$ as the corresponding value of $g([T])$ when all task similarity equal to μ. According to Theorem 3 in [**65**], we have

THEOREM 3.5.3. *If $\mu_1 \leq \mu_2$, then $g_{\mu_1}([T]) \geq g_{\mu_2}([T])$*

This shows that given the assumption that all tasks comes in order and number of tasks is fixed, when BSs/tasks are more similar, the regret bound is tighter. In our case, running all task independently is equivalent to setting the similarity as an identify matrix, i.e., $\mu = 0$. So, based on the previous two theorems, we show the benefits of our algorithm using the multi-task learning.

3.6. Evaluation

In this section, we evaluate the performance of the proposed approach Algorithm 1. and Algorithm 2. in both synthetic data and real network data.

3.6.1. Synthetic data evaluation. We use synthetic data to demonstrate the impact of similarity in multi-task regression. Thereafter, we test our algorithm performance based on synthetic data.

3.6.1.1. ***Similarity in regression.*** We generate the reward function of tasks with pre-defined ground truth similarity based on Gaussian process. Then we train the regression model using different similarity and measure the performance of regression. In detail, we generate 2-task data sets in the following manner: (1) Each data set has 100 data points, $D_1 = \{x_i^1, y_i^1\}_{i=1}^{100}$ and $D_2 = \{x_i^2, y_i^2\}_{i=1}^{100}$, and each x_i is randomly sampled from $[0,1] \times [0,1] \subset \mathbb{R}^2$ and $y \in \mathbb{R}$. (2) The ground truth similarity between two tasks is $sim_g = 0.8$. i.e., the similarity matrix K_Z is a symmetric 2×2 matrix with 1s in the main diagonal and 0.8s in the anti-diagonal. (3) The kernel of x is the Gaussian kernel with lengthscale 0.5. (4) $\mathbf{y} = [y_1^1, y_2^1, \cdots, y_{100}^1, y_1^2, y_2^2, \cdots, y_{100}^2]^T$ is sampled from a multivariate normal distribution with zero mean and whose covariance is the Kronecker product of similarity matrix K_Z and the Gram matrix of x, K_X added white noise, i.e., $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, K_Z \otimes K_X + \sigma_{noise}^2 \mathbf{I})$ with $\sigma_{noise}^2 = 0.05$. (5) We sampled Y for 100 times, and test the regression for each sampled Y. (6) For each task, the size of train set is 5, other 95 data points are test data.

In the training process, the hyper parameter of the kernel are the same as the ones in the data generating process. For any similarity value $sim_{train} \in [0,1]$ with granularity 0.01 between two tasks, we use Eq. (3.9) to train the regression function. The performance is measured by mean square error (MSE) for all test data. The results is shown in Fig. 3.3. The MSE is the the average of 100 samples $\mathbf{y}$. It shows that the relation between MSE and similarity sim_{train} is a convex form function. The case $sim_{train} = 0$ is to train two tasks independently, that is, no information is shared between tasks; The case $sim_{train} = 1$ is to train two tasks with the combination of the two data sets, that is, the difference between tasks is neglected. The best performance (minimum MSE) is achieved, when $sim_{train} = sum_g = 0.8$, that is, similarity used in training is equal to the ground truth similarity. This is in accordance with our motivation to take the similarity measurement into the multi-task learning.

3.6.1.2. ***Multi-task contextual bandit in synthetic data.*** We use synthetic data to test the performance of Algorithm 1 based on different similarity metrics in Sec. 3.4.4, the CKE, the average R^2 method and the hyper parameter method. Suppose that we have 5 tasks and 5 arms for each task, and define the context for each arm as $x_{a_t,t}^{(m)} \in \mathbb{R}^2$. To create the similar reward function for each task, we assume that there exits a hidden parameter u_t, which is randomly

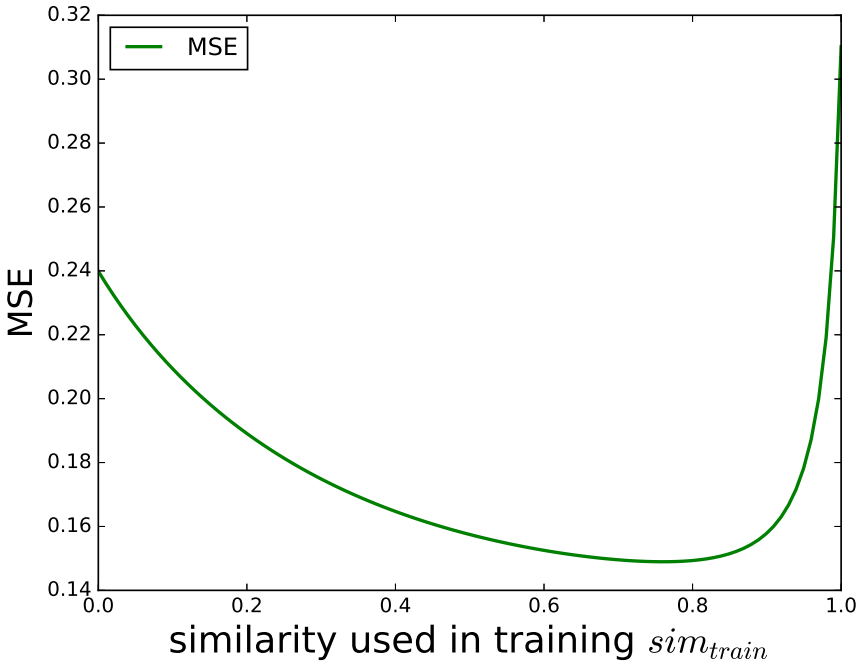


FIGURE 3.3. Similarity v.s MSE in 2-task regression.

sampled from $[0,1] \times [0,1] \subset \mathbb{R}^2$, and the context for each arm $x_{a_t,t}^{(m)}$ is a projection of u_t, and the projection angle depends on the arm and task. Specifically, we use $u_t[0]$ and $u_t[1]$ to denote vector u_t's first and second dimension. For task m, arm $a_t \in \mathcal{A} = \{1,2,3,4,5\}$, the corresponding $x_{a_t,t}^{(m)} = [u_t[0]cos(\frac{\pi}{2}(\frac{a_t}{5} + \frac{m}{10})), u_t[1]sin(\frac{\pi}{2}\frac{a_t}{5})]$ and the reward is $r_{a,t}^{(m)} = 1 - (u_t[0] - \frac{a_t}{5} + 0.3 - \frac{m}{10})^2$. We conduct the experiment in multi-task learning in parallel manner (same as Problem 1). The simulation result is shown in Fig.3.4. We compare the cumulative regret of Algorithm 1 with the performance of conducting Kernel-UCB [**49**] on each task independently.

Here, the cumulative regrets shown in Fig.3.4 are the sum of the cumulative regrets of the five tasks. Further, each data point is the average result of 10 individual simulations. It shows that the regret of multi-learning grows slower than the one of the Kernel-UCB. After 1000 time slots, the multi-task learning (Algorithm 1) using similarity base on CKE, the average R^2 and the hyper parameter method respectively decrease 70.8%, 64.2% and 36.7% of the regret compared to Kernel-UCB. We also test the sequential case (Algorithm 2) in this setting, the performances are similar.

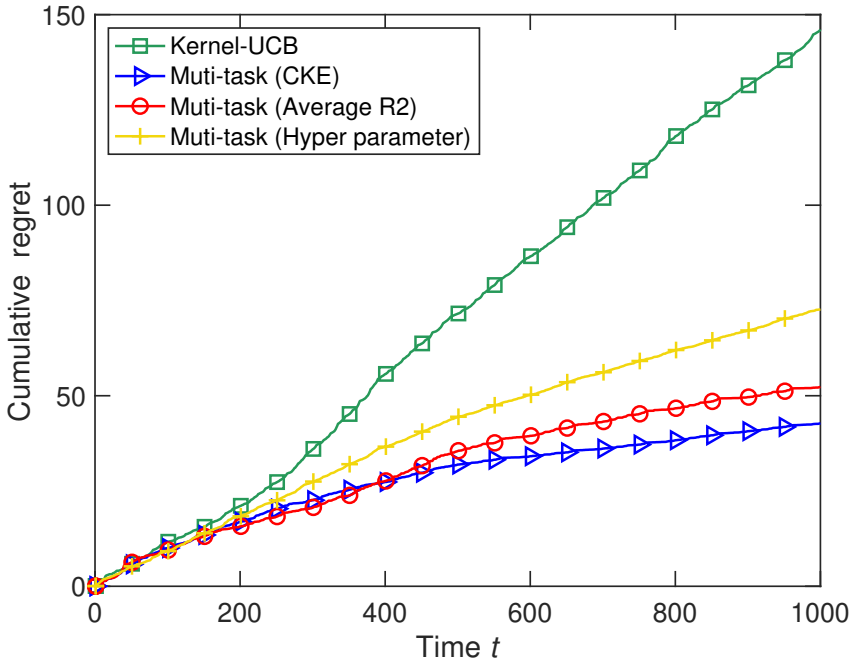


FIGURE 3.4. Multi-task learning in synthetic data.

TABLE 3.1. Sample Data.

BS ID	# Active users	% CQI	%Small packet SDUs	%Small packet volume	# Users	Threshold handover	%Users through-put ≥5Mbps
3714	0.083643988	0.342990	61.37669801	47.70435832	5.20244	-93	90.78014184
3714	0.163259998	0.606118	35.45774141	29.14181596	7.89750	-94	82.55813953
1217	1.471931100	0.242817	30.86999337	31.98075091	85.12305	-98	84.06884082
1217	1.479040265	0.437417	29.61262810	21.28883741	100.42472	-101	62.58613608

3.6.2. Real data evaluation. We start with the data collection and simulator construction procedure, and then discuss about the numerical results.

3.6.2.1. **Data Collection and Simulator Construction.** We build a network simulator based on data collected in real networks to provide interactive environment for bandit algorithms.

The data is collected in the real base station configuration experiments conducted by a service provider in a metropolitan city. We employ 105 BSs within the region to collect 56580 data samples, each for the statistics of a BS observed from 2pm to 10pm in 5 days. An example is illustrated in Table 3.1. These statistics include network measurements, and configured parameter. The network

measurements include user number, CQI, average packet size, etc, as illustrated from Column 2 to 6, used as **states** in our experiment. The configured parameter is handover threshold, as shown Column 7, employed as **actions**. To be specific, handover is a procedure for a BS to guarantee the user experience in cellular network. If one BS observes the signal strength of a user it serves is lower than the threshold, it will handover the user to another BS that has a better communication quality. The range of the configured parameter values is from -112 dBm to -84 dBm, with 1 dBm resolution. Each base station change its configured parameter randomly several times per day. The **reward** is the ratio of users with throughput no less than 5 Mbps, as shown in Column 8.

With the data, we build our simulator. The input is the state and configured parameter (s, c_a), and the output is the corresponding reward r. In detail, when the simulator receives the input (s, c_a), it returns the average of the rewards of the top k nearest neighbors of (s, c_a) in the data set, by Euclidean distance.

3.6.2.2. **Evaluation Setup and Results.** In this experiment, the dimension of the state space is 5. The action space is from -112 dBm to -84 dBm with 1 dBm resolution, that is, the number of arms in our model is 29. The reward space is $[0, 1]$. We test 3 methods to measure the similarity of 105 different BSs. In Fig. 3.5, each subplot corresponds to the similarity matrix K_Z trained by methods in Sec. 3.4.4, the CKE, the average R^2 method and the hyper parameter method. The value in Row i, Column j corresponds to the similarity between BS i and BS j.

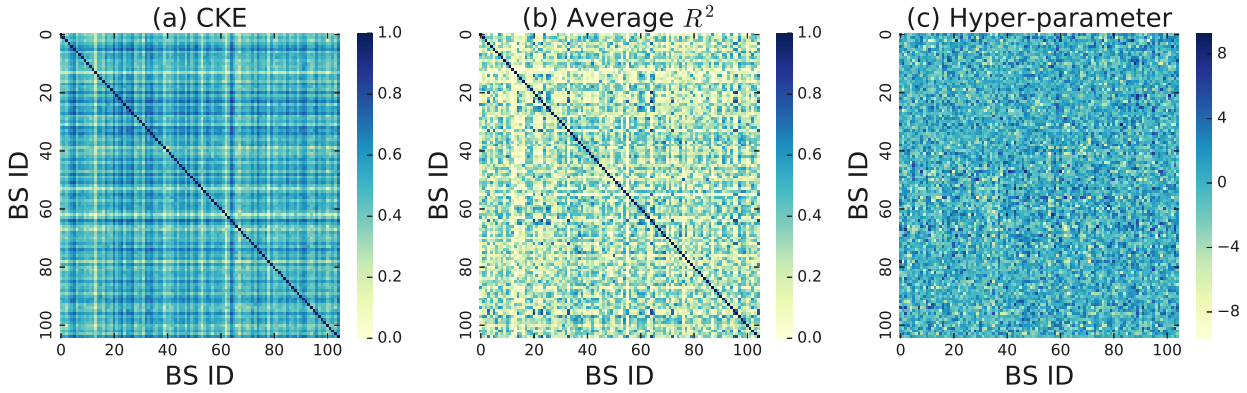
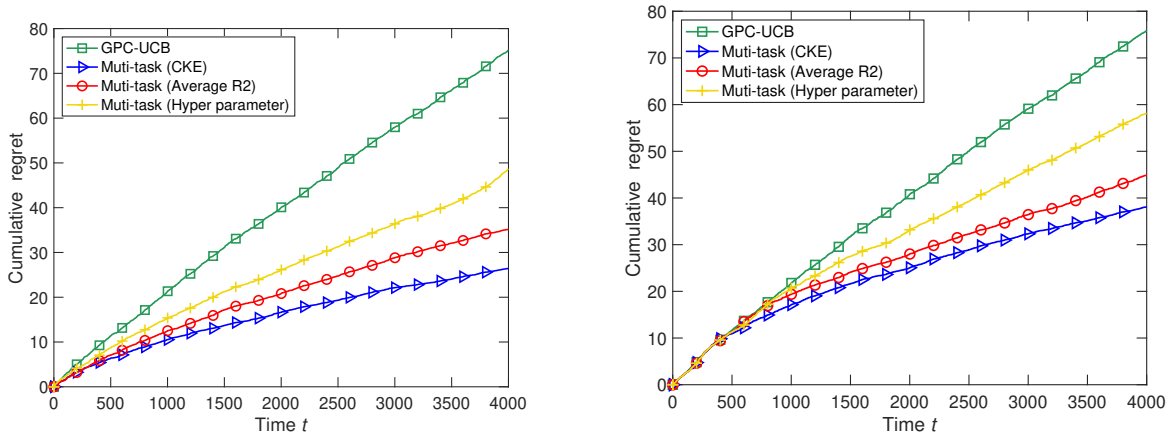


FIGURE 3.5. Similarity matrix among 105 BSs.

We test the multi-task learning case for all 105 BSs in the sequential and the parallel cases based on different similarity metrics. In Fig. 3.6 (a), the result for Algorithm 2 using similarity matrix K_Z

in Fig. 3.5 is shown. We compare the cumulative regret of our algorithm with the performance of conducting GPC-UCB [**118**] on each BS independently. To the best of our knowledge, GPC-UCB is the best algorithm acting on clear definitions of states and actions, therefore, we choose it as our baseline.

The cumulative regrets shown in Fig. 3.6 are the sum of the cumulative regrets of the all BSs. Each data point is the average result of 10 individual simulations. It can be seen that, when our algorithm is used, the regret increases much slower than the baseline. In sequential case, after 4000 time slots, our algorithm using similarity base on the CKE, the average R^2 and the hyper parameter method decreases $64,8\%$, 53.2% and 35.3% of the regret compared to the baseline. For the parallel case, in Fig. 3.6 (b) the result for Algorithm 1 using same similarity matrix K_Z is shown. To make a fair comparison, we rescale the time slots of the parallel case such that the size of the training data is the same as the one in the sequential case. In the parallel case, after 4000 time slots, our algorithm using similarity based on the CKE, the average R^2 and the hyper parameter method decreases 49.8%, 40.9% and 23.5% of the regret compared to the baseline. These figures show that the algorithm in the sequential case has better performance than the one in the parallel case. This is because the learning algorithm in sequential case can improve the model with the immediate feedback reward from each BS, while in the parallel case the algorithm only improves the model when all the feedback rewards from all BSs are collected.



(a) Sequential case

(b) Parallel case

FIGURE 3.6. Multi-task learning v.s. Independent learning in real data.

3.7. Conclusion

In this work, in order to address the multi-BS network configuration problem, we propose a kernel-based multi-task contextual bandits algorithm that leverages the similarity among BSs effectively. In the algorithm, we also provide an approach to measure the similarity among tasks based on conditional kernel embedding. Furthermore, we present theoretical bounds for the proposed algorithm in terms of regret and multi-task-learning efficiency. It shows that the bound of regret is tighter if the learning tasks are more similar. We also evaluate the effectiveness of our algorithm on the synthetic data and the real problem based on a simulator built by real traces. Future work includes possible experimental evaluations in real field tests and further studies on the impact of different similarity metrics.

CHAPTER 4

Opportunistic Learning for Contextual Bandits

4.1. Introduction

In sequential decision making problems such as contextual bandits [**16**, **17**, **18**, **19**], there exists an intrinsic trade-off between exploration (of unknown environment) and exploitation (of current knowledge). Existing algorithm design focuses on how to balance such a trade-off appropriately under the implicit assumption that the exploration cost remains the same over time. However, in a variety of application scenarios, the exploration cost is time varying and situation-dependent. Such scenarios present an opportunity to explore more when the exploration cost is relatively low and exploit more when that cost is high, thus adaptively balancing the exploration-exploitation trade-off to reduce the overall regret. Consider the following motivating examples.

Motivating scenario 1: return variation in recommendations. Contextual bandits have been widely used in recommendation systems [**48**]. In such scenarios, the candidate articles/products to be recommended are considered as the arms, the features of users as the context, and the click-through rate as the reward (i.e., the probability that a user accepts the recommendation). However, note that the monetary return of a recommendation (if accepted) can differ depending on 1) timing (e.g., holiday vs. non-holiday season) and 2) users with different levels of purchasing power or loyalty (e.g., diamond vs. silver status). Because the ultimate goal is to maximize the overall monetary reward, intuitively, when the monetary return of a recommendation (if accepted) is low, the monetary regret of pulling a suboptimal arm is low, leading to a low exploration cost, and correspondingly, high returns lead to high regret and high exploration cost.

Motivating scenario 2: load variation for network configuration. In computer networks, there are a number of parameters that can be configured and have a large impact on overall network performance. For example, in cellular networks, a cell tower can configure transmission power, radio spectrum, antenna, etc., that can affect network performance such as coverage, throughput,

and quality of service. Contextual bandit can be applied in network configuration [**15**]. In such problems, the goal of network configuration can be improving network performance for peak load scenario. In such a scenario, a possible configuration of a cellular base station can be considered as an arm, the characteristics of the cell station such as coverage area as the context, and network performance such as throughput as reward. However, network traffic load fluctuates over time, and thus the actual regret of using a suboptimal configuration varies accordingly.

Specifically, when the network load is low, dummy traffic can be injected into the network so that the total load (real plus dummy load) is the same as the peak load. In this manner, we can seek the optimal configuration under the peak load even in off-peak hours. Meanwhile, the regret of using a suboptimal configuration is low since the real load affected is low. In practice, the priority of the dummy traffic can be set to be lower than that of the real traffic. Because the network handles high priority traffic first, low priority traffic has little or no impact on the high priority traffic [**119**]. Thus, the regret on the actual load can be further reduced, leading to a low or even negligible exploration cost.

Opportunistic Contextual Bandits. Motivated by these application scenarios, we study opportunistic contextual bandits in this paper, focusing on the contextual bandit setting with linear payoffs. Specifically, we define *opportunistic contextual bandit* as a contextual bandit problem with the following characteristic: 1) The exploration cost (regret) of selecting a suboptimal arm varies depending on a time-varying external factor that we called the variation factor. 2) The variation factor is revealed first so that the learning agent can decide which arm to pull depending on this variation factor. As suggested by its name, in opportunistic contextual bandits, the variation of this external variation factor can be leveraged to reduce the actual regret. Further, besides the previous two examples, opportunistic contextual bandit algorithms can be applied to other scenarios that share these characteristics.

We also note that this can be considered as a special case of contextual bandits, by regarding the variation factor as part of context. However, the general contextual bandit algorithms do not take advantage of the opportunistic nature of the problem, and can lead to a less competitive performance.

Contributions. In this paper, we propose an Adaptive Upper-Confidence-Bound algorithm for opportunistic contextual bandits with Linear payoffs (AdaLinUCB). The algorithm is designed to dynamically balance the exploration-exploitation trade-off in opportunistic contextual bandits. To be best of our knowledge, this is the first work to study opportunistic learning for contextual bandits. We focus on the problem-dependent bound analysis here, which is a setting that allows a better bound to be achieved under stronger assumptions. To the best of our knowledge, such a bound does not exist for LinUCB in the existing literature. In this paper, we prove problem-dependent bounds for both the proposed AdaLinUCB and the traditional LinUCB algorithms. Both algorithms have a regret upper bound of $O\left((\log T)^2\right)$, and the coefficient of the AdaLinUCB bound is smaller than that of LinUCB. Furthermore, using both synthetic and real-world large-scale dataset, we show that AdaLinUCB significantly outperforms other contextual bandit algorithms, under large exploration cost fluctuations.

4.2. Related Work

Contextual bandit algorithms have been applied to many real applications, such as display advertising [**120**] and content recommendation [**48**, **74**]. In contrast to the classic K-arm bandit problem [**46**, **47**], side information called context is provided in contextual bandit problem before arm selection [**16**, **17**, **18**, **19**]. The contextual bandits with linear payoffs was first introduced in [**16**]. In [**48**], LinUCB algorithm is introduced based on the "optimism in the face of Uncertainty" principal for linear bandits. The LinUCB algorithm and its variances are reported to be effective in real application scenarios [**48**,**50**,**51**,**52**]. Compared to the classic K-armed bandits, the contextual bandits achieves superior performance in various application scenarios [**121**].

Although LinUCB is effective and widely applied, its analysis is challenging. In the initial analysis effort [**17**], instead of analyzing LinUCB, it presents an $O(\sqrt{T\ln^3(T)})$ regret bound for a modified version of LinUCB. The modification is needed to satisfy the independent requirement by applying Azuma/Hoeffding inequality. In another line of analysis effort, the authors in [**18**] design another algorithm for contextual bandits with linear payoffs and provide its regret analysis without independent requirement. Although the algorithm proposed in [**18**] is different from LinUCB and suffers from a higher computational complexity, the analysis techniques are helpful.

The opportunistic learning has been introduced in [**73**] for classic K-armed bandits. However, we note that opportunistic learning exists for any sequential decision making problem. In [**74**], the authors study into contextual bandits with HLCS (High-Level Critical Situations) set, and proposes a contextual-ϵ-greedy policy, a policy that has an opportunistic nature since the ϵ (exploration level) is adaptively adjusted based on the similarity to HLCSs (importance level). However, it only introduces a heuristic algorithm, and does not present a clearly formulation of opportunistic learning. Furthermore, the policy design in [**74**] implicitly makes the assumption that the contexts in HLCS have already been explored sufficiently beforehand, which is not a cold-start problem. To the best of our knowledge, no prior work has made formal mathematical formulation and rigorous performance analysis for opportunistic contextual bandits.

The opportunistic linear contextual bandits can be regarded as a special case of non-linear contextual bandits. However, general contextual bandit algorithms such as KernelUCB [**49**] do not take advantage of the opportunistic nature of the problem, and thus can lead to a less competitive performance, as shown in Appendix 4.8.5.3 for more details. Moreover, KernelUCB suffers from the sensitivity to hyper-parameter tuning, and the extremely high computational complexity for even moderately large dataset, which limits its application in real problems.

4.3. System Model

We use the following notation conventions. We use $\|x\|_2$ to denote the 2-norm of a vector $x \in \mathbb{R}^d$. For a positive-definite matrix $A \in \mathbb{R}^{d\times d}$, the weighted 2-norm of vector $x \in \mathbb{R}^d$ is defined by $\|x\|_A = \sqrt{x^\top Ax}$. The inner product of vectors is denoted by $\langle \cdot, \cdot \rangle$, that is, $\langle x, y\rangle = x^\top y$. Denote by $\lambda_{\min}(A)$ the minimum eigenvalue of a positive-definite matrix A. Denote by $\det(A)$ the determinant of matrix A. Denote by $\mathrm{trace}(A)$ the trace of matrix A.

Now, we present system model. We first introduce the setting of a standard linear contextual bandit problem. The time is slotted. In each time slot t, there exists a set of possible arms, denoted by set $\mathcal{D}_t$. For each arm $a \in \mathcal{D}_t$, there is an associated context vector $x_{t,a} \in \mathbb{R}^d$, and a nominal reward $r_{t,a}$. In each slot t, the learner can observe context vectors of all possible arms, and then choose an arm a_t and receive the corresponding nominal reward r_{t,a_t}. Note that only the nominal reward of the chosen arm is revealed for the learner in each time slot t. Further, the nominal

rewards of arms are assumed to be a noisy version of an unknown linear function of the context vectors. Specifically, $r_{t,a} = \langle x_{t,a}, \theta_\star \rangle + \eta_t$, where $\theta_\star \in \mathbb{R}^d$ is an unknown parameter, and η_t is a random noise with zero mean, i.e., $\mathbb{E}[\eta_t | x_{t,a_t}, \mathcal{H}_{t-1}] = 0$, with $\mathcal{H}_{t-1} = (x_{1,a_1}, \eta_1, \cdots, x_{t-1,a_{t-1}}, \eta_{t-1})$ representing historical observations.

The goal of a standard contextual bandit problem is to minimize the total regret in T slots, in terms of the nominal rewards. Particularly, the accumulated T-slot regret regarding nominal reward is defined as,

$$\mathbf{R}_{\text{total}}(T) = \sum_{t=1}^{T} R_t = \sum_{t=1}^{T} \mathbb{E}[r_{t,a_t^\star} - r_{t,a_t}], \tag{4.1}$$

where R_t is the one-slot regret regarding nominal reward for time slot t, $a_t^\star$ is the optimal arm at time slot t. Here, the optimal arm is the one with the largest expected reward, i.e., $a_t^\star = \arg\max_{a \in \mathcal{D}_t} \mathbb{E}[r_{t,a}]$. To simplify the notation, we denote $r_{t,\star} = r_{t,a_t^\star}$ in the following. That is, $r_{t,\star}$ is the optimal nominal reward at slot t.

In the **opportunistic learning environment**, let L_t be an external variation factor for time slot t. The **actual reward** $\tilde{r}_{t,a}$ that the agent receives has the following relationship with the **nominal reward**:

$$\tilde{r}_{t,a} = L_t r_{t,a}, \forall t, \forall a \in \mathcal{D}_t.$$

At each time slot, the learner first observes the context vectors associated with all possible arms, i.e., $x_{t,a}, \forall a \in \mathcal{D}_t$, as well as the current value of L_t. Based on which the learner selects current arm a_t, observes a nominal reward r_{t,a_t}, and receives the actual reward $\tilde{r}_{t,a} = L_t r_{t,a}$.

This model captures the essence of the opportunistic contextual bandits. For example, in the recommendation scenario, and L_t can be a seasonality factor, which captures the general purchase rate in current season. Or L_t can be purchasing power (based on historical information) or loyalty level of users (e.g., diamond vs. silver status). In the network configuration example, when the nominal reward $r_{t,a}$ captures the impact of a configuration at the peak load, the total load (the dummy load plus the real load) resembles the peak load. Then, L_t can be the amount of real load, and thus the actual reward is modulated by L_t as $L_t r_{t,a}$.

Algorithm 5 AdaLinUCB

Inputs: $\alpha \in \mathbb{R}_+$, $d \in \mathbb{N}$, $l^{(+)}$, $l^{(-)}$.
$A \leftarrow \boldsymbol{I}_d$ {The d-by-d identity matrix}
$b \leftarrow \mathbf{0}_d$
for $t = 1, 2, 3, \cdots, T$ **do**
  $\theta_{t-1} = A^{-1}b$
  Observe possible arm set $\mathcal{D}_t$, and observe associated context vectors $x_{t,a}, \forall a \in \mathcal{D}_t$.
  Observe L_t and calculate $\tilde{L}_t$ by (4.3).
  for $a \in \mathcal{D}_t$ **do**
    $a_t = \arg\max_{a \in \mathcal{D}_t} \theta_{t-1}^\top x_{t,a} + \alpha\sqrt{(1-\tilde{L}_t)x_{t,a}^\top A^{-1} x_{t,a}}$
  end for
  Choose action $a_t = \arg\max_{a \in \mathcal{D}_t} p_{t,a}$ with ties broken arbitrarily.
  Observe nominal reward r_{t,a_t}.
  $A \leftarrow A + x_{t,a_t} x_{t,a_t}^\top$
  $b \leftarrow b + x_{t,a_t} r_{t,a_t}$
end for

The goal of the learner is to minimize the total regret in T slots, in terms of the actual rewards. Particularly, the accumulated T-slot regret regarding actual reward is defined as,

$$\tilde{\mathbf{R}}_{\text{total}}(T) = \sum_{t=1}^{T} \mathbb{E}[R_t L_t] = \sum_{t=1}^{T} \mathbb{E}[L_t r_{t,\star} - L_t r_{t,a_t}]. \tag{4.2}$$

In a special case, equation (4.2) has an equivalent form: when L_t is i.i.d. over time with mean value $\bar{L}$ and r_{t,a_t} is independent of L_t conditioned on a_t, the total regret regarding actual reward is $\tilde{\mathbf{R}}_{\text{total}}(T) = \bar{L}\sum_{t=1}^{T}\mathbb{E}[r_{t,\star}] - \sum_{t=1}^{T}\mathbb{E}[L_t r_{t,a_t}]$. Note that in general, it is likely that $\mathbb{E}[L_t r_{t,a_t}] \neq \bar{L}\mathbb{E}[r_{t,a_t}]$, because the action a_t can depend on L_t.

4.4. Adaptive LinUCB

We note that the conventional LinUCB algorithm assumes that the exploration cost factor does not change over time, i.e., $L_t = 1$. Therefore, to minimize the the nominal reward is equivalent to that of the actual reward. When L_t is time-varying and situation dependent as discussed earlier, we need to maximize the total actual reward, which is affected by the variation factor L_t. Motivated by this distinction, we design the adaptive LinUCB algorithm (AdaLinUCB) as in Algo. 5.

$$a_t = \arg\max_{a \in \mathcal{D}_t} \theta_{t-1}^\top x_{t,a} + \alpha\sqrt{(1-\tilde{L}_t)x_{t,a}^\top A^{-1} x_{t,a}}$$

In Algo. 5, α is a hyper-parameter, which is an input of the algorithm, and $\tilde{L}_t$ is the normalized variation factor, defined as,

$$\tilde{L}_t = \left([L_t]_{l^{(-)}}^{l^{(+)}} - l^{(-)} \right) / \left(l^{(+)} - l^{(-)} \right), \tag{4.3}$$

where $l^{(-)}$ and $l^{(+)}$ are the lower and upper thresholds for truncating the variation factor, and $[L_t]_{l^{(-)}}^{l^{(+)}} = \max\{l^{(-)}, \min\{L_t, l^{(+)}\}\}$. That is, $\tilde{L}_t$ normalizes L_t into $[0, 1]$ to capture different ranges of L_t. To achieve good performance, the truncation thresholds should be appropriately chosen to achieve sufficient exploration. Empirical results show that a wide range of threshold values can lead to good performance of AdaLinUCB. Furthermore, these thresholds can be learned online in practice without prior knowledge on the distribution of L_t, as discussed in Sec. 4.6 and Appendix 4.8.5. Note that $\tilde{L}_t$ is only used in AdaLinUCB algorithm. The actual rewards and regrets are based on L_t, not $\tilde{L}_t$.

In Algo. 5, for each time slot, the algorithm updates a matrix A and a vector b. The A is updated in step 13, which is denoted as $A_t = I_d + \sum_{\tau=1}^{t} x_{\tau,a_\tau} x_{\tau,a_\tau}^\top$ in the following analysis. Note that A_t is a positive-definite matrix for any t, and that $A_0 = I_d$. The b is updated in step 14, which is denoted as $b_t = \sum_{\tau=1}^{t} x_{\tau,a_\tau} r_{\tau,a_\tau}$ in the following analysis. Then, we have $\theta_t = A_t^{-1} b_t$ (see step 5), which is the estimation of the unknown parameter $\theta_\star$ based on historical observations. Specifically, θ_t is the result of a ridge regression for estimating $\theta_\star$, which minimizes a penalized residual sum of squares, i.e., $\theta_t = \arg\min_\theta \left\{ \sum_{\tau=1}^{t} (r_{\tau,a_\tau} - \langle \theta, x_{\tau,a_\tau} \rangle)^2 + \|\theta\|_2^2 \right\}$.

In general, the AdaLinUCB algorithm explores more when the variation factor is relatively low, and exploits more when the variation factor is relatively high. To see this, note that the first term of the index $p_{t,a}$ in step 9, i.e., $\theta_{t-1}^\top x_{t,a}$, is the estimation of the corresponding reward; while the second part is an adaptive upper confidence bound modulated by $\tilde{L}_t$, which determines the level of exploration. At one example, when L_t is at its lowest level with $L_t \leq l^{(-)}$, $\tilde{L}_t = 0$, and the index $p_{t,a}$ is the same as that of the LinUCB algorithm, and then the algorithm selects arm in the same way as the conventional LinUCB. At the other extreme, when $\tilde{L}_t = 1$, i.e., $L_t \geq l^{(+)}$, the index $p_{t,a} = \theta_{t-1}^\top x_{t,a}$, which is the estimation of the corresponding reward. That is, when the variation factor is at its highest level, the AdaLinUCB algorithm purely exploits the existing knowledge and selects the current best arm.

4.5. Performance Analysis

We first summarize the technical assumptions needed for performance analysis: i. Noise satisfies C_{noise}-sub-Gaussian condition, as explained later in (4.4); ii. The unknown parameter $\theta_\star$ satisfies $||\theta_\star||_2 \leq C_{\text{theta}}$; iii. For $\forall t, \forall a \in \mathcal{D}_t$, $\|x_{t,a}\|_2 \leq C_{\text{context}}$ holds; iv. $\lambda_{\min}(I_d) \geq \max\{1, C^2_{\text{context}}\}$; v. the nominal reward r_{t,a_t} is independent of the variation factor L_t, conditioned on a_t.

We note that assumptions i.-iv. are widely used in contextual bandit analysis [**16**,**17**,**18**,**50**,**51**].

Specifically, the sub-Gaussian condition in assumption i. is a constraint on the tail property of the noise distribution, as that in [**18**]. That is, for the noise η_t, we assume that,

$$\forall \zeta \in \mathbb{R}, \quad \mathbb{E}[e^{\zeta \eta_t} | x_{t,a_t}, \mathcal{H}_{t-1}] \leq \exp\left(\frac{\zeta^2 C^2_{\text{noise}}}{2}\right), \tag{4.4}$$

with $\mathcal{H}_{t-1} = (x_{1,a_1}, \eta_1, \cdots, x_{t-1,a_{t-1}}, \eta_{t-1})$ and $C_{\text{noise}} > 0$. Note that the sub-Gaussian condition requires both (4.4) and $\mathbb{E}[\eta_t | x_{t,a_t}, \mathcal{H}_{t-1}] = 0$. Further, this condition indicates that $\text{Var}[\eta_t | F_{t-1}] \leq C^2_{\text{noise}}$, where $\{F_t\}_{t=0}^{\infty}$ is the filtration of σ-algebras for selected context vectors and noises, i.e., $F_t = \sigma(x_{1,a_1}, x_{2,a_2}, \cdots, x_{t+1,a_{t+1}}, \eta_1, \eta_2, \cdots, \eta_t)$. Thus, C^2_{noise} can be viewed as the (conditional) variance of the noise.

Examples for the distributions that satisfies the sub-Gaussian condition are: 1) A zero-mean Gaussian noise with variance at most C^2_{noise}; 2) A bounded noise with zero-mean and lying in an interval of length at most $2C_{\text{noise}}$.

Assumption iv. can be relaxed by changing the value of A_0 in Algo. 5 from the current identity matrix I_d to a positive-definite matrix with a higher minimum eigenvalue (see Appendix 4.8.1 for more details).

Assumption v. is valid in many application scenarios. For example, in the network configuration scenario, since the total load resembles the peak load, the network performance, i.e., the nominal reward r_{t,a_t}, is independent of the real load L_t, conditioned on configuration a_t. Also, in the recommendation scenario, the click-through rate (i.e., reward $r_{t,a}$) can be independent of the user influence (i.e., variation factor L_t).

4.5.1. Problem-Dependent Bounds. We focus on problem-dependent performance analysis here because it can lead to a tighter bound albeit under stronger assumptions. To derive the

problem-dependent bound, we assume that there are a finite number of possible context values, and denote this number as N. Then, let $\Delta_{\min}$ denote the minimum nominal reward difference between the best and the "second best" arms. That is, $\Delta_{\min} = \min_t \left\{ r_{t,\star} - \max_{a \in \mathcal{D}_t, r_{t,a} \neq r_{t,\star}} r_{t,a} \right\}$. Similarly, let $\Delta_{\max}$ denote the maximum nominal reward difference between arms. That is, $\Delta_{\max} = \max_t \{ r_{t,\star} - \min_{a \in \mathcal{D}_t} r_{t,a} \}$.

As in existing literature for problem-dependent analysis of linear bandits [**18**], we assume that single optimal context condition holds here. Specifically, for different time slot $t = 1, 2, \cdots$, there is a single optimal context value. That is, there exists $x_\star \in \mathbb{R}^d$, such that, $x_\star = x_{t,a_t^\star}, \forall t$.

4.5.2. AdaLinUCB under Binary-Valued Variation. We first introduce the result under a random binary-valued variation factor. We assume that the variation factor L_t is i.i.d. over time, with $L_t \in \{\epsilon_0, 1 - \epsilon_1\}$, where $\epsilon_0, \epsilon_1 \geq 0$ and $\epsilon_0 < 1 - \epsilon_1$. Let ρ denote the probability that the variation factor is low, i.e., $\mathbb{P}\{L_t = \epsilon_0\} = \rho$.

Firstly, we note that, for a $\tilde{\delta} \in (0, 1)$, there exists a positive integer C_{slots} such that,

$$\begin{aligned} \forall t \geq C_{\text{slots}},\ \rho t - \sqrt{\frac{t}{2} \log \frac{\tilde{\delta}}{2}} - \frac{16 C_{\text{noise}}^2 C_{\text{theta}}^2}{\Delta_{\min}^2} \Bigg[& \log(C_{\text{context}} t) \\ & + 2(d-1) \log \left(d \log \frac{d + t C_{\text{context}}^2}{d} + 2 \log \frac{2}{\tilde{\delta}} \right) + 2 \log \frac{2}{\tilde{\delta}} \\ & + (d-1) \log \frac{64 C_{\text{noise}}^2 C_{\text{theta}}^2 C_{\text{context}}}{\Delta_{\min}^2} \Bigg]^2 \geq \frac{4d}{\Delta_{\min}^2}. \end{aligned} \tag{4.5}$$

To see such an integer C_{slots} exists, note that for large enough t, in the left-hand side of the inequality (4.5), the dominant positive term is $O(t)$ while the dominant negative term is $O(\sqrt{t})$.

To interpret C_{slots}, it is an integer that is large enough so that during C_{slots}-slot period, enough exploration is done in the time slots when variation factor is relatively low, such that to have a relatively tight bound for the estimation of the optimal reward.

Then, we have the following results.

THEOREM 4.5.1. *Consider the opportunistic contextual bandits with linear payoffs and binary-valued variation factor. With probability at least* $1 - \tilde{\delta}$*, the accumulated regret (regarding actual*

reward) of AdaLinUCB algorithm satisfies,

$$\begin{aligned}\tilde{\mathbf{R}}_{\text{total}}(T) \leq \epsilon_0 \cdot \frac{16C_{\text{noise}}^2 C_{\text{theta}}^2}{\Delta_{\min}} \Bigg[& \log(C_{\text{context}} T) + 2\log\frac{2}{\tilde{\delta}} \\ & + 2(d-1)\log\Big(d\log\frac{d+TC_{\text{context}}^2}{d} + 2\log\frac{2}{\tilde{\delta}}\Big) \\ & + (d-1)\log\frac{64C_{\text{noise}}^2 C_{\text{theta}}^2 C_{\text{context}}}{\Delta_{\min}^2}\Bigg]^2 \\ & + (1-\epsilon_1)\Big[\big(\Delta_{\max} C_{\text{slots}} + 4d\frac{N-1}{\Delta_{\min}}\big) \\ & \cdot \Big(C_{\text{noise}}\sqrt{d\log\frac{2+2TC_{\text{context}}^2}{\tilde{\delta}}} + C_{\text{theta}}\Big)^2\Big]\end{aligned}$$

$$\tilde{\mathbf{R}}_{\text{total}}(T) \leq \frac{\mathbb{E}\big[L_t | L_t \leq l^{(-)}\big] O\left((\log T)^2\right)}{\Delta_{\min}} + \frac{\mathbb{E}\big[L_t | L_t > l^{(-)}\big] O\left(\log T\right)}{\Delta_{\min}}$$

$$\tilde{\mathbf{R}}_{\text{total}}(T) \leq \frac{\epsilon_0 O\left((\log T)^2\right)}{\Delta_{\min}} + \frac{(1-\epsilon_1) O\left(\log T\right)}{\Delta_{\min}}$$

$$\tilde{\mathbf{R}}_{\text{total}}(T) \leq \frac{1-\epsilon_1+\epsilon_0}{2} \frac{O\left((\log T)^2\right)}{\Delta_{\min}}$$

where C_{slots} *is a constant satisfying* (4.5).

Proof Sketch: Although the proof for Theorem 4.5.1 is complicated, the key is to treat the slots with low variation factor and the slots with high variation factor separately. For slots with low variation factor, the one-step regret is upper bounded by the weighted 2-norm of the selected context vectors, i.e., $R_t \mathbb{1}\{L_t = \epsilon_0\} \leq 2\alpha \|x_{t,a_t}\|_{A_{t-1}^{-1}}$, and then the accumulated regret can be analyzed accordingly. For the slots with high variation factor, by matrix analysis, we can show that when a particular context value has been selected enough times, its estimated reward is accurate enough in an appropriate sense. Further, it can benefit from regret bound for low variation factor slots that the optimal context has been selected enough time with high probability. Then, we combine these to prove the result. More details are shown in Appendix 4.8.2.

REMARK 4.5.1. *For the regret bound in Theorem 4.5.1, the first three lines cover the accumulated regret that is incurred during time slots when the variation factor is relatively low, i.e., during slots* t *with* $L_t = \epsilon_0$*, while the last two lines cover the accumulated regret that is incurred during time slots when the variation factor is relatively high, i.e., during slots* t *with* $L_t = 1 - \epsilon_1$*. Further, when* T *is large enough, the dominant term for the first three lines is* $O\left((\log T)^2\right)$*, while the dominant term for the last two lines is* $O(\log T)$*. That is, the bound for the accumulated regret during slots when the variation factor is relatively high actually increases slower than the bound for the accumulated regret during slots when the variation factor is relatively low. This is in consistent with the motivation of AdaLinUCB design: explore more when the variation factor is relatively low, and exploit more when the variation factor is relatively high.*

Furthermore, beside parameter T*, which is the time horizon, the regret bound in Theorem 4.5.1 is also affected by problem-dependent parameters: it is affected by* N*, which is the number of possible context values,* $\Delta_{\min}$*, which is the minimum nominal reward difference between the best and the "second best" arms, and* $\Delta_{\max}$*, which is the maximum nominal reward difference between arms. In general, a larger number of possible context values, i.e., a larger* N*, may lead to a larger* $\Delta_{\max}$ *and a smaller* $\Delta_{\min}$*, and in this way, results in a larger regret bound.*

4.5.3. AdaLinUCB under Continuous Variation. We now study AdaLinUCB in opportunistic contextual bandits under continuous variation factor. Under continuous variation factor, it is difficult to obtain regret bound for general values of $l^{(-)}$ and $l^{(+)}$ because exploration and exploitation mix in a complex fashion when $l^{(-)} < L_t < l^{(+)}$. Instead, inspired by the insights obtained from the binary-valued variation factor case, we illustrate the advantages of AdaLinUCB for special case with $l^{(-)} = l^{(+)}$.

In the special case of $l^{(-)} = l^{(+)}$, the normalized variation factor $\tilde{L}_t$ in (4.3) is redefined as $\tilde{L}_t = 0$ when $L_t \leq l^{(-)}$ and as $\tilde{L}_t = 1$ when $L_t > l^{(+)} = l^{(-)}$.

THEOREM 4.5.2. *In the opportunistic contextual bandits with linear payoffs and continuous variation factor that is i.i.d. over time, under AdaLinUCB with* $\mathbb{P}\{L_t \leq l^{(-)}\} = \rho > 0$ *and*

$l^{(-)} = l^{(+)}$, *with probability at least* $1-\tilde{\delta}$, *the accumulated regret (regarding actual reward) satisfies,*

$$\begin{aligned}\tilde{\mathbf{R}}_{\text{total}}(T) \leq & \mathbb{E}\big[L_t | L_t \leq l^{(-)}\big] \frac{16 C_{\text{noise}}^2 C_{\text{theta}}^2}{\Delta_{\min}} \Bigg[\log(C_{\text{context}} T) \\ & + 2(d-1) \log\left(d \log \frac{d + T C_{\text{context}}^2}{d} + 2 \log \frac{2}{\tilde{\delta}} \right) \\ & + (d-1) \log \frac{64 C_{\text{noise}}^2 C_{\text{theta}}^2 C_{\text{context}}}{\Delta_{\min}^2} + 2 \log \frac{2}{\tilde{\delta}} \Bigg]^2 \\ & + \mathbb{E}[L_t | L_t > l^{(-)}] \cdot \Bigg[\left(\Delta_{\max} C_{\text{slots}} + 4d \frac{N-1}{\Delta_{\min}} \right) \\ & \cdot \left(C_{\text{noise}} \sqrt{d \log \frac{2 + 2T C_{\text{context}}^2}{\tilde{\delta}}} + C_{\text{theta}} \right)^2 \Bigg],\end{aligned}$$

where C_{slots} *is a constant satisfying* (4.5).

PROOF. Recall that for the special case with $l^{(+)} = l^{(-)}$, we have $\tilde{L}_t = 0$ when $L_t \leq l^{(-)}$ and as $\tilde{L}_t = 1$ when $L_t > l^{(+)}$. Thus, this theorem can be proved analogically to the proof of Theorem 4.5.1, by noting the following: When $L_t \leq l^{(-)}$, we have $\tilde{L}_t = 0$ which corresponds to the case of $L_t = \epsilon_0$ ($\tilde{L}_t = 0$) in the binary-valued variation factor case; while when $L_t > l^{(+)}$ ($\tilde{L}_t = 1$) corresponds to the case of $L_t = 1 - \epsilon_1$ under binary-valued variation factor case. The conclusion of the theorem then follows by using the fact that all variation factor below $l^{(-)}$ are treated same by AdaLinUCB, i.e., $\tilde{L}_t = 0$ for $L_t \leq l^{(-)}$; while all variation factor above $l^{(-)}$ are treated same by AdaLinUCB, i.e., $\tilde{L}_t = 1$ for $L_t \leq l^{(+)}$. □

REMARK 4.5.2. *Similar to Remark 4.5.1 for Theorem 4.5.1, the regret bound in Theorem 4.5.2 can be divided into two parts: the first three lines cover the accumulated regret that is incurred during time slots when* $L_t \leq l^{(-)}$ *and is* $O\left((\log T)^2\right)$, *while the last two lines cover the accumulated regret for time slots when* $L_t > l^{(-)}$ *and is* $O\left((\log T)\right)$. *Furthermore, a larger* N, *i.e., a larger number of possible context values, can lead to a larger regret bound.*

4.5.4. Regret Bound of LinUCB. To the best of our knowledge, there exists no problem-dependent bound on LinUCB. (The initial analysis of LinUCB presents a more general and looser performance bound for a modified version of LinUCB. The modification is needed to satisfy the independent requirement by applying Azuma/Hoeffding inequality [**17**].) Furthermore, we note

that one can directly apply LinUCB to opportunistic contextual bandits using the linear relationship $\mathbb{E}[r_{t,a}|x_{t,a}] = \langle x_{t,a}, \theta_\star \rangle$, which is called LinUCBExtracted in numerical results. Therefore, we derive the regret upper bound for LinUCB here, both as an individual contribution as well as for comparison purpose.

THEOREM 4.5.3. *In the opportunistic contextual bandits with linear payoffs and continuous variation factor that is i.i.d. over time with mean $\bar{L}$, with probability at least $1-\delta$, the accumulated T-slot regret (regarding actual reward) of LinUCB satisfies,*

$$\begin{aligned}\tilde{\mathbf{R}}_{\text{total}}(T) \leq \frac{16\bar{L}C_{\text{noise}}^2 C_{\text{theta}}^2}{\Delta_{\min}} \Bigg[& \log(C_{\text{context}} T) + 2\log\frac{1}{\delta} \\ & + 2(d-1)\log\left(d\log\frac{d+TC_{\text{context}}^2}{d} + 2\log\frac{1}{\delta}\right) \\ & + (d-1)\log\frac{64C_{\text{noise}}^2 C_{\text{theta}}^2 C_{\text{context}}}{\Delta_{\min}^2}\Bigg]^2.\end{aligned}$$

The regret bound for LinUCB under non-opportunistic case can be shown by simply having $\bar{L} = 1$ in the above result. Here, note that problem-dependent bound analysis is a setting that allows a better bound to be achieved with stronger assumptions. Recall that the assumptions are discussed in Sec. 4.5.1. As a result, the problem-dependent bound of LinUCB is much better than its general bound, such as the bound for a modified version of LinUCB in [**17**]. More results for LinUCB and the proof of Theorem 4.5.3 can be found in Appendix 4.8.3.

REMARK 4.5.3. *Theorem 4.5.3 and Theorem 4.5.1 show that the problem-dependent regret bounds (regarding actual reward) for LinUCB and AdaLinUCB are both $O\left((\log T)^2\right)$. Further, for binary-valued variation factor, the asymptotically dominant term for the bound of LinUCB is $\frac{1-\epsilon_1+\epsilon_0}{2} \cdot \frac{16C_{\text{noise}}^2 C_{\text{theta}}^2}{\Delta_{\min}} (\log T)^2$. In comparison, for AdaLinUCB, it is $\epsilon_0 \cdot \frac{16C_{\text{noise}}^2 C_{\text{theta}}^2}{\Delta_{\min}} (\log T)^2$. Because $\epsilon_0 < 1-\epsilon_1$, in the scenario of binary-valued variation factor, the AdaLinUCB algorithm has a better asymptotic problem-dependent upper bound than that of the LinUCB algorithm. Similarly, in scenario with continuous variation factor, the AdaLinUCB algorithm with $l^{(+)} = l^{(-)}$ has a better problem-dependent bound than LinUCB algorithm as long as $\mathbb{E}[L_t | L_t \leq l^{(-)}] < \bar{L}$, which holds in most cases.*

4.5.5. Discussions on the Disjoint Model. The seminal paper on LinUCB [**48**] introduces different models for contextual bandits. The opportunistic learning applies to these different models. One of them is the joint model discussed above. Another model is the disjoint model, which assumes that, $\mathbb{E}[r_{t,a}|x_{t,a}] = \langle x_{t,a}, \theta_\star^{(a)} \rangle$, where $x_{t,a}$ is a context vector and $\theta_\star^{(a)}$ is the unknown coefficient vector for arm a. This model is called disjoint since the parameters are not shared among different arms. There is also a hybrid model that combines the joint model and the disjoint model.

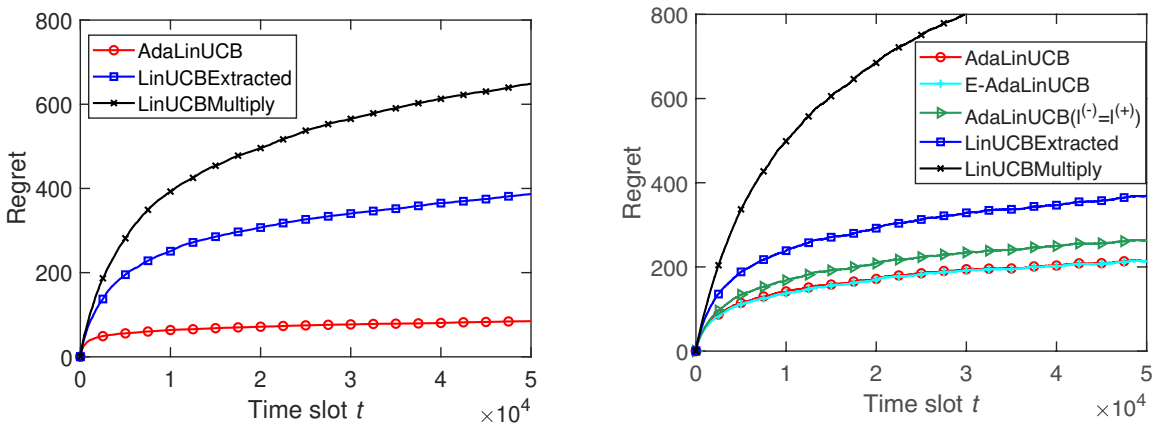
In this paper, we focus on the design and analysis of opportunistic contextual bandits using the joint model. However, it should be noted that, the AdaLinUCB algorithm in Algo. 5 can be modified slightly and applied to the disjoint model, see Appendix 4.8.4 for more details. Also, the analysis of the joint model can be extended to the disjoint one. Note that the disjoint model can be converted to a joint model when the number of possible arms is finite. Specifically, for an arbitrary disjoint-model opportunistic contextual bandit problem with $\theta_\star^{(a)}$ $\forall a$, an equivalent joint-model problem exists with the joint unknown parameter as $\theta_\star = ([\theta_\star^{(1)}]^\top, [\theta_\star^{(2)}]^\top, \cdots)^\top$ and the context vectors modified accordingly. Thus, the previous analytical results are valued for the disjoint model with appropriate modifications.

4.6. Numerical Results

We present numerical results to demonstrate the performance of the AdaLinUCB algorithm using both synthetic scenario and real-world datasets. We have implemented the following algorithms: 1) AdaLinUCB in Algo. 5; 2) LinUCB(Extracted) in Sec. 4.5.4; 3) **LinUCBMultiply**, another way to directly apply LinUCB in opportunistic case, where we use $L_t \cdot x_{t,a}$ as context vector; 4) **E-AdaLinUCB**, an algorithm that adjusts the threshold $l^{(+)}$ and $l^{(-)}$ based on the empirical distribution of L_t. In all the algorithms, we set $\alpha = 1.5$ to make a fair comparison.

We have also experimented **LinUCBCombine** algorithm, where we use $\tilde{x}_{t,a} = [L_t, x_{t,a}^\top]^\top$ as context vector to directlty apply LinUCB, and find that LinUCBCombine has a much worse performance compared to other algorithms.

Meanwhile, we also notice that the opportunistic linear contextual bandits can be regarded as a special case of non-linear contextual bandits by viewing the variation factor L_t as a part of context vector. Along this line of thinking, we have also experimented **KernelUCB** algorithm [**49**],



(a) Binary-valued variation factor

(b) Beta distributed variation factor

FIGURE 4.1. Regret under Synthetic Scenarios. In (a), $\epsilon_0 = \epsilon_1 = 0, \rho = 0.5$. In (b), AdaLinUCB: $l^{(-)}$=$l_0^{(-)}$, $l^{(+)}$= $l_0^{(+)}$; AdaLinUCB: ($l^{(-)}$= $l^{(+)}$), $l^{(-)}$= $l^{(+)}$= $l_{0.5}^{(-)}$

which is a general algorithm for non-linear contextual bandits. However, we find that KernelUCB is less competitive in performance and suffers from extremely high computational complexity (see Appendix 4.8.5.3 for more details). One reason is that a general contextual bandit algorithm such as KernelUCB does not take advantage of the opportunistic nature of the problem, and can, therefore, have a worse performance than AdaLinUCB.

4.6.1. Experiments on Synthetic Scenarios. The synthetic scenario has a total of 20 possible arms, each associated with a disjoint unknown coefficient $\theta_\star^{(a)}$. The simulator generates 5 possible groups of context vectors, and each group has context vectors associated with all the possible arms. At each time slot, a context group is presented before the decision. Further, each unknown coefficient $\theta_\star^{(a)}$ and each context $x_{t,a}$ is a 6-dimension vector, with elements in each dimension generated randomly, and is normalized such that the L2-norm of $\theta_\star^{(a)}$ or $x_{t,a}$ is 1.

Fig. 4.1(a) shows the regret for different algorithms under random binary-value variation factor with $\epsilon_0 = \epsilon_1 = 0$ and $\rho = 0.5$. AdaLinUCB significantly reduces the regret in this scenario. Specifically, at time slots $t = 5 \times 10^4$, AdaLinUCB achieves a regret that is only 10.3% of that of LinUCBMultiply, and 17.6% of that of LinUCBExtracted.

For continuous variation factor, Fig. 4.1(b) compares the regrets for the algorithms under a beta distributed variation factor. Here, we define $l_\rho^{(-)}$ as the lower threshold such that $\mathbb{P}\{L_t \leq l_\rho^{(-)} = \rho\}$, and $l_\rho^{(+)}$ as the higher threshold such that $\mathbb{P}\{L_t \geq l_\rho^{(+)} = \rho\}$. It is shown that AdaLinUCB still

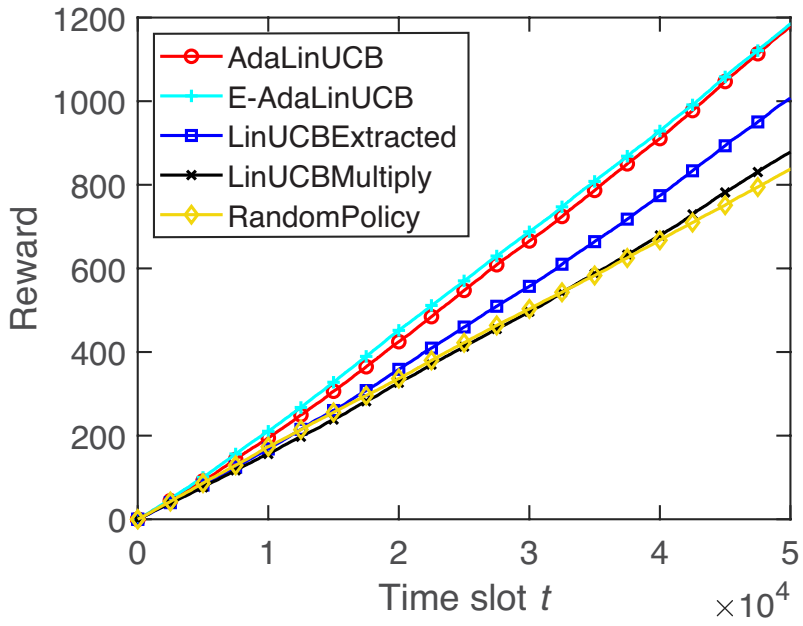


FIGURE 4.2. Rewards for Yahoo! Today Module $l^{(-)} = l_0^{(-)}$, $l^{(+)} = l_{0.3}^{(+)}$

outperforms other algorithms, and AdaLinUCB has a regret 41.8% lower than that of LinUCBExtracted. Furthermore, its empirical version, E-AdaLinUCB has a similar performance to that of AdaLinUCB. Even in the special case with a single threshold $l^{(-)} = l^{(+)} = l_{0.5}^{(-)}$, AdaLinUCB still outperforms LinUCBExtracted, reducing the regret by 28.6%.

We have conducted more simulations to evaluate the impact of environment and algorithm parameters such as variation factor fluctuation and the thresholds for variation factor truncation, and find that AdaLinUCB works well in different scenarios (see Appendix 4.8.5.1 and 4.8.5.2).

4.6.2. Experiments on Yahoo! Today Module. We also test the performance of the algorithms using the data from Yahoo! Today Module. This dataset contains over 4 million user visits to the Today module in a ten-day period in May 2009 [**48**]. To evaluate contextual bandits using offline data, the experiment uses the unbiased offline evaluation protocol proposed in [**120**]. For the variation factor, we use a real trace - the sales of a popular store. It includes everyday turnover in two years [**122**].

In this real recommendation scenario, because we do not know the ground truth; i.e., which article is best for a specific user, we cannot calculate the regret. Therefore, all the results are measured using the reward, as shown in Fig. 4.2. We note that AdaLinUCB increases the reward

by 17.0%, compared to LinUCBExtracted, and by 40.8% compared to the random policy. We note that an increase in accumulated reward is typically much more substantial than the same decrease in regret. We also note that E-AdaLinUCB, where one does not assume prior knowledge on the variation factor distribution, achieves a similar performance. This experiment demonstrates the effectiveness of AdaLinUCB and E-AdaLinUCB in practical situations, where the variation factor are continuous and are possibly non-stationary, and the candidate arms are time-varying. More details on the datasets and evaluation under different parameters can be found in Appendix 4.8.5.4.

4.7. Conclusions

In this paper, we study opportunistic contextual bandits where the exploration cost is time-varying depending on external conditions such as network load or return variation in recommendations. We propose AdaLinUCB that opportunistically chooses between exploration and exploitation based on that external variation factor, i.e., taking the slots with low variation factor as opportunities for more explorations. We prove that AdaLinUCB achieves $O\left((\log T)^2\right)$ problem-dependent regret upper bound, which has a smaller coefficient thatn that of the traditional LinUCB algorithm. Extensive experiment results based on both synthetic and real-world database demonstrate the significant benefits of opportunistic exploration under large exploration cost fluctuations.

4.8. Appendix

4.8.1. Preparatory Results for Analysis. We begin with some preparatory analysis results.

LEMMA 1. *Assume that the noise satisfies the C_{noise}-sub-Gaussian condition in* (4.4)*, and assume that $||\theta_\star||_2 \leq C_{\text{theta}}$. Then, the following results hold:*

1) For any $\delta \in (0,1)$, with probability at least $1-\delta$, for all $t \geq 0$, $\theta_\star \in \left\{\theta \in \mathbb{R}^d : \|\theta_t - \theta_\star\|_{A_t} \leq \det(I_d)^{\frac{1}{2}} C_{\text{theta}} + C_{\text{noise}} \sqrt{2\log\left(\frac{\det(A_t)^{\frac{1}{2}} \det(I_d)^{-\frac{1}{2}}}{\delta}\right)}\right\}$.

2) Further, if $\|x_{t,a}\|_2 \leq C_{\text{context}}$ holds for $\forall t, \forall a \in \mathcal{D}_t$, then, with probability at least $1-\delta$, $\theta_\star \in \left\{\theta \in \mathbb{R}^d : \|\theta_t - \theta_\star\|_{A_t} \leq C_{\text{theta}} + C_{\text{noise}} \sqrt{d\log\left(\frac{1+tC_{\text{context}}^2}{\delta}\right)}\right\}$.

PROOF. it simply follows from the fact that θ_t is the result of a ridge regression, and that the sub-Gaussian condition is assumed. The technique is as in [**18**] (specifically, Theorem 2 in [**18**]). □

Lemma 1 shows that the estimation θ_t is close to the unknown parameter $\theta_\star$ in an appropriate sense.

LEMMA 2. *For $\forall t \geq 1$, $\forall a \in \mathcal{D}_t$, the following result holds,*

$$|\langle \theta_\star, x_{t,a} \rangle - \langle \theta_{t-1}, x_{t,a} \rangle| \leq \|\theta_{t-1} - \theta_\star\|_{A_{t-1}} \cdot \|x_{t,a}\|_{A_{t-1}^{-1}}.$$

PROOF. We have the following,

$$\begin{aligned}
&|\langle \theta_\star, x_{t,a} \rangle - \langle \theta_{t-1}, x_{t,a} \rangle| \\
=&|(\theta_\star - \theta_{t-1})^\top x_{t,a}| \\
=&|(\theta_\star - \theta_{t-1})^\top A_{t-1}^{\frac{1}{2}} A_{t-1}^{-\frac{1}{2}} x_{t,a}| \\
=&\left| \langle A_{t-1}^{\frac{1}{2}} (\theta_\star - \theta_{t-1}),\ A_{t-1}^{-\frac{1}{2}} x_{t,a} \rangle \right| \\
\leq&\|A_{t-1}^{\frac{1}{2}}(\theta_\star - \theta_{t-1})\|_2 \cdot \|A_{t-1}^{-\frac{1}{2}} x_{t,a}\|_2 \\
=&\sqrt{(\theta_\star - \theta_{t-1})^\top A_{t-1}^{\frac{1}{2}} A_{t-1}^{\frac{1}{2}} (\theta_\star - \theta_{t-1})} \cdot \sqrt{x_{t,a}^\top A_{t-1}^{-\frac{1}{2}} A_{t-1}^{-\frac{1}{2}} x_{t,a}} \\
=&\|\theta_{t-1} - \theta_\star\|_{A_{t-1}} \cdot \|x_{t,a}\|_{A_{t-1}^{-1}},
\end{aligned}$$

where the third equality holds by noting that a positive-definite matrix A_{t-1} is symmetric; the inequality holds by Cauchy–Schwarz inequality. □

Lemma 2 presents an upper bound on the estimation error of the reward corresponding to a given context vector. Combining Lemma 1, the first term of this upper bound, i.e., $\|\theta_{t-1} - \theta_\star\|_{A_{t-1}}$, can be upper bounded with a high probability. To further consider the property of the second term, i.e., $\|x_{t,a}\|_{A_{t-1}^{-1}}$, we bound the summation by the following result.

LEMMA 3. *Assume that $\|x_{t,a}\|_2 \leq C_{\text{context}}$ holds for $\forall t, \forall a \in \mathcal{D}_t$, and assume that $\lambda_{\min}(I_d) \geq \max\{1, C_{\text{context}}^2\}$, then the following results hold,*

$$\begin{aligned}
\sum_{t=1}^{T} \|x_{t,a_t}\|_{A_{t-1}^{-1}}^2 &\leq 2 \log \Big(\det(A_T) / \det(I_d) \Big) \\
&\leq 2 \left[d \log \Big(\frac{\text{trace}(I_d) + T C_{\text{context}}^2}{d} \Big) - \log \det(I_d) \right].
\end{aligned} \tag{4.6}$$

Lemma 3 directly follows from [**18**] (specifically, Lemma 11 of [**18**]).

REMARK 4.8.1. *(**Relax assumption iv. in Sec. 4.5.**) Now, we briefly discuss the way to relax the assumption* $\lambda_{\min}(I_d) \geq \max\{1, C_{context}^2\}$ *in Lemma 3 and the theorems using this Lemma. For this assumption, it can be relaxed by changing the initial value of matrix A in Algo. 5. Currently, we have the initial matrix value* $A_0 = I_d$*, leading to the current results in Lemma 3. If the initial value* A_0 *is changed to a positive-definite matrix with a higher minimum eigenvalue, then with a modified assumption* $\lambda_{\min}(A_0) \geq \max\{1, C_{context}^2\}$*, Lemma 3 still holds after substituting the matrix* I_d *by the new* A_0*. One example of such a new* A_0 *can be* $C_{context} \cdot I_d$ *with* $C_{context} > 1$*. Also, we note that Lemma 2 still holds after changing* A_0 *to another positive-definite matrix. Further, when changing* A_0 *to another positive-definite matrix, the first statement of Lemma 1 still holds after substituting the matrix* I_d *by the new matrix* A_0*, while the second statement following from the first one and can be changed accordingly. Thus, for assumption* $\lambda_{\min}(I_d) \geq \max\{1, C_{context}^2\}$*, we can modify the choice of* A_0 *to relax this assumption.*

4.8.2. Proof for Theorem 4.5.1. Firstly, note that we have some preparatory analysis results, as shown in Appendix 4.8.1.

We begin with some notations. Let $\mathbf{R}_{\text{total}}^{(\text{low})}(T)$ denote the accumulated regret regarding nominal rewards when the variation factor is low, i.e., $\mathbf{R}_{\text{total}}^{(\text{low})}(T) = \sum_{t=1}^{T} R_t \mathbb{1}\{L_t = \epsilon_0\}$, where $\mathbb{1}\{\cdot\}$ is the indicator function. The $\mathbf{R}_{\text{total}}^{(\text{high})}(T)$ is defined similarly as $\mathbf{R}_{\text{total}}^{(\text{high})}(T) = \sum_{t=1}^{T} R_t \mathbb{1}\{L_t = 1 - \epsilon_1\}$.

Then, we have that,

$$\tilde{\mathbf{R}}_{\text{total}}(T) = \epsilon_0 \cdot \mathbf{R}_{\text{total}}^{(\text{low})}(T) + (1 - \epsilon_1) \cdot \mathbf{R}_{\text{total}}^{(\text{high})}(T).$$

As a result, to prove this theorem, it is sufficient to prove that, with probability at least $1-\tilde{\delta}$, both of the following results hold:

$$\begin{aligned}\mathbf{R}_{\text{total}}^{(\text{low})}(T) \leq \frac{16C_{\text{noise}}^2 C_{\text{theta}}^2}{\Delta_{\min}} \Bigg[& \log(C_{\text{context}}T) + 2\log\frac{2}{\tilde{\delta}} \\ & + 2(d-1)\log\left(d\log\frac{d+TC_{\text{context}}^2}{d} + 2\log\frac{2}{\tilde{\delta}}\right) \\ & + (d-1)\log\frac{64C_{\text{noise}}^2 C_{\text{theta}}^2 C_{\text{context}}}{\Delta_{\min}^2}\Bigg]^2, \end{aligned} \tag{4.7}$$

$$\begin{aligned}\mathbf{R}_{\text{total}}^{(\text{high})}(T) \leq & \Bigg[4d\frac{N-1}{\Delta_{\min}}\left(C_{\text{noise}}\sqrt{d\log\frac{2+2TC_{\text{context}}^2}{\tilde{\delta}}} + C_{\text{theta}}\right)^2 \\ & + \Delta_{\max}C_{\text{slots}}\left(C_{\text{noise}}\sqrt{d\log\frac{2+2TC_{\text{context}}^2}{\tilde{\delta}}} + C_{\text{theta}}\right)^2\Bigg]. \end{aligned} \tag{4.8}$$

4.8.2.1. *Regret for slots with low variation factor:* Firstly, we focus on $\mathbf{R}_{\text{total}}^{(\text{low})}(T)$. For the binary-valued variation factor, let the lower threshold $l^{(-)} = \epsilon_0$. Then the variation factor $L_t = \epsilon_0$, while the normalized variation factor $\tilde{L}_t = 0$. As a result, the index $p_{t,a}$ in step 9 of Algo. 5 becomes,

$$p_{t,a} = \theta_{t-1}^\top x_{t,a} + \alpha\sqrt{x_{t,a}^\top A_{t-1}^{-1} x_{t,a}},$$

Then, for $\forall t \geq 1$ with $L_t = \epsilon_0$, if $\alpha \geq \|\theta_{t-1} - \theta_\star\|_{A_{t-1}}$, we have,

$$\begin{aligned} R_t &= \langle x_{t,a_t^\star}, \theta_\star\rangle - \langle x_{t,a_t}, \theta_\star\rangle \\ &\leq \langle x_{t,a_t^\star}, \theta_{t-1}\rangle + \alpha\|x_{t,a_t^\star}\|_{A_{t-1}^{-1}} - \langle x_{t,a_t}, \theta_\star\rangle \\ &\leq \langle x_{t,a_t}, \theta_{t-1}\rangle + \alpha\|x_{t,a_t}\|_{A_{t-1}^{-1}} - \langle x_{t,a_t}, \theta_\star\rangle \\ &= \langle x_{t,a_t}, \theta_{t-1}\rangle - \langle x_{t,a_t}, \theta_\star\rangle + \alpha\|x_{t,a_t}\|_{A_{t-1}^{-1}} \\ &\leq \|\theta_{t-1} - \theta_\star\|_{A_{t-1}}\|x_{t,a_t}\|_{A_{t-1}^{-1}} + \alpha\|x_{t,a_t}\|_{A_{t-1}^{-1}} \\ &\leq 2\alpha\|x_{t,a_t}\|_{A_{t-1}^{-1}}, \end{aligned}$$

where the inequality in the second line holds by Lemma 2 and $\alpha \geq \|\theta_{t-1} - \theta_\star\|_{A_{t-1}}$; the inequality in the third line holds by the design of the AdaLinUCB algorithm, specifically, by step 11 of Algo. 5; the inequality in the fifth line holds by Lemma 2, and the last inequality holds by $\alpha \geq \|\theta_{t-1} - \theta_\star\|_{A_{t-1}}$. As a result, we have,

$$R_t \mathbb{1}\{L_t = \epsilon_0\} \leq 2\alpha \|x_{t,a_t}\|_{A_{t-1}^{-1}},$$

with $\alpha \geq \|\theta_{t-1} - \theta_\star\|_{A_{t-1}}$. Then, we have,

$$\mathbf{R}_{\text{total}}^{(\text{low})}(T) = \sum_{t=1}^{T} R_t \mathbb{1}\{L_t = \epsilon_0\} \leq \sum_{t=1}^{T} 2\alpha \|x_{t,a_t}\|_{A_{t-1}^{-1}}, \tag{4.9}$$

with $\alpha \geq \|\theta_{T-1} - \theta_\star\|_{A_{T-1}}$.

We also note that, by Lemma 1, with probability at least $1 - \frac{\tilde{\delta}}{2}$, for all t,

$$\begin{aligned} \theta_\star \in \Big\{ & \theta \in \mathbb{R}^d : \|\theta_t - \theta_\star\|_{A_t} \\ & \leq C_{\text{theta}} + C_{\text{noise}} \sqrt{d \log\left(\frac{2 + 2tC_{\text{context}}^2}{\tilde{\delta}}\right)} \Big\}, \end{aligned} \tag{4.10}$$

which substitutes the δ in Lemma 1 by $\frac{\tilde{\delta}}{2}$.

Further, we note that,

$$\mathbf{R}_{\text{total}}(T) = \sum_{t=1}^{T} R_t \leq \sum_{t=1}^{T} \frac{R_t^2}{\Delta_{\min}}, \tag{4.11}$$

where the inequality holds since either $R_t = 0$ or $\Delta_{\min} <= R_t$.

Then, by combining (4.9), (4.10), and (4.11), it follows from a similar argument as [**18**] (specifically, the proof of Theorem 5 in [**18**]) that (4.7) holds with probability at least $1 - \frac{\tilde{\delta}}{2}$. Note that the proof procedure uses Lemma 3 and the single optimal context condition.

4.8.2.2. *Regret for slots with high variation factor.* Now, we focus on $\mathbf{R}_{\text{total}}^{(\text{high})}(T)$. We begin with some notations. The N possible values of context vectors are denoted by $x_{(1)}, x_{(2)}, \cdots, x_{(N)}$ respectively. Without loss of generality, we assume that $x_{(1)}$ is the optimal context value, i.e., $x_{(1)} = x_\star$. Let $m_{t,\star}$ be the number of times that the arm with the optimal context value has been pulled before time slot t, i.e., $m_{t,\star} = \sum_{\tau=1}^{t} \mathbb{1}\{x_{\tau,a_\tau} = x_\star\}$. Similarly, let $m_{t,(n)}$ be the

number of times that the arm with context value $x_{(n)}$ has been pulled before time slot t, i.e., $m_{t,(n)} = \sum_{\tau=1}^{t} \mathbb{1}\{x_{\tau,a_\tau} = x_{(n)}\}$. In addition, let $m_{t,\star}^{(\text{low})}$ be the number of times when the variation factor is low and the arm with the optimal context value $x_\star$ has been pulled during t-slot period, i.e., $m_{t,\star}^{(\text{low})} = \sum_{\tau=1}^{t} \mathbb{1}\{x_{\tau,a_\tau} = x_\star\} \cdot \mathbb{1}\{L_\tau = \epsilon_0\}$. Let $m_{t,\text{all}}^{(\text{low})}$ be the number of times when the variation factor is low during t-slot period, i.e., $m_{t,\text{all}}^{(\text{low})} = \sum_{\tau=1}^{t} \mathbb{1}\{L_\tau = \epsilon_0\}$. Further, let $m_{t,\text{subopt}}^{(\text{low})}$ be the number of times when the variation factor is low and the arm with a suboptimal context value has been pulled during t-slots, i.e., $m_{t,\text{subopt}}^{(\text{low})} = m_{t,\text{all}}^{(\text{low})} - m_{t,\star}^{(\text{low})}$.

LEMMA 4. *For the AdaLinUCB algorithm, the following inequality holds, for any* $n = 1, 2, \cdots, N$,

$$\|x_{(n)}\|_{A_{t-1}^{-1}} \leq \sqrt{\frac{d}{m_{t-1,(n)}}}.$$

PROOF. We note that,

$$\begin{aligned}
d &= \text{trace}(I_d) = \text{trace}\left(A_{t-1}^{-1} A_{t-1}\right) \\
&= \text{trace}\left(A_{t-1}^{-1}\left[\sum_{i=1}^{N} m_{t-1,(i)} \cdot x_{(i)} x_{(i)}^\top + I_d\right]\right) \\
&= \text{trace}\left(\sum_{i=1}^{N} m_{t-1,(i)} \cdot A_{t-1}^{-1} x_{(i)} x_{(i)}^\top + A_{t-1}^{-1}\right) \\
&= \sum_{i=1}^{N} m_{t-1,(i)} \cdot \text{trace}\left(A_{t-1}^{-1} x_{(i)} x_{(i)}^\top\right) + \text{trace}\left(A_{t-1}^{-1}\right) \\
&= \sum_{i=1}^{N} m_{t-1,(i)} \cdot \text{trace}\left(x_{(i)}^\top A_{t-1}^{-1} x_{(i)}\right) + \text{trace}\left(A_{t-1}^{-1}\right) \\
&= \sum_{i=1}^{N} m_{t-1,(i)} \cdot x_{(i)}^\top A_{t-1}^{-1} x_{(i)} + \text{trace}\left(A_{t-1}^{-1}\right) \\
&\geq m_{t-1,(n)} \cdot x_{(n)}^\top A_{t-1}^{-1} x_{(n)}, \quad \forall n = 1, 2, \cdots, N,
\end{aligned}$$

where the last inequality holds by noting that A_{t-1}^{-1} is a positive-definite matrix. Then, the results follow. □

In the following, we let,

$$\alpha_T = C_{\text{noise}} \sqrt{d \log \left(\frac{2 + 2TC_{\text{context}}^2}{\tilde{\delta}} \right)} + C_{\text{theta}}. \tag{4.12}$$

Thus, by (4.10), with probability at least $1 - \frac{\tilde{\delta}}{2}$, for all $t \leq T$, the following inequality holds.

$$\|\theta_t - \theta_\star\|_{A_t} \leq \alpha_T. \tag{4.13}$$

Let the α in AdaLinUCB algorithm be $\alpha = \alpha_T$.

LEMMA 5. *When inequality* (4.13) *holds, for any slot t with variation factor $L_t = 1 - \epsilon_1$, if,*

$$\langle x_\star, \theta_\star \rangle - \langle x_{(n)}, \theta_\star \rangle > \alpha_T \|x_\star\|_{A_{t-1}^{-1}} + \alpha_T \|x_{(n)}\|_{A_{t-1}^{-1}}, \ \forall n, \tag{4.14}$$

then the arm with the optimal context is pulled in slot t, i.e., $R_t = 0$.

PROOF. For the binary-valued variation factor, let the higher threshold of variation factor $l^{(+)} = 1 - \epsilon_1$. Thus, when the variation factor is high, i.e., $L_t = 1 - \epsilon_1$, the truncated variation factor becomes $\tilde{L}_t = 1$. As a result, the index in step 9 of Algo. 5 becomes,

$$p_{t,a} = \theta_{t-1}^\top x_{t,a} = \langle x_{t,a}, \theta_{t-1} \rangle.$$

As a result, to prove that the arm with optimal context value is selected, it is sufficient to prove that,

$$\langle x_\star, \theta_{t-1} \rangle - \langle x_{(n)}, \theta_{t-1} \rangle > 0, \ \forall n = 2, \cdots, N.$$

When inequality (4.13) holds, by Lemma 2 , we have that,

$$\langle x_\star, \theta_{t-1} \rangle \geq \langle x_\star, \theta_\star \rangle - \alpha_T \|x_\star\|_{A_{t-1}^{-1}},$$

and,

$$\langle x_{(n)}, \theta_{t-1} \rangle \leq \langle x_{(n)}, \theta_\star \rangle - \alpha_T \|x_{(n)}\|_{A_{t-1}^{-1}}.$$

As a result, for any $n = 2, 3, \cdots, N$,

$$\begin{aligned}
&\langle x_\star, \theta_{t-1}\rangle - \langle x_{(n)}, \theta_{t-1}\rangle \\
\geq &\langle x_\star, \theta_\star\rangle - \langle x_{(n)}, \theta_\star\rangle - \alpha_T \|x_\star\|_{A_{t-1}^{-1}} - \alpha_T \|x_{(n)}\|_{A_{t-1}^{-1}} \\
> &0,
\end{aligned}$$

where the last inequality holds by the condition (4.14) of this Lemma, which completes the proof. □

By Lemma 5, when inequality (4.13) holds, for any slot t with variation factor $L_t = 1 - \epsilon_1$, if both of the following inequalities holds,

$$\alpha_T \|x_\star\|_{A_{t-1}^{-1}} \leq \frac{\Delta_{\min}}{2}, \tag{4.15}$$

$$\alpha_T \|x_{(n)}\|_{A_{t-1}^{-1}} < \frac{\langle x_\star, \theta_\star\rangle - \langle x_{(n)}, \theta_\star\rangle}{2}, \ \forall n = 2, \cdots, N, \tag{4.16}$$

then the arm with the optimal context is selected with probability at least $1 - \tilde{\delta}$.

Now, we analyze when (4.16) holds. For any suboptimal context value $x_{(n)}$ with $n \neq 1$, by Lemma 4, (4.16) holds when,

$$m_{t-1,(n)} > \frac{4d\alpha_T^2}{\left[\langle x_\star, \theta_\star\rangle - \langle x_{(n)}, \theta_\star\rangle\right]^2}.$$

As a result, before (4.16) is satisfied, pulling the arms with the suboptimal context values increases $\mathbf{R}_{\text{total}}^{(\text{high})}(T)$ by at most,

$$\sum_{n=2}^{N} \frac{4d\alpha_T^2}{\langle x_\star, \theta_\star\rangle - \langle x_{(n)}, \theta_\star\rangle} \leq (N-1)\frac{4d}{\Delta_{\min}}\alpha_T^2. \tag{4.17}$$

Note that the r.h.s. of (4.17) is the first term of (4.8) by recalling α_T definition in (4.12).

Now, we focus on analyzing when (4.15) holds. For optimal context value $x_{(1)} = x_\star$, by Lemma 4, (4.15) holds when,

$$m_{t-1,\star} > \frac{4d\alpha_T^2}{\Delta_{\min}^2}. \tag{4.18}$$

To analyze when (4.18) holds, we can take advantage of (4.7), and note that,

$$m_{t,\text{subopt}}^{(\text{low})} \leq \frac{\mathbf{R}_{\text{total}}^{(\text{low})}(t)}{\Delta_{\min}}.$$

Thus, by (4.7), with probability at least $1 - \frac{\tilde{\delta}}{2}$, for all t,

$$\begin{aligned} m_{t,\text{subopt}}^{(\text{low})} \leq \frac{16C_{\text{noise}}^2 C_{\text{theta}}^2}{\Delta_{\min}^2} \Bigg[& \log(C_{\text{context}} t) \\ & + 2(d-1)\log\left(d\log\frac{d + tC_{\text{context}}^2}{d} + 2\log\frac{2}{\tilde{\delta}}\right) \\ & + (d-1)\log\frac{64C_{\text{noise}}^2 C_{\text{theta}}^2 C_{\text{context}}}{\Delta_{\min}^2} + 2\log\frac{2}{\tilde{\delta}}\Bigg]^2, \end{aligned} \tag{4.19}$$

where the probability is introduced by (4.10) when proving (4.7).

Further, for the binary-valued variation factor, by Hoeffding's inequality, we have that, with probability at least $1 - \frac{\tilde{\delta}}{2}$,

$$m_{t,\text{all}}^{(\text{low})} \geq \rho t - \sqrt{\frac{t}{2}\log\frac{2}{\tilde{\delta}}},$$

which also holds when $t = \alpha_T^2 C_{\text{slots}}$. Thus, with probability at least $1 - \frac{\tilde{\delta}}{2}$,

$$m_{\alpha_T^2 C_{\text{slots}},\text{all}}^{(\text{low})} \geq \rho \cdot \alpha_T^2 C_{\text{slots}} - \sqrt{\frac{\alpha_T^2 C_{\text{slots}}}{2}\log\frac{2}{\tilde{\delta}}}. \tag{4.20}$$

Then, by combining (4.19) and (4.20), and by recalling C_{slots} definition in (4.5), with probability at least $1 - \tilde{\delta}$,

$$\begin{aligned} m_{\alpha_T^2 C_{\text{slots}},\star} &\geq m_{\alpha_T^2 C_{\text{slots}},\star}^{(\text{low})} \\ &= m_{\alpha_T^2 C_{\text{slots}},\text{all}}^{(\text{low})} - m_{\alpha_T^2 C_{\text{slots}},\text{subopt}}^{(\text{low})} \\ &\geq \frac{4d\alpha_T^2}{\Delta_{\min}^2}. \end{aligned}$$

Thus, with probability at least $1 - \tilde{\delta}$, for $\forall t \geq \alpha_T^2 C_{\text{slots}}$, the inequality (4.15) holds. As a result, with probability at least $1 - \tilde{\delta}$, before (4.15) is satisfied, pulling the arms with the suboptimal

Algorithm 6 LinUCB(Extracted)

1: Inputs: $\alpha \in \mathbb{R}_+$, $d \in \mathbb{N}$.
2: $A \leftarrow \boldsymbol{I}_d$ {The d-by-d identity matrix}
3: $b \leftarrow \mathbf{0}_d$
4: **for** $t = 1, 2, 3, \cdots, T$ **do**
5: $\quad \theta_{t-1} = A^{-1}b$
6: $\quad$ Observe possible arm set $\mathcal{D}_t$, and observe associated context vectors $x_{t,a}, \forall a \in \mathcal{D}_t$.
7: $\quad$ **for** $a \in \mathcal{D}_t$ **do**
8: $\quad\quad p_{t,a} \leftarrow \theta_{t-1}^\top x_{t,a} + \alpha\sqrt{x_{t,a}^\top A^{-1} x_{t,a}}$ {Computes upper confidence bound}
9: $\quad\quad a_t = \arg\max_{a \in \mathcal{D}_t} \theta_{t-1}^\top x_{t,a} + \alpha\sqrt{x_{t,a}^\top A^{-1} x_{t,a}}$ {Computes upper confidence bound}
10: $\quad$ **end for**
11: $\quad$ Choose action $a_t = \arg\max_{a \in \mathcal{D}_t} p_{t,a}$ with ties broken arbitrarily.
12: $\quad$ Observe nominal reward r_{t,a_t}
13: $\quad A \leftarrow A + x_{t,a_t} x_{t,a_t}^\top$
14: $\quad b \leftarrow b + x_{t,a_t} r_{t,a_t}$
15: **end for**

context values increases $\mathbf{R}_{\text{total}}^{(\text{high})}(T)$ by at most $\alpha_T^2 C_{\text{slots}} \Delta_{\max}$. By combining (4.17), we have that, with probability at least $1 - \tilde{\delta}$, the inequality (4.8) for $\mathbf{R}_{\text{total}}^{(\text{high})}(T)$ holds.

4.8.2.3. *Combine Results and Finish Proof.* Further by noting that probabilities introduced in this proof procedure only comes from two events: i) confidence set for θ_t in (4.10); ii) lower bound for number of total slots with low variation factor as in (4.20). Note that each of these two events with probability at least $1 - \frac{\tilde{\delta}}{2}$ and that they are independent. Thus, with probability at least $1 - \tilde{\delta}$, both inequalities (4.7) and (4.8) hold, which completes the proof.

4.8.3. Performance Analysis of LinUCB.

4.8.3.1. *LinUCB Algorithm Notation.* In opportunistic contextual bandit problem, one way to select bandits is to ignore the variation factor, i.e., L_t, and just employ the LinUCB algorithm, as shown in Algorithm 6. This algorithm is denoted as LinUCBExtracted in numerical restuls.

In Algo. 6, for each time slot, the algorithm updates an matrix A and a vector b, so that to estimate the unknown parameter for the linear function of context vector. To make the notation clear, denote $A_t = I_d + \sum_{\tau=1}^{t} x_{\tau,a_\tau} x_{\tau,a_\tau}^\top$, which is the matrix A updated in step 13 for each time slot. It directly follows that $A_t, \forall t \geq 0$ is a positive-definite matrix. Denote $b_t = \sum_{\tau=1}^{t} x_{\tau,a_\tau} r_{\tau,a_\tau}$, which is the vector b updated in step 14 for each time slot t.

As a result, the estimation of the unknown parameter $\theta_\star$ is denoted by θ_t, as shown in step 5, which satisfies,

$$\begin{aligned}\theta_t &= A_t^{-1} b_t \\ &= \left(I_d + \sum_{\tau=1}^{t} x_{\tau,a_\tau} x_{\tau,a_\tau}^\top \right)^{-1} \sum_{\tau=1}^{t} x_{\tau,a_\tau} r_{\tau,a_\tau}.\end{aligned} \tag{4.21}$$

Note that θ_t is the result of a ridge regression. That is, θ_t is the coefficient that minimize a penalized residual sum of squares, i.e.,

$$\theta_t = \arg\min_{\theta} \left\{ \sum_{\tau=1}^{t} \left(r_{\tau,a_\tau} - \langle \theta, x_{\tau,a_\tau} \rangle \right)^2 + \|\theta\|_2^2 \right\} \tag{4.22}$$

Here, the complexity parameter that controls the amount of shrinkage is chosen as 1.

Also, we note that the upper confidence index $p_{t,a}$, as shown in step 9 of Algo. 6 consists of two parts. The first part $\theta_{t-1}^\top x_{t,a} = \langle \theta_{t-1}, x_{t,a} \rangle$ is the estimation of the corresponding reward, using the up-to-date estimation of the unknown parameter, i.e., θ_{t-1}. The second part, i.e., $\alpha\sqrt{x_{t,a}^\top A_{t-1}^{-1} x_{t,a}} = \alpha \|x_{t,a}\|_{A_{t-1}^{-1}}$, is related to the uncertainty of reward estimation.

In the following, to analyze the performance of LinUCB algorithm, we assume the same assumptions as in Sec. 4.5.

4.8.3.2. *General Performance Bound.* Now, we analyze the performance of LinUCB algorithm. We note that the initial analysis effort of LinUCB [**17**] presents analysis result for a modified version of LinUCB to satisfied the independent requirement by applying Azuma/Hoeffding inequality [**17**]. As a result, we firstly provide the general performance analysis of LinUCB. We have used analysis technique as in [**18**]. (Note that [**18**] provides analysis for another algorithm instead of LinUCB, but its technique is helpful.)

Firstly, we note that since A_t, b_t, θ_t has the same definition as that in AdaLinUCB Algorithm, the previous Lemma 1, Lemma 2, and Lemma 3 also hold here for LinUCB algorithm. Then, we have the following results.

THEOREM 4.8.1. *(The general regret bound of LinUCB). For the LinUCB algorithm in Algo. 6, consider traditional contextual bandits with linear payoffs, the following results hold.*

1) $\forall t \geq 1$*, if* $\alpha \geq \|\theta_{t-1} - \theta_\star\|_{A_{t-1}}$*, then the one-step regret (regarding nominal reward) satisfies,*

$$R_t \leq 2\alpha \|x_{t,a_t}\|_{A_{t-1}^{-1}}.$$

2) $\forall \delta \in (0,1)$*, with probability at least* $1-\delta$*, the accumulated* T*-slot regret (regarding nominal reward) satisfies,*

$$\begin{aligned}\mathbf{R}_{\text{total}}(T) \leq & \sqrt{8T}\left[C_{\text{noise}}\sqrt{d\log\left(\frac{1+TC_{\text{context}}^2}{\delta}\right)} + C_{\text{theta}}\right] \\ & \cdot \sqrt{d\log\left[\frac{\text{trace}(I_d) + TC_{\text{context}}^2}{d}\right] - \log\det(I_d)}.\end{aligned} \tag{4.23}$$

Proof. We begin by analyzing the one-step regret (regarding nominal reward) of LinUCB algorithm in Algo. 6. For $\forall t \geq 1$, with $\alpha \geq \|\theta_{t-1} - \theta_\star\|_{A_{t-1}}$, we have,

$$\begin{aligned} R_t &= \langle x_{t,a_t^\star}, \theta_\star\rangle - \langle x_{t,a_t}, \theta_\star\rangle \\ &\leq \langle x_{t,a_t^\star}, \theta_{t-1}\rangle + \alpha\|x_{t,a_t^\star}\|_{A_{t-1}^{-1}} - \langle x_{t,a_t}, \theta_\star\rangle \\ &\leq \langle x_{t,a_t}, \theta_{t-1}\rangle + \alpha\|x_{t,a_t}\|_{A_{t-1}^{-1}} - \langle x_{t,a_t}, \theta_\star\rangle \\ &= \langle x_{t,a_t}, \theta_{t-1}\rangle - \langle x_{t,a_t}, \theta_\star\rangle + \alpha\|x_{t,a_t}\|_{A_{t-1}^{-1}} \\ &\leq \|\theta_{t-1} - \theta_\star\|_{A_{t-1}}\|x_{t,a_t}\|_{A_{t-1}^{-1}} + \alpha\|x_{t,a_t}\|_{A_{t-1}^{-1}} \\ &\leq 2\alpha\|x_{t,a_t}\|_{A_{t-1}^{-1}}, \end{aligned}$$

where the inequality in the second line holds by Lemma 2 and $\alpha \geq \|\theta_{t-1} - \theta_\star\|_{A_{t-1}}$; the inequality in the third line holds by the design of the LinUCB algorithm, specifically, by step 11 of Algo. 6; the inequality in the fifth line holds by Lemma 2, and the last inequality holds by $\alpha \geq \|\theta_{t-1} - \theta_\star\|_{A_{t-1}}$. As a result, the first statement is proved.

Now, we analyze the accumulated regret. Let,

$$\alpha_T = C_{\text{noise}}\sqrt{d\log\left(\frac{1+TC_{\text{context}}^2}{\delta}\right)} + C_{\text{theta}}.$$

Then, by Lemma 1, with probability at least $1-\delta$, for $\forall t \in [1,T]$, $\alpha_T \geq \|\theta_{t-1} - \theta_\star\|_{A_{t-1}}$. As a result, with probability at least $1-\delta$,

$$\begin{aligned}
\mathbf{R}_{\text{total}}(T) = \sum_{t=1}^{T} R_t &\leq \sqrt{T \sum_{t=1}^{T} R_t^2} \\
&\leq \sqrt{T \cdot 4\alpha_T^2 \sum_{t=1}^{T} \|x_{t,a_t}\|^2_{A_{t-1}^{-1}}} \\
&\leq \sqrt{8T\alpha_T^2} \\
&\cdot \sqrt{d \log \left[\frac{\text{trace}(I_d) + TC_{\text{context}}^2}{d} \right] - \log\det(I_d)},
\end{aligned}$$

where the first inequality holds by Jensen's inequality; the second inequality holds by statement 1); the third inequality holds by Lemma 3. Thus, by substituting the value of α_T, the inequality (4.23) holds. □

4.8.3.3. *Problem-Dependent Bound.* Now, we study the problem-dependent bound of LinUCB, and have the following results.

THEOREM 4.8.2. *For the LinUCB algorithm in Algo. 6, consider traditional contextual bandit setting with linear payoffs, the accumulated T-slot regret (regarding nominal reward) satisfies,*

$$\begin{aligned}
\mathbf{R}_{\text{total}}(T) \leq & \frac{16C_{\text{noise}}^2 C_{\text{theta}}^2}{\Delta_{\min}} \Bigg\{ \log(C_{\text{context}} T) + 2\log\frac{1}{\delta} \\
& + 2(d-1)\log\left[d\log\frac{d + TC_{\text{context}}^2}{d} + 2\log\frac{1}{\delta} \right] \\
& + (d-1)\log\frac{64C_{\text{noise}}^2 C_{\text{theta}}^2 C_{\text{context}}}{\Delta_{\min}^2} \Bigg\}^2.
\end{aligned}$$

PROOF. We note that,

$$\mathbf{R}_{\text{total}}(T) = \sum_{t=1}^{T} R_t \leq \sum_{t=1}^{T} \frac{R_t^2}{\Delta_{\min}},$$

Algorithm 7 LinUCB(Multiply)

Inputs: $\alpha \in \mathbb{R}_+$, $d \in \mathbb{N}$.
$A \leftarrow \boldsymbol{I}_d$ {The d-by-d identity matrix}
$b \leftarrow \mathbf{0}_d$
for $t = 1, 2, 3, \cdots, T$ **do**
  $\theta_{t-1} = A^{-1}b$
  Observe possible arm set $\mathcal{D}_t$, and observe associated context vectors $x_{t,a}, \forall a \in \mathcal{D}_t$.
  Observe L_t, and get $\tilde{x}_{t,a} = L_t \cdot x_{t,a}, \forall a \in \mathcal{D}_t$.
  for $a \in \mathcal{D}_t$ **do**
    $p_{t,a} \leftarrow \theta_{t-1}^\top \tilde{x}_{t,a} + \alpha\sqrt{\tilde{x}_{t,a}^\top A^{-1} \tilde{x}_{t,a}}$ {Computes upper confidence bound}
  end for
  Choose action $a_t = \arg\max_{a \in \mathcal{D}_t} p_{t,a}$ with ties broken arbitrarily.
  Observe nominal reward r_{t,a_t} and get actual reward $\tilde{r}_{t,a_t} = L_t \cdot r_{t,a_t}$.
  $A \leftarrow A + \tilde{x}_{t,a_t}\tilde{x}_{t,a_t}^\top$
  $b \leftarrow b + \tilde{x}_{t,a_t}\tilde{r}_{t,a_t}$
end for

where the inequality holds since either $R_t = 0$ or $\Delta_{\min} <= R_t$. Then, the results follows from same proof as in [**18**] (see the proof of Theorem 5 in [**18**]). Note that the proof procedure uses Lemma 3 and the single optimal context condition. □

4.8.3.4. *Performance for Opportunistic Case - Proof of Theorem 4.5.3.* Note that the arm selection strategy in LinUCB in Algo. 6 is independent of the value of L_t. Thus, when L_t is i.i.d. over time, we have that $\tilde{\mathbf{R}}_{\text{total}}(T) = \bar{L}\mathbf{R}_{\text{total}}(T)$. As a result, Theorem 4.5.3 directly follows from Theorem 4.8.2.

4.8.3.5. *Another Way to Apply LinUCB in Opportunistic Linar Bandits.* Beside the LinUCBExtracted algorithm in Algo. 6, we also note that there is another way to directly apply in LinUCB in opportunistic contextual bandit environment. Recall that the LinUCBExtracted algorithm in Algo. 6 is based on the linear relationship, $\mathbb{E}[r_{t,a}|x_{t,a}] = \langle x_{t,a}, \theta_\star \rangle$. We can also apply the LinUCBMultiply algorithm in Algo. 7, which is based on the linear relationship, $\mathbb{E}[L_t \cdot r_{t,a}|x_{t,a}, L_t] = \langle L_t \cdot x_{t,a}, \theta_\star \rangle$, i.e., regarding $L_t \cdot x_{t,a}$ as context vector.

Thus, we have also implemented LinUCBMultiply in the numerical results. However, from the experiment results, LinUCBExtracted algorithm has a better performance than LinUCBMultiply.

4.8.4. AdaLinUCB for Disjoint Model. In above, we focus on the design and analysis of opportunistic contextual bandit for the joint model. However, it should be noted that, the

Algorithm 8 AdaLinUCB - Disjoint Model

Inputs: $\alpha \in \mathbb{R}_+$, $d \in \mathbb{N}$, $l^{(+)}$, $l^{(-)}$.
$A^{(a)} \leftarrow \boldsymbol{I}_d, \ \forall a$
$b^{(a)} \leftarrow \mathbf{0}_d, \ \forall a$
for $t = 1, 2, 3, \cdots, T$ **do**
  Observe possible arm set $\mathcal{D}_t$, and observe associated context vectors $x_{t,a}, \forall a \in \mathcal{D}_t$.
  Observe L_t and calculate $\tilde{L}_t$ by (4.3).
  for $a \in \mathcal{D}_t$ **do**
    $\theta_{t-1}^{(a)} = [A^{(a)}]^{-1} b^{(a)}$
    $p_{t,a} \leftarrow [\theta_{t-1}^{(a)}]^\top x_{t,a} + \alpha \sqrt{(1 - \tilde{L}_t) x_{t,a}^\top [A^{(a)}]^{-1} x_{t,a}}$
  end for
  Choose action $a_t = \arg\max_{a \in \mathcal{D}_t} p_{t,a}$ with ties broken arbitrarily.
  Observe nominal reward r_{t,a_t}.
  $A^{(a)} \leftarrow A^{(a)} + x_{t,a_t} x_{t,a_t}^\top$
  $b^{(a)} \leftarrow b^{(a)} + x_{t,a_t} r_{t,a_t}$
end for

AdaLinUCB algorithm in Algo. 5 can be modified slightly and then be applied to the disjoint model, which is shown in the Algo. 8.

Here, we note that the joint model is the model introduced in Sec. 4.3:, which assumes that,

$$\mathbb{E}[r_{t,a} | x_{t,a}] = \langle x_{t,a}, \theta_\star \rangle,$$

where $x_{t,a}$ is a context vector and $\theta_\star$ is the unknown coefficient vector. Another model is the disjoint model, which assumes that,

$$\mathbb{E}[r_{t,a} | x_{t,a}] = \langle x_{t,a}, \theta_\star^{(a)} \rangle,$$

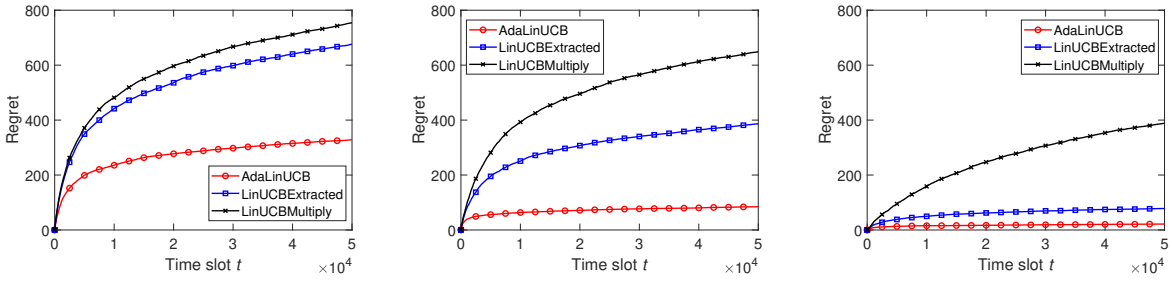
where $x_{t,a}$ is a context vector and $\theta_\star^{(a)}$ is the unknown coefficient vector for arm a. This model is called disjoint since the parameters are not shared among different arms.

The joint and disjoint models correspond to different models for linear contextual bandit problems, as introduced in the seminal paper on LinUCB [**48**].

4.8.5. More Numerical Results. We have implemented AdaLinUCB (as in Algo. 5), LinUCBExtracted (as in Algo. 6), and LinUCBMultiply (as in Algo. 7). We have also implemented **E-AdaLinUCB** algorithm, which is an algorithm that adjusts the threshold $l^{(+)}$ and $l^{(-)}$ based on the empirical distribution of L_t. Specifically, the E-AdaLinUCB algorithm maintains the empirical histogram for the variation factors (or its moving average version for non-stationary cases), and

selects $l^{(+)}$ and $l^{(-)}$ accordingly. Furthermore, the results for **KernelUCB** is shown in Appendix 4.8.5.3.

4.8.5.1. *Synthetic Scenario with Binary-Valued variation Factor.* Fig. 4.3 shows the performance of different algorithms with binary-valued variation factor for different value of ρ. From the simulation result, the AdaLinUCB algorithm significantly outperforms other algorithms for different values of ρ.



(a) $\rho = 0.1$ (b) $\rho = 0.5$ (c) $\rho = 0.9$

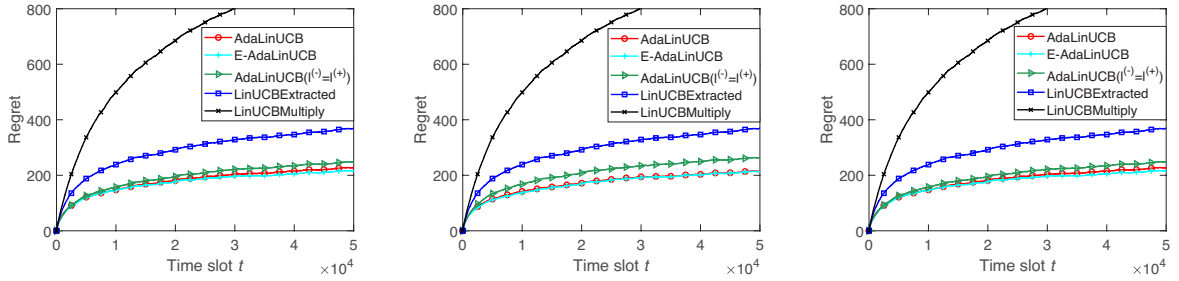
FIGURE 4.3. Regret under binary-valued variation factor.

4.8.5.2. *Synthetic Scenario with Beta Distributed variation Factor.* Here, we define $l_\rho^{(-)}$ as the lower threshold such that $\mathbb{P}\{L_t \leq l_\rho^{(-)} = \rho\}$, and $l_\rho^{(+)}$ as the lower threshold such that $\mathbb{P}\{L_t \geq l_\rho^{(+)} = \rho\}$. The simulation results demonstrate that, with appropriately chosen parameters, the proposed AdaLinUCB algorithm (and its empirical version E-AdaLinUCB) achieves good performance by leveraging the variation factor fluctuation in opportunistic contextual bandits. Furthermore, it turns out that, for a large range of $l^{(+)}$ and $l^{(-)}$ values, AdaLinUCB performs well. Meanwhile, E-AdaLinUCB has a similar performance as that of AdaLinUCB in different scenarios.

In Fig. 4.4, we implement both AdaLinUCB with a single threshold $l^{(-)} = l^{(+)}$, and AdaLinUCB (and E-AdaLinUCB) with two different threshold values. We find that AdaLinUCB and E-AdaLinUCB perform well for all these appropriate choices of $l^{(-)}$ and $l^{(+)}$. In addition, even in the special case with a single threshold $l^{(-)} = l^{(+)}$, AdaLinUCB has a better performance than other algorithms.

We evaluate the impact of $l^{(-)}$ and $l^{(+)}$ separately with the other one fixed in Fig. 4.5 and Fig. 4.6, respectively. Compared them, we can see that the impact of threshold values under continuous variation factor is insignificant (when $l^{(+)}$ and $l^{(-)}$ are changing in wide appropriate

ranges), and the regret of AdaLinUCB is significantly lower than that of LinUCBExtracted and LinUCBMultiple.

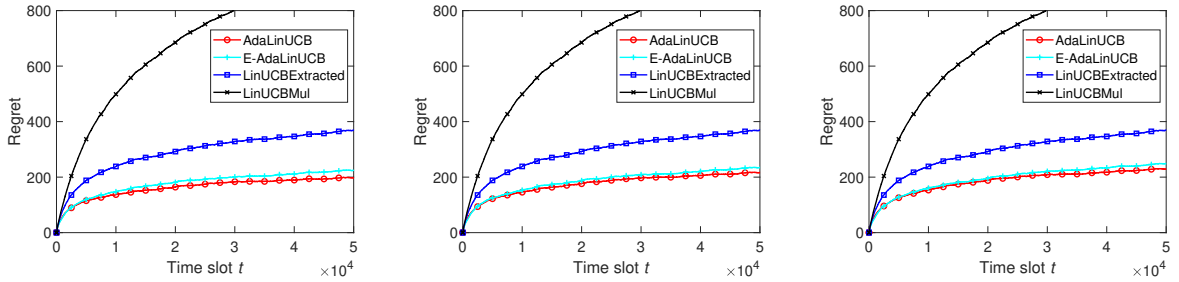


(a) AdaLinUCB: $l^{(-)} = l^{(-)}_{0.05}, l^{(+)} = l^{(+)}_{0.05}$; AdaLinUCB($l^{(-)} = l^{(+)}$):$l^{(-)} = l^{(+)} = 0.45$

(b) AdaLinUCB: $l^{(-)} = l^{(-)}_{0}, l^{(+)} = l^{(+)}_{0}$; AdaLinUCB($l^{(-)} = l^{(+)}$): $l^{(-)} = l^{(+)} = 0.5$

(c) AdaLinUCB: $l^{(-)} = l^{(-)}_{0.1}, l^{(+)} = l^{(+)}_{0.1}$; AdaLinUCB($l^{(-)} = l^{(+)}$): $l^{(-)} = l^{(+)} = 0.55$

FIGURE 4.4. Regret under beta distributed variation factor with a single threshold.



(a) $l^{(-)} = l^{(-)}_{0.05}, l^{(+)} = l^{(+)}_{0}$

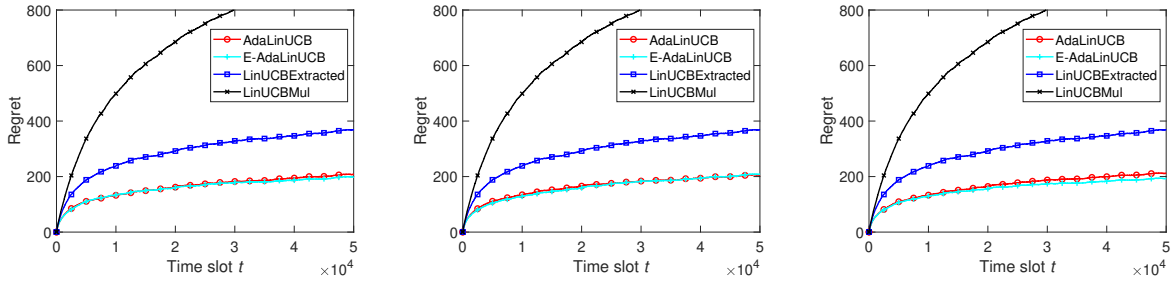
(b) $l^{(-)} = l^{(-)}_{0.1}, l^{(+)} = l^{(+)}_{0}$

(c) $l^{(-)} = l^{(-)}_{0.15}, l^{(+)} = l^{(+)}_{0}$

FIGURE 4.5. Regret under beta distributed variation factor with different values of $l^{(-)}$.

4.8.5.3. *Compare with KernelUCB.* We have also implemented KernelUCB [**49**] which is a kernel-based upper confidence bound algorithm. It applies for general contextual bandits with non-linear payoffs. It can characterize general non-linear relationship between the context vector and reward based on the kernel that defines the similarity between two data points. There are many widely used kernels, such as Gaussian kernel, Laplacian kernel and polynomial kernel [**123**].

We demonstrate KernelUCB in Algo. 9. The algorithm is based on paper [**49**]. Furthermore, in line 10, we have actually used the technique of Schur complement [**115**] to update of kernel matrix $\boldsymbol{K}_t$ so as to boost the implementation of KernelUCB.



(a) $l^{(-)} = l_0^{(-)}, l^{(+)} = l_{0.05}^{(+)}$ (b) $l^{(-)} = l_0^{(-)}, l^{(+)} = l_{0.1}^{(+)}$ (c) $l^{(-)} = l_0^{(-)}, l^{(+)} = l_{0.15}^{(+)}$

FIGURE 4.6. Regret under beta distributed variation factor with different values of $l^{(+)}$.

Fig. 4.7 demonstrates the performance of AdaLinUCB, LinUCBExtracted, and KernelUCB (with carefully selected hyper-parameters) for different scenarios. Note that the performance of KernelUCB highly depends on the choice of hyper-parameter. To make a fair comparison, we test the performance of KernelUCB for different hyper-parameter values, and chooses the hyper-parameters with the best performance (among the hyper-parameter values that we have experimented), i.e., $\Gamma_{\text{kernel}} = 2$ for Gaussian kernel $k(z_1, z_2) = \exp(-\Gamma_{\text{kernel}}||z_1 - z_2||^2)$, $\lambda_{\text{regularization}} = 0.5$ for kernel ridge regression. As shown in Fig. 4.7, AdaLinUCB outperforms KernelUCB under both binary-valued variation factor and continuous variation factor.

Besides the less competitive performance, as show in Fig. 4.7, there are two other severe drawbacks that prevents the application of KernelUCB in many practical scenarios. Firstly, its performance is highly sensitive to the choice of hyper-parameters. As discussed above, we have tested the performance of KernelUCB for different hyper-parameter values, and chooses the hyper-parameters with the best performance for a fair comparison. However, even when the hyper-parameters just changes slightly (or environment such as variation factor fluctuation changes slightly), the performance of KernelUCB can deteriorate severely such that it performs even worse then LinUCBExtracted.

Secondly, KernelUCB suffers from the high computational complexity problem. Even if we have used the technique of Schur complement [**115**] to update of $\boldsymbol{K}_t$ so as to boost the implementation of KernelUCB as paper [**49**] , it still suffers from prohibitively high computational complexity even for moderately long time horizon. This is also the reason why Fig. 4.7 has a shorter time horizon than other figures. Specifically, even to run a 10^4-slot simulation, the time to run KernelUCB algorithm

Algorithm 9 KernelUCB

Inputs: $\alpha \in \mathbb{R}_+$, $d \in \mathbb{N}$, $k(\cdot,\cdot)$, $\lambda = \lambda_{\text{regularization}}$.
for $t = 1, 2, 3, \cdots, T$ **do**
  Observe possible arm set $\mathcal{D}_t$, and observe associated context vectors $x_{t,a}, \forall a \in \mathcal{D}_t$.
  Observe L_t, and get augment context $\tilde{x}_{t,a} = [L_t, x_{t,a}^\top]^\top, \forall a \in \mathcal{D}_t$.
  if $t = 1$ **then**
    Choose the first actions $a_t \in \mathcal{D}_t$ (at start first action is pulled)
  else
    for $a \in \mathcal{D}_t$ **do**
      $k_{t,a} \leftarrow [k(\tilde{x}_{t,a}, \tilde{x}_{1,a_1}), k(\tilde{x}_{t,a}, \tilde{x}_{2,a_2}), \cdots, k(\tilde{x}_{t,a}, \tilde{x}_{t-1,a_{t-1}})]^\top$
      $\boldsymbol{K}_t \leftarrow$ kernel matrix of $(\tilde{x}_{1,a_1}, \cdots, \tilde{x}_{t-1,a_{t-1}})$
      $p_{t,a} \leftarrow k_{t,a}^\top [\boldsymbol{K}_t + \lambda \boldsymbol{I}]^{-1} y_{t-1} + \alpha \sqrt{k(\tilde{x}_{a,t}, \tilde{x}_{a,t}) - k_{t,a}^T [\boldsymbol{K}_t + \lambda \boldsymbol{I}]^{-1} k_{t,a}}$
    end for
    Choose action $a_t = \arg\max_{a \in \mathcal{D}_t} p_{t,a}$ with ties broken arbitrarily.
  end if
  Observe nominal reward r_{t,a_t}.
  $y_t \leftarrow [r_{1,a_1}, r_{2,a_2}, \cdots, r_{t,a_t}]^\top$
end for

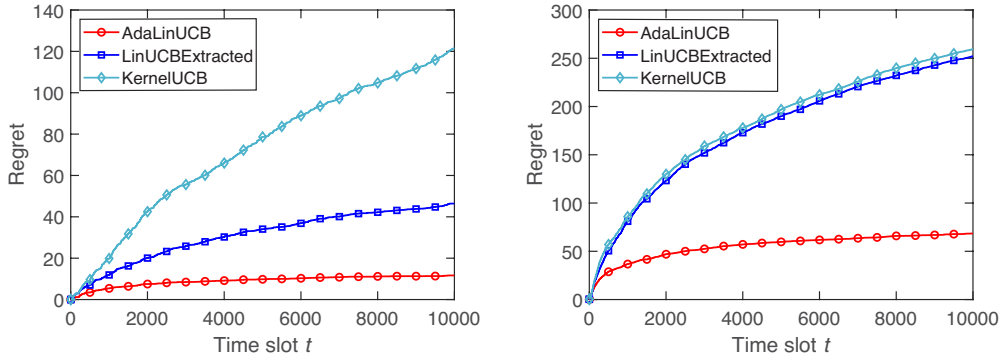
is at least 70 times longer than the time to run AdaLinUCB algorithm. In addition, when the time horizon is even larger, the time to run KernelUCB can be prohibitively long. This is because KernelUCB needs more computation with more existing data samples. As a result, KernelUCB is not applicable for applications with large number of data samples in practice.

4.8.5.4. *More for experiments on Yahoo! Today Module.* We also test the performance of the algorithms using the data from Yahoo! Today Module. This dataset contains over 4 million user visits to the Today module in a ten-day period in May 2009 [**48**]. To evaluate contextual bandits using offline data, the experiment uses the unbiased offline evaluation protocol proposed in [**120**].

In Yahoo! Today Module, for each user visit, there are 10 candidate articles to be selected. The candidate articles are updated in a timely manner and are different for different time slots. Further, both the user and each of the candidate articles are associated with a 6-dimensional feature vector, which are generated by a conjoint analysis with a bilinear model [**124**].

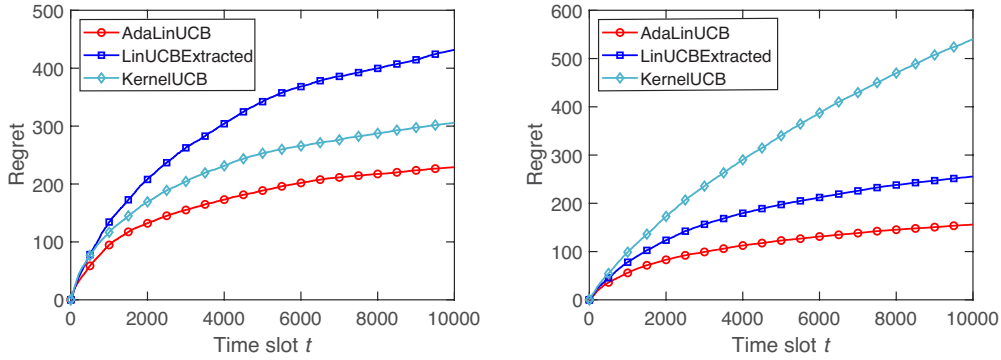
For the variation factor, we use a real trace - the sales of a popular store . It includes everyday turnover in two years [**122**]. The normalized variation factor variation is demonstrated in Fig. 4.9.

Similarly to the experiments in Fig. 4.4, Fig. 4.5 and Fig. 4.6, we have evaluated the impact of of $l^{(-)}$ and $l^{(+)}$ in this data set in Fig. 4.9. We can see that the impact of threshold values for



(a) Binary-valued L_t with $\rho = 0.9$. ($\epsilon_0 = \epsilon_1 = 0$.)

(b) Binary-valued L_t with $\rho = 0.5$. ($\epsilon_0 = \epsilon_1 = 0$.)



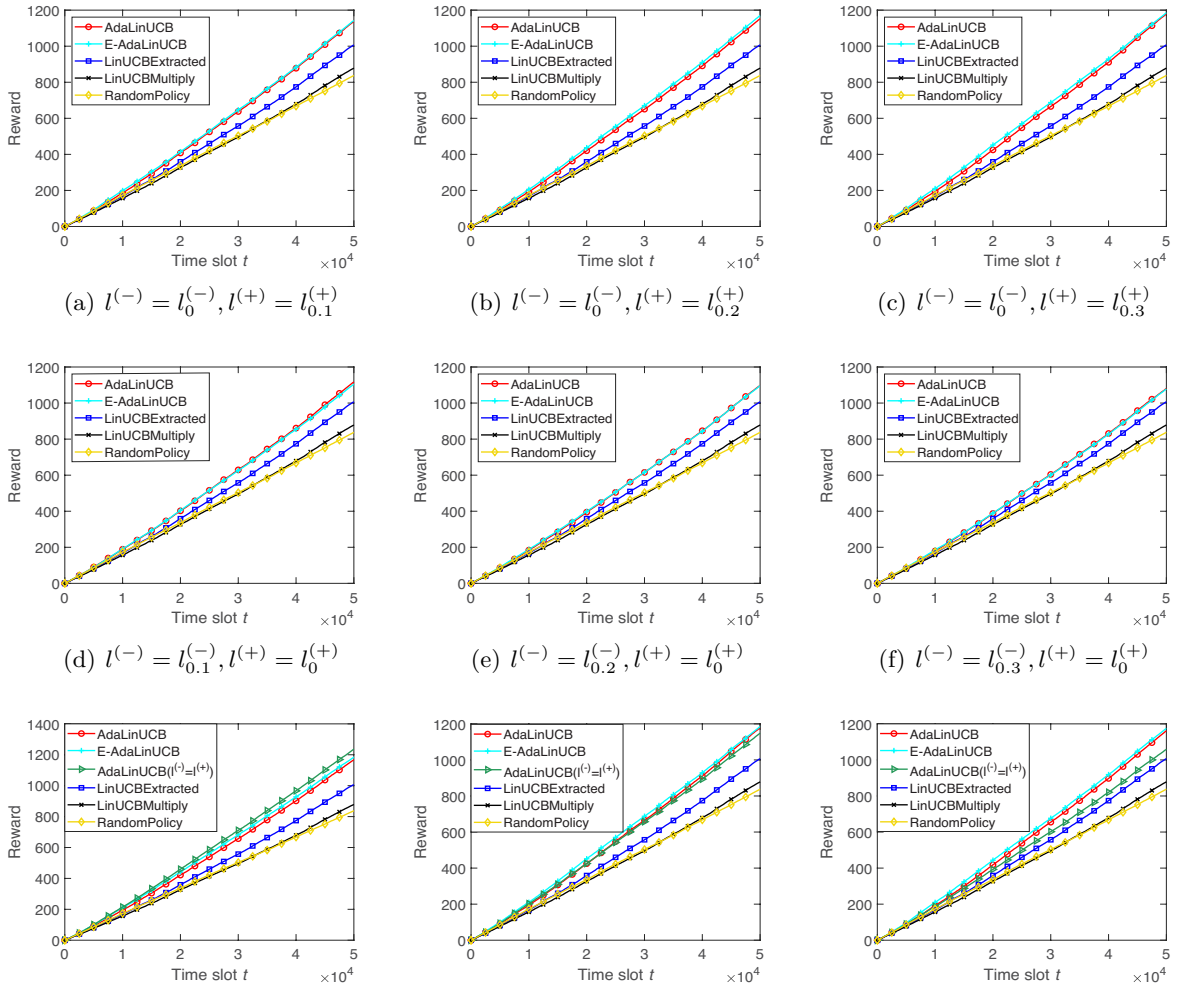
(c) Binary-valued L_t with $\rho = 0.1$. ($\epsilon_0 = \epsilon_1 = 0$.)

(d) Beta distributed variation factor; AdaLinUCB with $l^{(-)} = 0, l^{(+)} = l_0^{(+)}$.

FIGURE 4.7. Performance Comparison with KernelUCB.

experiments on this real-world dataset is insignificant (when they are changing in a relatively large appropriate range) and the rewards of AdaLinUCB and E-AdaLinUCB are always higher than that of LinUCBExtracted and LinUCBMultiple.

$$\tilde{\mathbf{R}}_{\text{total}}(T) \leq \frac{\mathbb{E}\left[L_t | L_t \leq l^{(-)}\right] O\left((\log T)^2\right)}{\Delta_{\min}} + \frac{\mathbb{E}\left[L_t | L_t > l^{(-)}\right] O\left(\log T\right)}{\Delta_{\min}}$$



(a) $l^{(-)} = l_0^{(-)}, l^{(+)} = l_{0.1}^{(+)}$ (b) $l^{(-)} = l_0^{(-)}, l^{(+)} = l_{0.2}^{(+)}$ (c) $l^{(-)} = l_0^{(-)}, l^{(+)} = l_{0.3}^{(+)}$

(d) $l^{(-)} = l_{0.1}^{(-)}, l^{(+)} = l_0^{(+)}$ (e) $l^{(-)} = l_{0.2}^{(-)}, l^{(+)} = l_0^{(+)}$ (f) $l^{(-)} = l_{0.3}^{(-)}, l^{(+)} = l_0^{(+)}$

(g) AdaLinUCB: $l^{(-)} = l_{0.05}^{(-)}, l^{(+)} = l_{0.3}^{(+)}$; AdaLinUCB ($l^{(-)} = l^{(+)}$):$l^{(-)} = l^{(+)} = 0.4$ (h) AdaLinUCB: $l^{(-)} = l_0^{(-)}, l^{(+)} = l_{0.3}^{(+)}$; AdaLinUCB ($l^{(-)} = l^{(+)}$):$l^{(-)} = l^{(+)} = 0.5$ (i) AdaLinUCB: $l^{(-)} = l_{0.1}^{(-)}, l^{(+)} = l_{0.3}^{(+)}$; AdaLinUCB ($l^{(-)} = l^{(+)}$): $l^{(-)} = l^{(+)} = 0.6$

FIGURE 4.8. Performance comparison with different $l^{(-)}$ and $l^{(+)}$ values on Yahoo! Today Module.

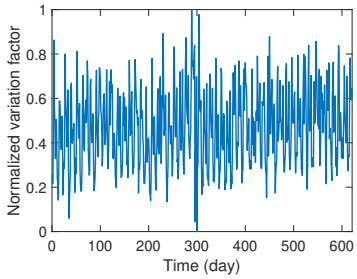


FIGURE 4.9. Normalized variation factor demonstration.

CHAPTER 5

Opportunistic Learning for Episodic Reinforcement Learning

5.1. Introduction

Recently, reinforcement learning (RL) has shown spectacular success. By experimenting, computers can learn how to perform tasks that no programmer could teach them autonomously. However, the effectiveness of these approaches varies significantly depending on the application domain. In general, there is a need for reinforcement learning algorithms that deliver both strong empirical performance and solid theoretical guarantees. These objectives cannot be achieved without efficient environmental exploration, which has been extensively studied in episodic RL.

In episodic RL, an agent interacts with the environment in a series of episodes while aiming to maximize the total reward accumulated over time [**20**, **21**]. This learning process presents a fundamental trade-off: Should the agent explore insufficiently-understood states and actions to gain new knowledge and achieve better long-term performance, or should it exploit its existing information to maximize short-term rewards? The existing algorithms focus on striking an appropriate balance between these choices, under the implicit assumption that the exploration cost remains consistent over time. However, in numerous application scenarios, the exploration cost is both time-varying and situation-dependent. These scenarios provide an opportunity to explore more when the exploration cost is relatively low, and exploit more when it's high, thereby adaptively balancing the exploration-exploitation trade-off to achieve better performance. Consider the following motivating examples:

Motivating scenario 1: Return variation in games. In a game or gambling machine, certain rounds may offer players special multipliers (2×, 4×, ..., etc.) on their rewards. Players can win a large number of points by getting lucky and landing large prizes supplemented by substantial multipliers. Consequently, when a player is given a large multiplier, they should ideally make the move they believe to be best. This conservative approach is less risky, especially given that making

a "bad" move in this situation could result in a significant loss. Conversely, in rounds with no or small multipliers, attempting experimental actions will be less risky since the regret of trying a suboptimal move will be relatively low.

Motivating scenario 2: Value variation in sequential recommendations. In e-commerce's sequential recommender systems, the system continuously suggests candidate products to users with the goal of maximizing the total click-through rate (i.e., the probability that a user accepts the recommendation), which is based on users' preferences and browsing history. It is important to note that the real monetary return of a recommendation (if accepted) can vary based on other factors such as the purchasing power or loyalty level of users (e.g., diamond vs silver status). Because the ultimate goal is to maximize the overall monetary reward, intuitively, when the monetary return of a recommendation (if accepted) is low, the monetary regret of suggesting suboptimal products is also low, resulting in a low exploration cost. Conversely, high returns lead to high regret and a correspondingly high exploration cost.

Opportunistic reinforcement learning. Motivated by these examples, we propose and study *opportunistic episodic reinforcement learning*, a new paradigm of reinforcement learning problems where the regret of executing a suboptimal action is influenced by a varying cost referred to as the *variation factor*, associated with environmental conditions. When the variation factor is low, the cost/regret of selecting a suboptimal action is likewise low, and vice versa. Therefore, intuitively, we should explore more when the variation factor is low and exploit more when the variation factor is high. As the name suggests, opportunistic RL allows us to leverage the dynamics of the variation factor to reduce regret.

Contributions. In this work, we propose the OppUCRL2 algorithm for opportunistic learning in episodic RL that introduces variation factor-awareness to balance the exploration-exploitation trade-off. The OppUCRL2 can significantly outperform the UCRL2 [**22**] in the simulation and have the same theoretical guarantee with respect to the regret. The opportunistic RL concept is also easy to generalize for other reinforcement algorithms. To demonstrate it, we design the OppPSRL algorithm based on PSRL [**23**]. It also achieves better performance compared with the original version in the simulation. To the best of our knowledge, this is the first work proposing and studying the concept of opportunistic reinforcement learning. We believe this work will serve

as a foundation for the opportunistic reinforcement learning concept and help further address the exploration-exploitation trade-off.

5.2. Related Work

Optimism in the face of uncertainty (OFU) is a popular paradigm for the exploration-exploitation trade-off in RL. Here, each pair of states and actions is assigned an optimism bonus. The agent then chooses a policy that is optimal under this "optimistic" model of the environment. To learn efficiently, the agent maintains control over its uncertainty by assigning a larger optimism bonus to potentially informative states and actions. This bonus can stimulate and guide the exploration process. Most OFU algorithms provide strong theoretical guarantees [**22**, **66**, **67**, **68**, **69**]. An alternate approach is inspired by Thompson sampling (TS) [**70**]. In RL, TS approaches [**71**] maintain a posterior distribution over the reward function and the transition kernel, then compute the optimal policy for a randomly sampled MDP from the posterior. One of the well-known TS algorithms in literature is the Posterior Sampling for Reinforcement Learning (PSRL) [**23**, **72**].

The opportunistic learning idea has been introduced in [**73**] for classic K-armed bandits and in [**4**] for context bandits. In reinforcement learning, the authors in [**125**] consider the case where each episode has a side context and propose the ORLC algorithm that can use the context information to estimate the dynamics of the environment but does not include the opportunistic concept, which distinguishes our work. To the best of our knowledge, no prior work has made a formal mathematical formulation and rigorous performance analysis for opportunistic reinforcement learning.

5.3. Problem Formulation

We consider an RL problem in an episodic finite-horizon Markov decision process (MDP), $M := \langle \mathcal{S}, \mathcal{A}, H, P, r \rangle$, where $\mathcal{S}$ is a finite state space with carnality $|\mathcal{S}| = S$, $\mathcal{A}$ is a finite action space with carnality $|\mathcal{A}| = A$, H is the horizon that represents the number of time steps in each episode, P is a state transition distribution such that $P(\cdot|s, a)$ dictates a distribution over state $\mathcal{S}$ if action a is taken for state s, and $r : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the deterministic reward function. For simplicity, we assume that the reward function r is known to the agent, but the transition distribution P is unknown.

In each episode of this MDP, an initial state $s_1 \in \mathcal{S}$ is chosen arbitrarily by the environment before it starts. For each step $h \in [H]$[1], the agent observes a state $s_h \in \mathcal{S}$, selects an action $a_h \in \mathcal{A}$, receives a reward $r(s_h, a_h)$ and then the state transits to next state $s_{h+1} \in \mathcal{S}$ that is drawn from the distribution $P(\cdot|s_h, a_h)$. The episode ends in state s_{H+1}.

A policy for an agent during the episode is expressed as a mapping $\pi : \mathcal{S} \times [H] \to \mathcal{A}$. We write $V_h^\pi : \mathcal{S} \to \mathbb{R}$ as the value function at step h under policy π. For a state $s \in \mathcal{S}$, $V_h^\pi(s)$ is the expected return (i.e., the sum of rewards) received under policy π, starting from $s = s_h \in \mathcal{S}$, i.e., $V_h^\pi(s) := \mathbb{E}\left[\sum_{i=h}^{H} r(s_i, \pi(s_i, i))\middle| s_h = s\right]$. Because the action space, state space, and horizon are finite, and the reward function is deterministic, there always exists an optimal policy π^* that attains the best value $V_h^*(s) = sup_\pi V_h^\pi(s)$ for all $s \in \mathcal{S}$ and $h \in [H]$. For an episode with initial state s_1, the quality of a policy π is measured by the regret that is the gap between the value function at step 1 under policy π and that under the optimal policy, i.e., $V_1^*(s_1) - V_1^\pi(s_1)$. The goal of the classic RL problem is to consider an RL agent interacts with the environment (MDP M) for K episodes $k \in [K]$ in a sequential manner and find the optimal policy.

Next, we introduce the **opportunistic reinforcement learning** in an episodic finite-horizon MDP. For each episode $k \in [K]$, let $L_k \geq 0$ be an external **variation factor** and not change during the episode. We assume L_k is independent of the MDP M for $k \in [K]$. To distinguish different episodes, we use $s_{k,h}$, $a_{k,h}$, $r_{k,h}$ to denote the state, action and reward in step h of episode k. The **expected actual return** for the episode k is defined as $\mathbb{E}[L_k V_1^\pi(s_{k,1})]$ if the initial state is $s_{k,1}$ and the policy of agent is π. Before the k-th episode, the agent can observe the initial state $s_{k,1}$ and the current value of L_k. Based on the policy π_k that the agent selected, the expected actual return that the learner receives is $\mathbb{E}[L_k V_1^\pi(s_{k,1}])$.

This model captures the essence of the opportunistic RL paradigm for the motivating scenarios in the introduction. In the opportunistic RL model, we notice that the optimal policy that maximizes $\mathbb{E}[L_k V_h^{\pi_k}(s_{k,1})]$ for each episode $k \in [K]$ does not change over episodes and is same as the optimal policy $\pi*$ in the standard RL problem for an MDP M. So, the best expected actual return for an episode k is $\mathbb{E}[L_k V_h^*(s_{k,1})]$.

[1]We write $[n]$ for $i \in \mathbb{N}, 1 \leq i \leq n$

The goal is to minimize the **actual total regret** for K episodes in terms of the expected actual return. Particularly, we define the actual total regret in opportunistic RL problems over K episodes regarding the expected actual return as:

$$Regret(K) := \sum_{k=1}^{K} \mathbb{E}\left[L_k V_1^*(s_{k,1}) - L_k V_1^{\pi_k}(s_{k,1})\right] \tag{5.1}$$

In a special case, equation (5.1) has an equivalent form: when L_k is i.i.d. over the episodes with mean value $\bar{L}$, the total regret regarding actual reward is $Regret(K) = \bar{L}\sum_{k=1}^{K} V_1^*(s_{k,1}) - \sum_{k=1}^{K} \mathbb{E}[L_k V_1^{\pi_k}(s_{k,1})]$. Note that in general, it is likely that $\mathbb{E}[L_k V_1^{\pi_k}(s_{k,1})] \neq \bar{L}\mathbb{E}[V_1^{\pi_k}(s_{k,1})]$, because the policy π_k can depend on L_k.

5.4. Opportunistic Reinforcement Learning Algorithm

In this section, we propose two opportunistic algorithms that are designed based on optimism in the face of uncertainty and posterior sampling, respectively.

We first introduce the OppUCRL2 algorithm, an opportunistic variant of UCRL2 [**22**].

In Alg.10, $\delta \in (0, 1]$ is a hyper-parameter, and $\tilde{L}_k$ is the normalized variation factor, defined as,

$$\tilde{L}_k = \frac{[L_k]_{l_{\min}}^{l_{\max}} - l_{\min}}{l_{\max} - l_{\min}} \tag{5.2}$$

where $l_{\min}$ and $l_{\max}$ are, respectively, the lower and upper thresholds for truncating the variation factor level, and $[L_k]_{l_{\min}}^{l_{\max}} = \max\{l_{\min}, \min\{L_k, l_{\max}\}\}$. The variation factor normalization restricts the impact of the variation factor term in the confidence bounds, which avoids under or over-explorations. We note that the normalized variation factor $\tilde{L}_k$ is only employed in the algorithm itself. Indeed, the regret depends on the real variation factor L_k and not $\tilde{L}_k$.

In the initialization, $N(s, a)$ and $N(s, a, s')$ are the counts for state-action pair (s, a) played and tuple (s, a, s) happened up to current episode. $t_k = H(k-1)$ is the start time of the episode k. Before the start of the k-th episode, the algorithm observes the variation factor L_k and normalize it by Eq.5.2 in Line 4. The empirical estimate $\hat{P}_k(\cdot|s, a)$ of $P(\cdot|s, a)$ is calculated by all historical transitions observed so far in Line 6. The width of the high probability confidence regions of $\hat{P}(\cdot|s, a)$ is estimated by Hoeffding's inequality and normalized variation factor $\tilde{L}_k$ in line 7. Then, a plausible MDP set $\mathcal{M}_k$ is created in line 8 that consists of finite-horizon MDP M' with same

known reward function r and the transition probability $P'(\cdot|s,a)$ in the high probability confidence regions of $\hat{P}(\cdot|s,a)$ with width $d_k(s,a)$ for all state-action pairs.

Algorithm 10 OppUCRL2

1: **Input**: l_{min}, l_{max}, δ
2: **Initialization**: $N(s,a,s') = 0$, $N(s,a) = 0$ $\forall, s \in \mathcal{S}, a \in \mathcal{A}, s' \in \mathcal{S}$
3: **for** episode $k = 1, 2, \ldots, K$ **do**
4: Observe L_k and calculate $\tilde{L}_k$ by Eq. 5.2
5: $t_k = H(k-1)$
6: $\hat{P}_k(s'|s,a) = \frac{N(s,a,s')}{N(s,a)}$
7: $d_k(s,a) = \sqrt{\frac{2S(1-\tilde{L}_k)\log(2SAt_k/\delta)}{\max\{1,N(s,a)\}}}$
8: $\mathcal{M}_k := \{M' : \left\| P'(.|s,a) - \hat{P}_k(.|s,a) \right\|_1 \leq d_k(s,a)\ \forall (s,a) \in \mathcal{S} \times \mathcal{A}\}$
9: $\pi_k, \tilde{M}_k \leftarrow ExtendedValueIteration(\mathcal{M}_k)$
10: **for** time step $h = 1, \ldots, H$ **do**
11: $a_{k,h} = \pi_k(s_{k,h})$.
12: Observe $r_{k,h}$ and $s_{k,h+1} \sim P(\cdot|s_{k,h}, a_{k,h})$.
13: $N(s_{k,h}, a_{k,h}, s_{k,h+1}) \leftarrow N(s_{k,h}, a_{k,h}, s_{k,h+1}) + 1$
14: $N(s_{k,h}, a_{k,h}) \leftarrow N(s_{k,h}, a_{k,h}) + 1$
15: **end for**
16: **end for**

Next, in line 9, Alg.10 calls a subroutine Finite Horizon Extended Value Iteration Alg. 11 that returns an optimistic MDP $\tilde{M}_k$ with the best achievable reward from $\mathcal{M}_k$ and the optimistic policy π_k. The idea behind finite horizon extended value iteration is the same as [**68**, **126**]. Last, the policy π_k executes throughout the episode k adn updates the counts $N(s,a,s')$ and $N(s,a)$.

Alg.11 Finite Horizon Extended Value Iteration is used as a subroutine for Alg.10 OppUCRL2. The input of Alg.10 is an MDP set. The output is the optimistic MDP $\tilde{M}_k$ with the best achievable reward from $\mathcal{M}_k$ and the optimistic policy π_k in line 9. In practice, we define the value function in finite horizon MDP M' under policy π as $V_h^{M'(\pi)}(s) := \mathbb{E}_{p(\cdot|s,a)\sim M'}[\sum_{i=h}^{H} r(s_i, \pi(s_i,i))|s_h = s]$. Then the optimistic MDP $\tilde{M}_k$ and optimistic policy π_k are $\tilde{M}_k, \pi_k = \arg\max_{M' \in \mathcal{M}_k, \pi} V_1^{M'(\pi)}(s)$ for all $s \in \mathcal{S}$. The idea behind finite horizon extended value iteration is the same as [**68**, **126**], putting as much transition probability as possible to the state with maximal value at the expense of transition probabilities to states with small values. Then, in order to make $\tilde{P}$ correspond to a probability distribution again, the transition probabilities with small values are reproduced iteratively with respect to the constraint $d(s,a)$. This implies that extended value iteration solves a

Algorithm 11 Finite Horizon Extended Value Iteration.

1: **Input**: MDP set $\mathcal{M}$
2: **Initialize** $V_{H+1}(s) = 0$ for all $s \in \mathcal{S}$
3: **for** $h = H, H-1, \ldots, 1$ **do**
4: Sort the states in $\mathcal{S}$ in the descending order w.r.t. their values: Let $\mathcal{S} = \{s'_1, s'_2, ..., s'_S\}$ such that $V_{h+1}(s'_1) \geq V_{h+1}(s'_2) \geq \cdots \geq V_{h+1}(s'_S)$
5: **for** $(s, a) \in \mathcal{S} \times \mathcal{A}$ **do**
6: $\tilde{P}(s'_1|s, a) = \min\left\{1, \hat{P}(s'_1|s, a) + \frac{d(s,a)}{2}\right\}$
7: $\tilde{P}(s'_i|s, a) = \hat{P}(s'_i|s, a)$ for all $1 < i \leq S$
8: Set $j = S$
9: **while** $\sum_{s'_i \in \mathcal{S}} \tilde{P}_k(s'_i|s, a) > 1$ **do**
10: $\tilde{P}(s'_j|s, a) = \max\{0, 1 - \sum_{s'_i \neq s'_j} \tilde{P}(s'_i|s, a)\}$
11: $j = j - 1$
12: **end while**
13: $Q_h(s, a) = r(s, a) + \sum_{s' \in \mathcal{S}} \tilde{P}(s'|s, a) V_{h+1}(s')$
14: **end for**
15: $V_h(s) = \max_{a \in \mathcal{A}} Q_h(s, a)$
16: **end for**
17: $\pi(s, h) = \arg\max_{a \in \mathcal{A}} Q_h(s, a)$ for all $s \in \mathcal{S}$, $h \in [H]$
18: **Output:** MDP with transition probabilities $\tilde{P}$, and optimal policy π

linear optimization problem over the convex polytope constructed by the set of transition probabilities satisfying conditions and $d(s, a)$.

In general, Alg.10 explores more when the variation factor is relatively low, and exploits more when the variation factor is relatively high. To see this, note that $d_k(s, a)$ in line 7 is the adaptive width of the confidence region modulated by $\tilde{L}_k$ for MDP set $\mathcal{M}_k$, which determines the level of exploration. For example, when L_k is at its lowest level with $L_k \leq l_{min}$, $\tilde{L}_t = 0$, and the width of confident region d^k is the same as that of the UCRL2 algorithm, and then the algorithm learns the policy in the same way as the conventional UCRL2. At the other extreme, when $\tilde{L}_k = 1$, i.e., $L_k \geq l_{max}$, the width $d_k = 0$, that is, when the variation factor is at its highest level, the algorithm purely exploits the existing knowledge and selects the best policy. With the exploitation of variation factor awareness capabilities and given that the actual regret is scaled with the variation factor level, OppUCRL2 could achieve a lower regret than the original UCRL2.

Similarly, we also can generalize the opportunistic RL concept into the sampling-based algorithm, Alg. [**?**] OppPSRL, which is a variant of Posterior Sampling for Reinforcement Learning

(PSRL) [**72**]. In each episode, PSRL samples a single MDP from the plausible MDP set and then selects a policy that has the maximum value for that MDP.

Algorithm 12 OppPSRL

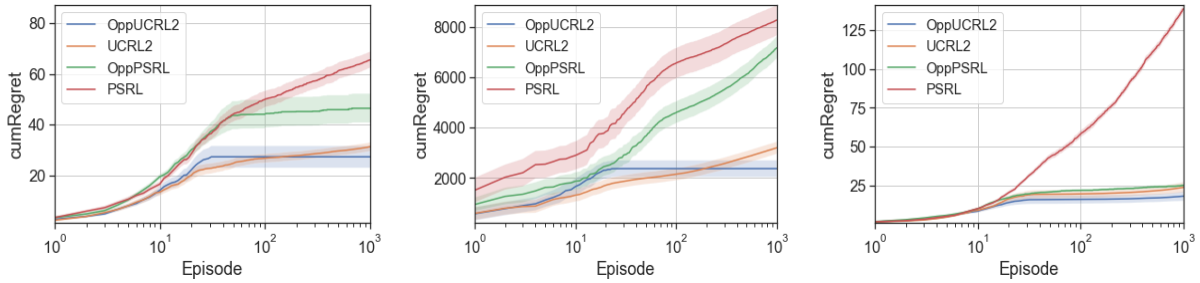
1: **Input**: prior distribution $\phi(\boldsymbol{\alpha}_{0,1})$ of M
2: **for** episode $k = 1, 2, \ldots, K$ **do**
3: Observe L_k and calculate $\tilde{L}_k$ by Eq.5.2
4: $\boldsymbol{\alpha}_k = \tilde{L}_k \boldsymbol{\alpha}_{k-1,1}$
5: Sample MDP $M_k \sim \phi(\cdot|\boldsymbol{\alpha}_k)$
6: Compute $\pi_k = arg\,max_\pi V_1^{M_k(\pi)}$
7: **for** time step $h = 1, \ldots, H$ **do**
8: $a_{k,h} = \pi_k(s_{k,h})$.
9: Observe $r_{k,h}$ and $s_{k,h+1} \sim P(\cdot|s_{k,h}, a_{k,h})$.
10: Update the parameters $\boldsymbol{\alpha}_{k,h}$ of posterior distribution by $(s_{k,h}, a_{k,h}, r_{k,h}, s_{k,h+1})$.
11: **end for**
12: **end for**

Inspired by the opportunistic learning idea, we propose Alg.12 OppPSRL. The input is a prior distribution of the MDP. In general, we can formulate the state transition distribution $P(\cdot|s, a)$ as a Dirichlet distribution ϕ with parameters $\boldsymbol{\alpha}$. At the beginning of each episode, Alg. 12 calculates the normalized variation factor $\tilde{L}_t$ in line 3. Then it uses $\tilde{L}_k$ to rescale the parameter $\boldsymbol{\alpha_k}$. Next, an MDP M_k is sampled from the scaled distribution. In line 6, it computes the policy π_k that has the maximum value for the MDP. Finally, the policy π_k is executed throughout the episode k and the posterior distribution is updated by the new observations.

The core step of Alg. 12 is Line 4. Intuitively, when $\tilde{L}_k$ is small, it can decrease the value of $\boldsymbol{\alpha_k}$ and the corresponding Dirichlet distribution is more concentrated, then the sampled MDP in Line 5 is similar to the empirical MDP with high probability, so the policy π_k in Line 6 is more conservative and less exploratory. When $\tilde{L}_k$ is larger, the distribution flattens, and it provides the opportunity for the agent to explore new MDP and try under-explored actions.

5.5. Regret Analysis for OppUCRL2

In this section, we present an upper bound on the regret of OppUCRL2. We study a simple case with a periodic square wave variation factor. Specifically, we assume that the variation factor for an episode k is $L_k = \epsilon_0$ if the episode index k is even, and $L_k = \epsilon_1$ if k is odd. Because we use a sophisticated variation factor-aware regret expression as described in Eq. 5.1 for the opportunistic



(a) *River Swim* (b) *Cliff Walking* (c) *Frozen Lake*

FIGURE 5.1. Regret under binary variation factor scenarios.

learning that is different from the classical regret definition, in order to compare OppUCRL2 and original UCRL2 algorithm fairly, we should derive the regret bounds for both them based on Eq. 5.1. Following the same logic as [**22**, **127**], we can get Theorem 6 and Theorem 5.5.2 that show UCRL2 and OppUCRL2 can achieve the same regret bound $\tilde{O}(HS\sqrt{AT})$ in the periodic square wave variation factor case.

THEOREM 5.5.1 (Regret Bound for UCRL2 under Periodic Square Wave Variation Factor). *For a finite horizon MDP,* $M := \langle \mathcal{S}, \mathcal{A}, H, P, r \rangle$*, and* $L_k = \epsilon_0$ *if the episode index* k *is even, and* $L_k = 1 - \epsilon_1$ *if* k *is odd, consider a parameter* δ*, then the regret of UCRL2 is bounded with a probability at least* $1 - \delta$ *by,*

$$Regret(K) = \tilde{O}(HS\sqrt{CAT})$$

where $C = \log(2SAT/\delta)$.

THEOREM 5.5.2 (Regret Bound for OppUCRL2 under Periodic Square Wave Variation Factor). *For a finite horizon MDP,* $M := \langle \mathcal{S}, \mathcal{A}, H, P, r \rangle$*, and* $L_k = \epsilon_0$ *if the episode index* k *is even, and* $L_k = 1 - \epsilon_1$ *if* k *is odd, consider a parameter* δ*, then the regret of OppUCRL2 is bounded with a probability at least* $1 - \delta$ *by,*

$$Regret(K) = \tilde{O}(HS\sqrt{CAT})$$

where $C = \log(2SAT/\delta)$.

Next, we introduce the proof of the theorems in the main paper. According to Theorem 2 in [**22**] and Theorem 1 in [**127**], we have the following lemma that shows the regret bound for UCRL2 in finite horizon MDP.

LEMMA 6 (Regret Bound for UCRL2 in finite horizon MDP). *For a finite horizon MDP, $M := \langle \mathcal{S}, \mathcal{A}, H, P, r \rangle$, consider a parameter δ, the regret of UCRL2 is bounded with a probability at least $1-\delta$ by,*

$$Regret(K) = \tilde{O}(HS\sqrt{CAT})$$

where $C = \log(2SAT/\delta)$.

Proof of Theorem 1.

PROOF. For the periodic square wave variation factor case, we can categorize the episodes into two groups, and then analyze the regret independently, which can still guarantee an upper bound for the regret because the variation factor is independent of the MDP and UCRL2 algorithm. Specifically, from Eq. 5.1, we have

$$\begin{aligned} Regret(K) &= \sum_{k\in[K]} \mathbb{E}\left[L_k V_1^*(s_{k,1}) - L_k V_1^{\pi_k}(s_{k,1})\right] \\ &= \sum_{k\in[K],\text{k is odd}} \mathbb{E}\left[L_k V_1^*(s_{k,1}) - L_k V_1^{\pi_k}(s_{k,1})\right] + \\ &\quad \sum_{k\in[K],\text{k is even}} \mathbb{E}\left[L_k V_1^*(s_{k,1}) - L_k V_1^{\pi_k}(s_{k,1})\right] \\ &= \epsilon_0 * Regret_{original,odd}(K) + \\ &\quad (1-\epsilon_1) * Regret_{original,even}(K) \end{aligned} \tag{5.3}$$

So, based on the union bound and Lemma 1, we can get bound in Theorem 1. □

Proof of Theorem 2.

PROOF. In order to bound the regret of OppUCRL2, we can still do the same decomposition as Eq. 5.3. The difference is that the exploration in OppUCRL2 is related to the variation factor, so we cannot directly apply lemma 1 for analysis. However, we notice that in the analysis of UCRL2 in finite horizon MDPs, the regret bound is mainly dominated by the time of visits for all state-action pairs. In order to get the upper bound of the regret of OppUCRL2 in periodic square wave variation factor, we can regard it as two independent algorithms with different exploration

parameters in odd and even episodes to get the upper bound of regret, because the time of visits for each state-action pairs in OppUCRL2 is at least same as that in two independent UCRL2 cases and the difference only affects the constant coefficient in the bound. So, OppUCRL2 can achieve the bound shown in Theorem 5.5.2. □

Theorem 6 and Theorem 5.5.2 show that UCRL2 and OppUCRL2 in the periodic square wave variation factor can achieve the same regret bound $\tilde{O}(HS\sqrt{AT})$.

5.6. Experimental Evaluation

In this section, we evaluate the empirical performance of OppUCRL2 and OppPSRL compared to the original UCRL2 and PSRL algorithms. We use three classic examples of the OpenAI Gym, namely River Swim, Cliff Walking and Frozen Lake that represent three different test cases [**128**]: undiscounted reward in a stochastic environment, undiscounted reward deterministic environment, and discounted reward in a deterministic environment.

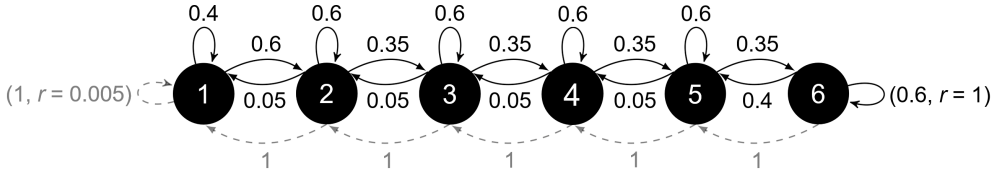


FIGURE 5.2. *River Swim* - consisting of six states arranged in a chain and two actions. Continuous and dotted arrows represent the MDP under the actions "right" and "left", respectively. The agent always starts in state 1. Swimming left (with the current) is always successful, but swimming right (against the current) often fails. The agent receives a small reward for reaching the starting state, but the optimal policy is to attempt to swim right and receive a much larger reward. We set the horizon $H = 15$ and length of the chain $S = 6$.

The stochastic and deterministic describe state transition distribution. The River Swim and Cliff Walking RL environments can be formulated as undiscounted, episodic MDPs, while Frozen Lake is a discounted, episodic task with a discount factor $\gamma = 0.95$. We report the results for the average of 20 simulations with different seeds while showing 95% confidence interval. We use the same scaling factors for both algorithms, chosen experimentally for each environment. We compare all algorithms with their best input precision hyper-parameters obtained by grid search.

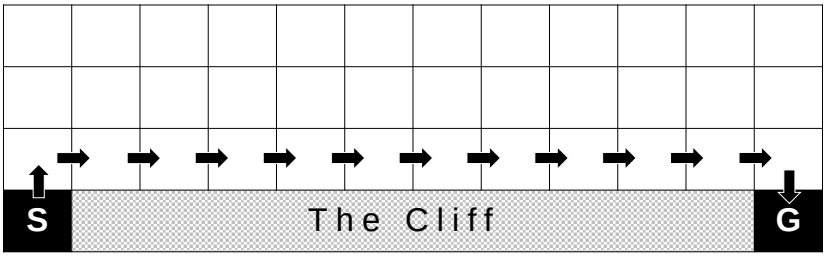


FIGURE 5.3. *Cliff Walking* - consisting of a grid. The start state is in the left lower corner while the goal state is in the right lower corner of the grid. The possible actions causing movement are UP, DOWN, RIGHT, and LEFT. Each transition incurs -1 reward, except for stepping into the gray region marked "The Cliff", which incurs -100 reward and a reset to the start. An episode terminates when the agent reaches the goal or the episode time expires (reach the horizon). A suboptimal policy can be thought of as avoiding moving closely to "The Cliff" region. The optimal policy is the shortest path to avoiding "The Cliff" region as shown in the figure. We set the horizon $H = 50$ and the shape of grid as 4×12.

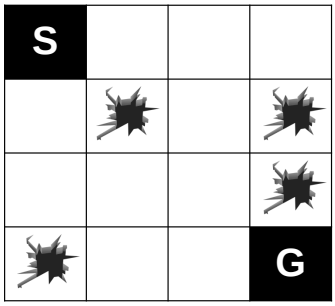
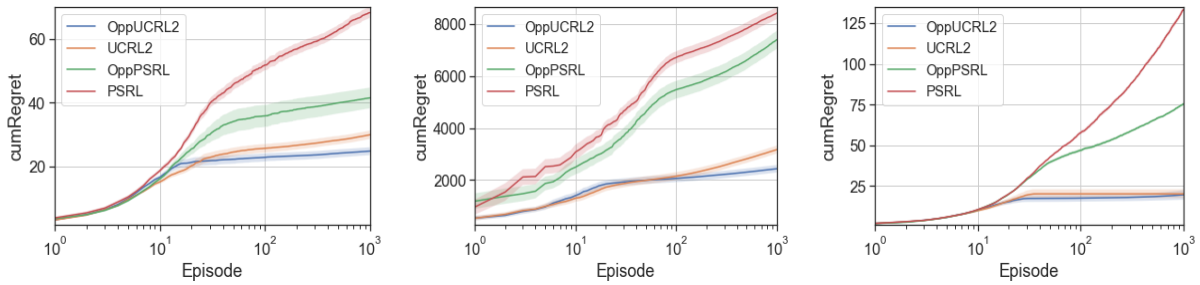


FIGURE 5.4. *Frozen Lake* - consisting of a grid world representing a frozen lake. The water is mostly frozen, but there are a few holes where the ice has melted. Thus, some tiles of the grid are walkable, and others lead to the agent falling into the water. The agent is rewarded for finding a walkable path to a goal tile. The episode ends when it reaches the goal, fall in a hole or the episode time expires. The agent receives a reward of 1 if it reaches the goal, -1 if it falls in a hole and zero otherwise.We set the horizon $H = 20$ and the shape of grid as 4×4.

We first introduce the result under a random binary-valued variation factor. We assume that the variation factor L_k is i.i.d. over the episodes, with $L_k \in \{\epsilon_0, 1 - \epsilon_1\}$, where $\epsilon_0, \epsilon_1 \geq 0$ and $\epsilon_0 < 1 - \epsilon_1$. Let ρ denote the probability that the variation factor is low, i.e., $\mathbb{P}\{L_k = \epsilon_0\} = \rho$. Fig. 5.1 shows the regret for different algorithms under random binary-value variation factor with $\epsilon_0 = \epsilon_1 = 0$ and $\rho = 0.5$.

Opportunistic RL algorithms outperform the corresponding original RL algorithms across every environment by significantly reducing regret. More significantly, for River Swim, Cliff Walking and



(a) *River Swim* (b) *Cliff Walking* (c) *Frozen Lake*

FIGURE 5.5. Regret under Beta *variation factor* scenarios

Frozen Lake, at the end of the 10^3-th episode, OppUCRL2 reduces the regret by 12.7%, 25.9% and 23.7% respectively, compared with UCRL2. OppPSRL reduces the regret by 29.1%, 13.3% and 81.9%, respectively, compared with PSRL. We also notice that OppUCRL2 achieves $O(1)$ regret converging to the optimal policy in a constant time. This is because it pushes most exploration moves to the episodes where the variation factor is equal to zero. As a result, the exploration cost is negligible. Although OppUCRL2 largely outperforms UCRL2, it may have higher regret at the beginning, especially in environments with less deterministic behaviour such as River Swim and Cliff Walking. OppUCRL2 emphasizes the main insight of the exploration-exploitation trade-off: we may sacrifice some short-term rewards to improve future performance. This observation, combined with the constant-time optimal regret, demonstrates OppUCRL2's capability to learn and adapt to the environment's dynamics over time.

We also test these algorithms in the continuous variation factor case and find the opportunistic version of the algorithms also has a better performance. We assume that the variation factor L_k is i.i.d. over episodes and sampled from a Beta distribution, i.e., $L_k \sim Beta(2, 2)$. Figure 5.5 shows the regrets for different algorithms and environments. Here, we define the lower threshold l_{min} such that $\mathbb{P}(L_k \leq l_{min}) = \rho$ where $\rho = 0.05$, and the upper threshold l_{max} such that $\mathbb{P}(L_k \geq l_{max}) = \rho$.

For River Swim, Cliff Walking and Frozen Lake, at the end of the 10^3-th episode, OppUCRL2 reduces regret by 17.1%, 23.3% and 2.7% respectively, compared with UCRL2. OppPSRL reduces regret by 39.2%, 12.2% and 43.4%, respectively, compared with PSRL. For OppUCRL2, we also see similar trends with previous experiments. However, with the beta variation factor, the OppUCRL2 algorithm does not achieve a constant-time regret. This is due to the fact that the variation factor does not vary radically between 0 and 1, and the exploration carried out in the low variation factor

episode usually does not have a zero variation factor, thus, generating an extra overhead compared to the previous experimental case.

5.7. Discussion

In this section, we discuss the limitations of our work and possible future improvements.

Weakly communicating MDPs: In this paper, we focused on the finite horizon MDPs setting. While some previous approaches to exploration provide regret bounds for the more general setting of weakly communicating MDPs [**22**, **67**], we believe our analysis can be extended to this more general case using existing techniques such as the "doubling trick" [**22**].

Computational and statistical efficiency: Our proposed algorithm is computationally tractable. In each episode, it performs optimistic value iteration, which has a computational cost on the same order as solving a known MDP. Furthermore, our regret bounds guarantee the statistical efficiency of the algorithm with high probability.

Theoretical Regret Bound: In our current work, we show th at both OppUCRL2 and UCRL2 can achieve the same bound in the case of a periodic square wave variation factor. However, in simulations, the opportunistic version yields significantly better results. This suggests that the bound for OppUCRL2 is not tight, at least under some circumstances. Existing literature on finite horizon MDP analysis considers all state-action pairs together to establish an upper bound for regret, which overlooks differences in exploration strength. Therefore, to achieve a better bound for the opportunistic reinforcement learning algorithm, such as improved bounds in the opportunistic bandit setting [**73**] and [**4**], we may need to consider each state-action pair independently. We plan to explore this direction in future work.

5.8. Conclusion

In this paper, we studied opportunistic reinforcement learning, where the regret of choosing a suboptimal action depends on an external condition denoted as the variation factor. We established the OppUCRL2 and OppPSRL algorithms, which are variants of the well-known UCRL2 and PSRL algorithms. We also analyzed the regret of OppUCRL2 and presented $\tilde{O}(HS\sqrt{AT})$ regret bounds. Experimental results demonstrated substantial benefits from employing low-cost opportunistic exploration in the OppUCRL2 and OppPSRL algorithms under variation factor fluctuations.

CHAPTER 6

Causal Explanation for Reinforcement Learning: State and Temporal Importance

6.1. Introduction

Reinforcement learning (RL) is increasingly being considered in domains with significant social and safety implications such as healthcare, transportation, and finance. This growing societal-scale impact has raised a set of concerns, including trust, bias, and explainability. For example, can we explain how an RL agent arrives at a certain decision? When a policy performs well, can we explain why? These concerns mainly arise from two factors. First, many popular RL algorithms, particularly deep RL, utilize neural networks, which are essentially black boxes with their inner workings being opaque not only to lay persons but also to data scientists. Second, RL is a trial-and-error learning algorithm in which an agent tries to find a policy that minimizes a long-term reward by repeatedly interacting with its environment. Temporal information such as relationships between states at different time instances plays a key role in RL and subsequently adds another layer of complexity compared to supervised learning.

The field of explainable RL (XRL), a sub-field of explainable AI (XAI), aims to partially address these concerns by providing explanations as to why an RL agent arrives at a particular conclusion or action. While still in its infancy, XRL has made good progress over the past few years, particularly by taking advantage of existing XAI methods [**75**, **76**, **77**]. For instance, inspired by the saliency map method [**129**] in supervised learning which explains image classifiers by highlighting "important" pixels in terms of classifying images, some XRL methods attempt to explain the decisions made by an RL agent by generating maps that highlight "important" state features [**78**, **130**, **131**]. However, there exist at least two major limitations in state-of-the-art XRL methods. First, the majority of them take an *associational* perspective. For instance, the aforementioned studies quantify the "importance" of a feature by calculating the correlation between the state

feature and an action. Since it is well known that "correlation doesn't imply causation" [**132**], it is possible that features with a high correlation may not necessarily be the real "cause" of the action, resulting in a misleading explanation that can lead to user skepticism and possibly even rejection of the RL system. Second, *temporal* information is not generally considered. Temporal effects, such as the interaction between states and actions over time, which as mentioned previously is essential in RL, are not taken into account.

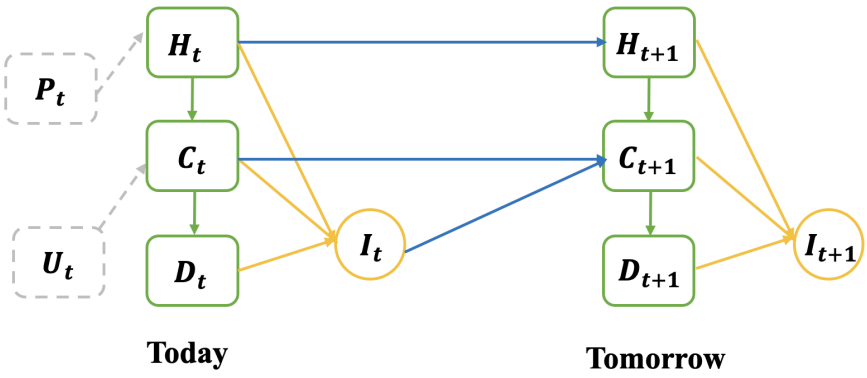


FIGURE 6.1. Causal graph of the crop irrigation problem. Endogenous and exogenous states are denoted by dashed and solid rectangles, respectively, while actions are denoted by circles. More details about causal graphs can be found in the **Preliminaries** section.

In this paper, we propose a *causal* XRL mechanism. Specifically, we explain an RL policy by incorporating a causal model that we have about the relationship between states and actions. To best illustrate the key features of our XRL mechanism, we use a concrete crop irrigation problem as an example, as shown in Fig. 6.1 (more details can be found in the **Evaluation** section). In this problem, an RL policy π controls the amount of irrigation water (I_t) based on the following endogenous (observed) state variables: humidity (H_t), crop weight (C_t), and radiation (D_t). Its goal is to maximize the crop yield during harvest. Crop growth is also affected by some other features, including the observed precipitation (P_t) and other exogenous (unobserved) variables U_t. To explain why policy π arrives at a particular action I_t at the current state, our XRL method quantifies the *causal* importance of each state feature, such as H_t, in the context of this action I_t via *counterfactual reasoning* [**133**, **134**], i.e., by calculating how the action would have changed if the feature had been different.

Our proposed XRL mechanism addresses the aforementioned limitations as follows. First, our method can generate inherently causal explanations. To be more specific, in essence, importance measures used in associational methods can only capture *direct* effects while our causal importance measures capture *total* causal effects. For example, for the state feature H_t, our method can account for two causal chains: the direct effect chain $H_t \rightarrow I_t$ and the indirect effect chain $H_t \rightarrow C_t \rightarrow I_t$, while associational methods only consider the former. Second, our method can quantify the temporal effect between actions and states, such as the effect of today's humidity H_t on tomorrow's irrigation I_{t+1}. In contrast, associational methods, such as saliency map [**78**], cannot measure how previous state features can affect the current action because their models only formulate the relationship between state and action in one time step and ignore temporal relations. To the best of our knowledge, our XRL mechanism is the *first work* that explains RL policies by causally explaining their actions based on causal state and temporal importance. It has been studied that humans are more receptive to a contrastive explanation, i.e., humans answer a "Why X?" question through the answer to the often only implied-counterfactual "Why not Y instead?" [**134**, **135**]. Because our causal explanations are based on contrastive samples, users may find our explanations more intuitive.

6.2. Related Work

Explainable RL (XRL). Based on how an XRL algorithm generates its explanation, we can categorize existing XRL methods into state-based, reward-based, and global surrogate explanations [**75**, **76**, **77**]. State-based methods explain an action by highlighting state features that are important in terms of generating the action [**78**, **79**]. Reward-based methods generally apply reward decomposition and identify the sub-rewards that contribute the most to decision making [**80**]. Global surrogate methods generally approximate the original RL policy with a simpler and transparent (also called intrinsically explainable) surrogate model, such as decision trees, and then generate explanations with the surrogate model [**81**]. In the context of state-based methods, there are generally two ways to quantify feature importance: (i) gradient-based methods, such as simple gradient [**82**] and integrated gradients [**83**], and (ii) sensitivity-based methods, such as LIME [**84**] and SHAP [**85**]. Our work belongs to the category of state-based methods. However, instead of

using associations to calculate importance, a method generally used in existing state-based methods, our method adopts a causal perspective. The benefits of such a causal approach have been discussed in the **Introduction** section.

Causal Explanation. Causality has already been utilized in XAI, mainly in supervised learning settings. Most existing studies quantify feature importance by either using Granger causality [**86**] and average or individual causal effect metric [**87**] or by applying random valued interventions [**88**]. Two recent studies [**89**] and [**90**] are both focused on causal explanations in an RL setting. Compared with [**89**], the main difference is that we provide a different type of explanation. Our method involves finding an importance vector that quantifies the impact of each state feature, while [**89**] provides a causal chain starting from the action. We also demonstrate the ability of our approach to provide temporal importance explanations that can capture the impact of a state feature or action on the future state or action. This aspect has been discussed in the crop irrigation experiment in Section 6.6.1. Additionally, we construct structural causal models(SCM) differently. While the action is modeled as an edge in the SCM in the paper [**89**], our method formulates the action as a vertex in the SCM model, allowing us to quantify the state feature impact on action. As for [**90**], our approach is unique in that it can calculate the temporal importance of a state, which is not achievable by their method. Furthermore, we have provided a value-based importance definition of Q-value that differs from their method. Another significant difference between our approach and [**90**] is the underlying assumption. Our method takes into account intra-state relations, which are ignored in Olson's work. Neglecting intra-state causality is more likely to result in an invalid state after the intervention, leading to inaccurate estimates of importance. Therefore, our approach considers the causal relationships between state features to provide a more accurate and comprehensive explanation of the problem.

6.3. Preliminaries

We introduce the notations used throughout the paper. We use capital letters such as X to denote a random variable and small letters such as x for its value. Bold letters such as $\mathbf{X}$ denote a vector of random variables and superscripts such as $\mathbf{X}^{(i)}$ denote its i-th element. Calligraphic letters such as $\mathcal{X}$ denote sets. For a given natural number n, $[n]$ denotes the set $\{1, 2, \cdots, n\}$.

Causal Graph and Skeleton. Causal graphs are probabilistic graphical models that define data-generating processes [**132**]. Each vertex of the graph represents a variable. Given a set of variables $\mathcal{V} = \{V_i, i \in [n]\}$, a directed edge from a variable V_j to V_i denotes that V_i responds to changes in V_j when all other variables are held constant. Variables connected to V_i through directed edges are defined as the parents of V_i, or "direct causes of V_i," and the set of all such variables is denoted by $\mathcal{P}a_i$. The skeleton of a causal graph is defined as the topology of the graph. The skeleton can be obtained using background knowledge or learned using causal discovery algorithms, such as the classical constraint-based PC algorithm [**136**] and those based on linear non-Gaussian models [**137**]. In this work, we assume the skeleton is given.

SCM. In a causal graph, we can define the value of each variable V_i as a function of its parents and exogenous variables. Formally, we have the following definition of SCM: let $\mathcal{V} = \{V_i, i \in [n]\}$ be a set of endogenous(observed) variables and $\mathcal{U} = \{U_i, i \in [n]\}$ be a set of exogenous(unobserved) variables. A SCM [**132**] is defined as a set of structural equations in the form of

$$V_i = f_i(\mathcal{P}a_i, U_i), \mathcal{P}a_i \subset \mathcal{V}, U_i \subset \mathcal{U}, i \in [n], \tag{6.1}$$

where function f_i represents a causal mechanism that determines the value of V_i using its parents and the exogenous variables.

Intervention and Do-operation. SCM can be used for causal interventions, denoted by the $do(\cdot)$ operator. $do(V_i = v)$ means setting the value of V_i to a constant v regardless of its structural equation in the SCM, i.e., ignoring the edges into the vertex V_i. Note that the do-operation differs from the conditioning operation in statistics. Conditioning on a variable implies information about its parent variables due to correlation.

Counterfactual Reasoning. Counterfactual reasoning allows us to answer "what if" questions. For example, assume that the state is $X_t = x$ and the action is $A_t = a$. We are interested in knowing what would have happened if the state had been at a different value x'. This implies a counterfactual question [**132**]. The counterfactual outcome of A_t can be represented as $A_{t,X_t=x'}|X_t = x, A_t = a$. Given an SCM, we can perform counterfactual reasoning based on intervention through the following two steps:

(1) Recover the value of exogenous variable U as u through the structural function f and the values $X_t = x$, $A_t = a$;

(2) Calculate the counterfactual outcome as $A_t|do(X_t = x'), U = u$. More specifically, in SCM, we set up the value of X_t to x'. Then we substitute all exogenous variable values to the right side of the functions and get the counterfactual outcome A_t.

MDP and RL. An infinite-horizon Markov Decision Process (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, P, R)$, where $\mathcal{S} \in \mathbb{R}^m$ and $\mathcal{A} \in \mathbb{R}$ are finite sets of states and actions, $P(\mathbf{s}, a, \mathbf{s}')$ is the probability of transitioning from state $\mathbf{s}$ to state $\mathbf{s}'$ after taking action a, and $R(\mathbf{s}, a)$ is the reward for taking a in $\mathbf{s}$. An RL policy π returns an action to take at state $\mathbf{s}$, and its associated Q-function, $Q_\pi(\mathbf{s}, a)$, provides the expected infinite-horizon γ-discounted cumulative reward for taking action a at state $\mathbf{s}$ and following π thereafter.

6.4. Problem Formulation

Our focus is on policy explainability, and we assume that the policy π and its associated Q-function, $Q_\pi(\mathbf{s}, a)$, are given. Note that the policy may or may not be optimal. We require a dataset containing trajectories of the agent interacting with the MDP using the policy π. A single trajectory consists of a sequence of $(\mathbf{s}, a, r, \mathbf{s}')$ tuples. Additionally, We assume that the skeleton of the causal graph, such as the one shown in Fig. 6.1 for the crop irrigation problem, is known. We do not assume that the SCM, more specifically its structural functions, is given. We assume the additive noise for the SCM but not its linearity (discussed in Eq. (6.2) in Section 6.5.1). The **goal** is to answer the question "why does the policy π select the current action a at the current state $\mathbf{s}$?" We provide causal explanations for this question from two perspectives: state importance and temporal importance.

Importance vector for state. The first aspect of our explanation is to use the important state feature to provide an explanation. Specifically, we seek to construct an **importance vector** for the state, where each dimension measures the impact of the corresponding state feature on the action. For instance, in the crop irrigation problem, we can answer the question "why does the RL agent irrigate more water today?" by stating that "the impact of humidity, crop weight, and radiation on the current irrigation decision is quantified as $[0.8, 0.1, 0.1]$ respectively. Formally, we have the

following definition of the importance vector for state explanation. Given state $\mathbf{s}_t$ and policy π, the importance of each feature of $\mathbf{s}_t$ for the current action a_t is quantified as $\mathbf{w}_t$. The explanation is that the features in state $\mathbf{s}_t$ have causal importance $\mathbf{w}_t$ on policy π to select action a_t at state $\mathbf{s}_t$.

Temporal importance of action/state. The second aspect of our explanation considers the temporal aspect of RL. Here, we measure how the actions and states in the past impact the current action. We can generalize the importance vector above to past states and actions. Formally, given state $\mathbf{s}_t$, policy π and the history trajectory of the agent $\mathcal{H}_t := \{(\mathbf{s}_\tau, a_\tau), \tau \leq t\}$, we define the effect of a past action a_τ on the current action a_t as $w_t^{a_\tau}$. Similarly, for a past state $\mathbf{s}_\tau$, we define the temporal importance vector $\mathbf{w}_t^\tau$, in which each dimension measures the impact of the corresponding state feature at time step τ on current action a_t. Then we use $w_t^{a_\tau}$ and $\mathbf{w}_t^\tau$ to quantify the impact of past states and action.

6.5. Explanation

6.5.1. Importance Vector for State. Our mechanism implements the following two steps to obtain the importance vector $\mathbf{w}_t$.

(1) Train SCM structural functions between the states and actions using the data of historical trajectories of the RL agent;

(2) Compute the important vector by intervening in the SCM.

First, we notice that there are three types of causal relations between the states and actions: intra-state, policy-defined, and transition-defined relations. As shown in Fig. 6.2, the green directed edges represent the intra-state relations, which are defined by the underlying causal mechanism. The orange edges describe the policy and represent how the state variables affect the action. The third type of relation shown as blue edges is the causal relationship between the states across different times. They represent the dynamics of the environment and depend on the transition probability $P(\mathbf{s}_t, a_t, \mathbf{s}_{t+1})$ in the MDP.

We assume that the intra-state and transition-defined causal relations are captured by the causal graph skeleton. For the policy-defined relations, we assume a general case where all state features are the causal parents of the action. In the causal graph, each edge defines a causal relation, and the vertex defines a variable V with a causal structural function f. Then we only need to learn

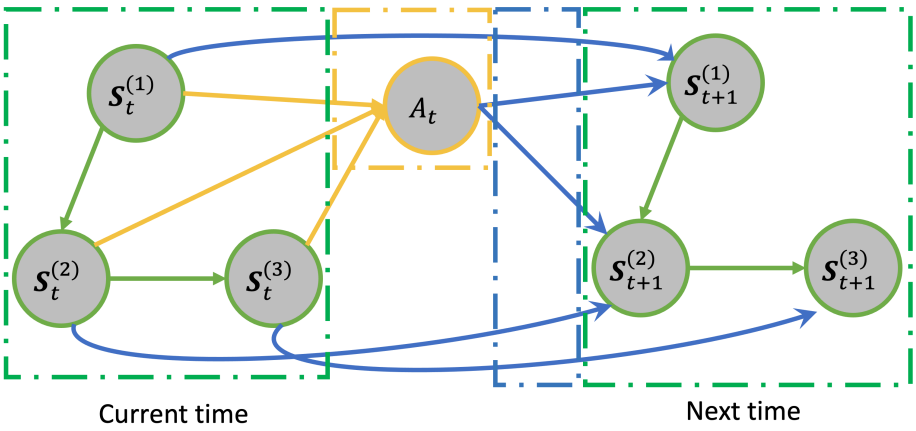


FIGURE 6.2. Example causal graph between the state and action.

Example causal graph between the state and action. $\mathbf{S}_t^{(i)}$ is the i-th dimension of the interested state $\mathbf{S}$ at time t. Each vertex also has a corresponding exogenous variable, which has no parent and its only child is the associated endogenous variable. Per causality conventions, the exogenous variables are omitted in the graph.

the causal structural functions between the vertices. To achieve this, we can learn each vertex's function separately. For a vertex V_i and its parents $\mathcal{P}a_i$, based on Eq. (7.1), we make an additive noise assumption to simplify the problem and formulate the function mapping between V_i and $\mathcal{P}a_i$ as

$$V_i = f_i(\mathcal{P}a_i) + U_i, \tag{6.2}$$

where U_i is an exogenous variable. We note that the additive noise assumption is widely used in the causal discovery literature [**138**, **139**]. We then use supervised learning to learn the function mapping among the vertices. Specifically, f_a for action a_t is defined as

$$A_t = f_a(\mathbf{S}_t^{(1)}, \cdots, \mathbf{S}_t^{(m)}, U_a),$$

where m is the dimension of the state, and U_a is the exogenous variable for the actions.

For the state variables, we denote all exogenous variables as a vector $\mathbf{U_S} := [U_1, \cdots, U_m]$ and learn the structural functions. Intuitively, the exogenous variables U_a and $\mathbf{U_S}$ represent not only random noise but also hidden features or the stochasticity of the policy for the intra-state and policy-defined causal relations. For transition-defined relations, the exogenous variables can be regarded as the stochasticity in the environment.

6.5.2. Action-based Importance. Given a state $\mathbf{s}_t$ and an action a_t, the importance vector $\mathbf{w}_t$ is calculated by applying intervention on the learned SCM. Based on the additive noise assumption, we recover the values of the exogenous variables $\mathbf{U_s}$ and U_a according to the value of a_t, $\mathbf{s_t}$ and the learned causal structural functions. Then we define $\mathbf{w}_t$ using the intervention operation (counterfactual reasoning). Specifically, we define the importance vector $\mathbf{w}_t = [\mathbf{w}_t^{(1)}, \cdots, \mathbf{w}_t^{(m)}]$ as

$$\mathbf{w}_t^{(i)} = \frac{\left| \left(A_{t,\mathbf{S}_t^{(i)}=\mathbf{s}_t^{(i)}+\delta} \middle| \mathbf{S}_t = \mathbf{s}_t, A_t = a_t \right) - a_t \right|}{\delta}, \tag{6.3}$$

where $|\cdot|$ is a vector norm (e.g., absolute-value norm) and δ is a small perturbation value chosen according to the problem setting. The term $A_{t,\mathbf{S}_t^{(i)}=\mathbf{s}_t^{(i)}+\delta} | \mathbf{S}_t = \mathbf{s}_t, A_t = a_t$ represents the counterfactual outcome of A_t if we set $\mathbf{S}_t^{(i)} = \mathbf{s}_t^{(i)} + \delta$. In our case, the value of the exogenous variables can be recovered using the additive noise assumption, so the value of $A_{t,\mathbf{S}_t^{(i)}=\mathbf{s}_t^{(i)}+\delta} | \mathbf{S}_t = \mathbf{s}_t, A_t = a_t$ can be determined. We interpret the result as that the features with a larger $\mathbf{w}_t^{(i)}$ have a more significant causal impact on the agent's action a_t. Note that in the simulation, we average the importance from both positive and negative δ and return the average as the final score. The perturbation amount δ is a hyperparameter and should be selected according to each problem setting.

6.5.3. Q-value-based Importance. While action-based importance can capture the causal impact of states on the change of the action, it may not capture the more subtle causal importance when the selected action does not change, especially when the action space is discrete. Specifically, $A_{t,\mathbf{S}_t^{(i)}=\mathbf{s}_t^{(i)}+\delta} | \mathbf{S}_t = \mathbf{s}_t, A_t = a_t$ may not change after a perturbation of δ, which will result in a $\mathbf{w}_t^{(i)} = 0$. However, this is different from when there are no causal paths from feature $\mathbf{S}_t^{(i)}$ to the action A_t, also resulting in a $\mathbf{w}_t^{(i)} = 0$. Therefore, we also define Q-value-based importance as follows:

$${}^{Q}\mathbf{w}_t^{(i)} = \frac{|Q_\pi^{\text{perturb}} - Q_\pi(\mathbf{s}_t, a_t)|}{\delta}, \tag{6.4}$$

where $Q_\pi^{\text{perturb}} = Q_\pi(\mathbf{S}_{t,\mathbf{S}_t^{(i)}=\mathbf{s}_t^{(i)}+\delta}, A_{t,\mathbf{S}_t^{(i)}=\mathbf{s}_t^{(i)}+\delta} | \mathbf{S}_t = \mathbf{s}_t, A_t = a_t)$. In detail, we use counterfactual reasoning to compute the counterfactual outcome of A_t and S_t after setting $\mathbf{S}_t^{(i)} = \mathbf{s}_t^{(i)} + \delta$ and then substituting them into Q_π to evaluate the corresponding Q-value. Similar to the action-based importance, we account for both positive and negative importance in practice. See the Blackjack

Section 6.6.3 in evaluation for a comparison between Eq. (6.3) and Eq. (6.4) on an example with a discrete action space.

In most RL algorithms, Q-value critically impacts which actions to choose. Therefore, we consider Q-valued-based importance as explanations on the action through the Q-value. However, we note that the Q-value-based importance method sometimes cannot reflect which features the policy really depends on. Some features may contribute largely to the Q-value of all state-action pairs ($\{Q(\mathbf{s}_t, a_t), a_t \in \mathcal{A}\}$, but not to the decision making process - the action with the largest Q-value ($\arg\max_{a_t \in \mathcal{A}} Q(\mathbf{s}_t, a_t)$). In such cases, these features may have an equal impact on the Q-value regardless of the action. For example, in the crop irrigation problem, crop pests have an impact on the crop yield (Q-value) but don't impact the amount of irrigation water (the action). Some related simulations are shown in Appendix 6.9.6. In summary, we suggest using the action-based importance method by default and the Q-value-based method as a supplement.

6.5.4. Temporal Importance and Cascading SCM. Temporal importance allows us to quantify the impact of past states and actions on the current action. In RL, estimating of temporal effect is important because policies are generally non-myopic, and actions should affect all future states and actions. To measure the importance beyond the previous step, we define an extended causal model that includes state features and actions in the previous time step, as shown in Fig. 6.1. In this model, the vertices in the graph are $\{\mathbf{S}_\tau, A_\tau\}_{\tau=1}^{T}$. For simplicity, we assume the system is stationary, so the causal relations are stationary and do not change over time. Therefore, the structural functions are the same as those defined in Fig. 6.2, i.e., the mechanism of an edge $(\mathbf{S}_\tau^{(i)}, \mathbf{S}_{\tau+1}^{(j)})$ will be the same as the edge $(\mathbf{S}_t^{(i)}, \mathbf{S}_{t+1}^{(j)})$. The extended causal model can be regarded as a cascade of multiple copies of the same module, where each module is similar to that in Fig. 6.2. We can estimate the effect of perturbing any features or actions at any step through intervention, and the effect will propagate through the modules to the final time step. We illustrate the temporal importance in the Blackjack experiment in Section 6.6.3.

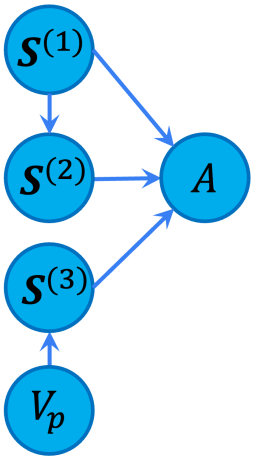
6.5.5. Comparison with Associational Methods. In Eq. (6.3), we define importance by applying intervention. If we change the *do* action to the conditioning operation, we have the

following definition, which is the same as the association-based saliency map method:

$$ {}_{\text{sal}}\mathbf{w}_t^{(i)} = \frac{\left| A_t | \mathbf{S}_t = [\mathbf{s}_t^{(1)}, \cdots, \mathbf{s}_t^{(i)} + \delta, \cdots, \mathbf{s}_t^{(m)}] - a_t \right|}{\delta} \tag{6.5} $$

Associational models cannot perform individual-level counterfactual reasoning and hence cannot infer the counterfactual outcome after changing the value of one feature of the current state. As pointed out by [**132**], counterfactual reasoning can infer the specific property of the considered individual that is related to the exogenous variables, and then derives what would have happened if the agent had been in an alternative state. In our method, we use counterfactual reasoning to recover the environment at the current state and estimate how the action responds to the change in one of the state features. So our causal importance can capture more insights compared to the associational methods.

In Fig. 6.3, we use a one-step MDP toy example to demonstrate the difference. Omitting the time step subscript in the notation, we assume the policy is defined on the state space $\mathbf{S} = [\mathbf{S}^{(1)}, \mathbf{S}^{(2)}, \mathbf{S}^{(3)}]$. An observed variable V_p is a causal parent of $\mathbf{S}^{(3)}$ but is not defined in the state space. We define the ground truth of the state and policy as Eq. (6.6), where $c_1, c_2, c_3, c_{12}, c_p$ are constant parameters and U_a, U_1, U_2, U_3, U_p are exogenous variables. We use a linear SCM to show the difference between the two methods. We do not assume the SCM to have linear dependencies.



$$ \begin{cases} \mathbf{S}^{(1)} & = U_1 \\ \mathbf{S}^{(2)} & = c_{12}\mathbf{S}^{(1)} + U_2 \\ \mathbf{S}^{(3)} & = c_p V_p + U_3 \\ A & = c_1\mathbf{S}^{(1)} + c_2(\mathbf{S}^{(2)})^2 + c_3\mathbf{S}^{(3)} + U_a \\ V_p & = U_p \end{cases} \tag{6.6} $$

FIGURE 6.3. Example of a one-step MDP.

We assume that both the associational method saliency map and our causal method can learn the ground truth functions. Given a state $\mathbf{s}$, the importance vectors using the two methods are

compared in Table 6.1. We notice that, for $\mathbf{s}^{(1)}$, our method can capture the effect of $\mathbf{s}^{(1)}$ through two causal chains $\mathbf{S}^{(1)} \to A$ and $\mathbf{S}^{(1)} \to \mathbf{S}^{(2)} \to A$, while the saliency map method captures only $\mathbf{S}^{(1)} \to A$. Our causal method considers the fact that a change in $\mathbf{S}^{(1)}$ will result in a change of $\mathbf{S}^{(2)}$ and thus additionally influence the action A. The non-direct paths are also meaningful in explanation and should be considered in measuring the importance of $\mathbf{S}^{(1)}$. However, they are ignored in the saliency map method. The causal importance vector for $\mathbf{s}^{(1)}$ also considers the effect of u_2, which is recovered through counterfactual reasoning. This makes the causal-based importance specific to the current state. Additionally, our method can calculate the effect of V_p on the action A, which can not be achieved by the associational method saliency map.

TABLE 6.1. Importance vector on the environment in Fig. 6.3 using our method and the saliency map method.

	Our method	Saliency map
$\mathbf{s}^{(1)}$	$c_1 + c_2c_{12}(c_{12}(2\mathbf{s}^{(1)} + \delta) + 2u_2)$	c_1
$\mathbf{s}^{(2)}$	$c_2(2\mathbf{s}^{(2)} + \delta)$	$c_2(2\mathbf{s}^{(2)} + \delta)$
$\mathbf{s}^{(3)}$	c_3	c_3
v_p	c_pc_3	N/A

We also note that for features $\mathbf{s}^{(2)}$ and $\mathbf{s}^{(3)}$, the two methods obtain the same result. In cases where a state feature is (1) not a causal parent of other features, (2) the policy is deterministic, and (3) there are no exogenous variables, our method is equivalent to the saliency-style approach. However, these conditions may not be common in RL. In general, there are causal relations among state features, such as the chess positions in the game of chess, the state features [position, velocity, acceleration] in a self-driving problem, and the state features [radiation, temperature, humidity] in a greenhouse control problem.

6.6. Evaluation

We test our causal explanation framework in three toy environments: crop irrigation (Section 6.6.1), collision avoidance (Section 6.6.2), and Blackjack (Section 6.6.3). We also conduct experiments on Lunar Lander, which is a more sophisticated RL environment (Appendix 6.9.3). For each experiment, the system dynamics, policy, training details, and perturbation values used can be found in Appendix 6.9.1. The source code is available in reference [**6**].

6.6.1. Crop Irrigation Problem. We show the results of our explanation algorithm for the crop irrigation problem. We assume a simplified environment dynamic based on agriculture models [**140**]. The growth of the plant at each step is determined by the state features humidity (H_t), crop weight (C_t), and radiation (D_t). The policy controls the amount of water to irrigate each day. Intuitively, it irrigates more when the crop weight is high, and less when the crop weight is low. Details about the environment dynamics and policy are described in Appendix 6.9.1.1. We use Fig. 6.1 as the causal skeleton and apply a neural network to learn the structural equations. Fig. 6.4 shows the importance vector of the state for a given environment $[P_t = 0.07, H_t = 0.12, C_t = 0.44, D_t = 0.70]$ and its corresponding action $I_t = 0.67$. First, we notice that our method can estimate the importance of the feature precipitation(P_t), which is not defined in the state space of the policy. Second, in estimating the causal importance of H_t, our method can estimate the effect of $H_t \to C_t \to I_t$, which results in higher importance compared to the saliency map method. Since an intervention on H_t can induce a change in C_t, causing the action to change more drastically. This effect cannot be measured without a causal model. The same applies to the feature D_t. The full trajectory and the importance vector at each time step can be found in Fig. 6.10 in Appendix 6.9.1.1.

The causality-based action influence model [**89**] can find a causal chain $I_t \to C_t \to \texttt{CropYield}$ and provide the explanation as "the agent takes current action to increase C_t at this step, which aims to increase the eventual crop yield." This explanation only provides the information that C_t is an important factor in the decision-making for the current action but can't quantify it. Moreover, this explanation can't provide information for other state features, such as H_t and D_t which are also measured in our importance vector.

6.6.2. Collision Avoidance Problem. We use a collision avoidance problem to further illustrate that our causal method can find a more meaningful importance vector than saliency map, i.e., which state feature is more impactful to decision-making.

Fig. 6.5(a) shows the state definition for this problem. A car with zero initial velocity travels from the start point to an endpoint over a distance of X_{goal}. The system is controlled in a discrete-time-slot manner and we assume acceleration of the car is constant within each time step. The state $\mathbf{S}_t$ includes the distance from the start X_t, the distance to the end D_t, and the velocity V_t of

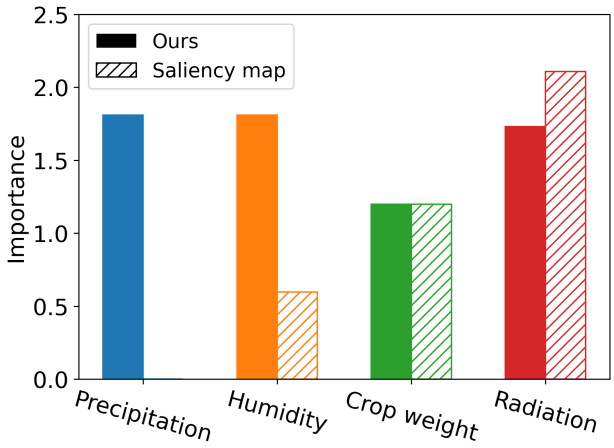
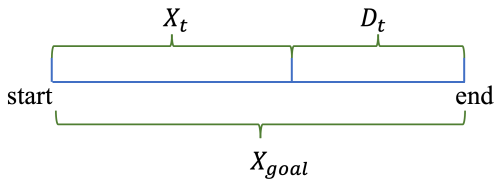
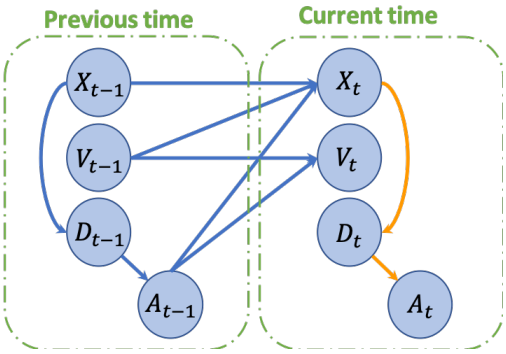


FIGURE 6.4. The importance vector for the crop irrigation problem.



(a) The state definition



(b) The causal graph of states and actions

FIGURE 6.5. The collision avoidance problem and its corresponding SCM skeleton.

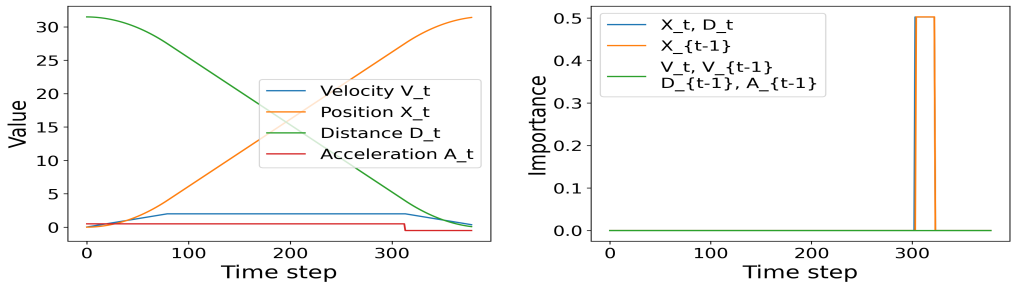
the car, i.e., $\mathbf{S}_t := [V_t, X_t, D_t]$, where $V_t \leq v_{\max}$ and $v_{\max}$ is the maximum speed of the car. The action A_t is the car's acceleration, which is bounded $|A_t| \leq e_{\max}$. We assume the acceleration of the car is constant within each time step. More detailed settings are described in the simulation section in the supplementary materials. The objective is to find a policy π to minimize the traveling time under the condition that the final velocity is zero at the endpoint (collision avoidance).

An RL agent learns the following **optimal** control policy for this avoidance problem, which is also known as the bang-bang control (optimal under certain technical conditions) [**141**]:

$$A_t = \begin{cases} e_{\max} & \text{if } D_t \leq v_{\max}^2/(2e_{\max}) \\ -e_{\max} & \text{otherwise} \end{cases} \tag{6.7}$$

Intuitively, this policy accelerates as much as possible until reaching the critical point defined above. Then it will decelerate until reaching the goal.

We use Fig. 6.5(b) as the SCM skeleton and use linear regression to learn the structural equations as the entire dynamics are linear. The detail about the system dynamics is described in the appendix.



(a) A trajectory of using bang-bang control on the collision avoidance problem.

(b) The result importance from our algorithm on the trajectory in Fig. 6.6(a).

FIGURE 6.6. Trajectory and importance on the collision avoidance problem.

Fig. 6.6(a) shows a trajectory under the policy bang-bang control and Fig. 6.6(b) shows its corresponding causal importance results. The importance of $V_t, A_{t-1}, D_{t-1}, V_{t-1}$ are zero throughout the time history, and those of X_t, D_t, X_{t-1} have peak importance of $[0.502, 0.502, 0.502]$, respectively, between time step 303-322, during which the car changes the direction of acceleration to avoid hitting the obstacle. The importance curves of X_t, D_t, and X_{t-1} have the same shape, but that of X_{t-1} is off by one time step, corresponding to their time step subscript. If we were to use the associational saliency method [**78**] X_t would have a constant zero importance since the action is solely determined by the feature D_t. In comparison, our method can find non-zero importance through the edge $X_t \to D_t$. It is reasonable that X_t causally affects A_t, because, in the physical

world, the path length X_t is the cause of the measurement of the distance to the end D_t. Although in Eq. (6.7) the action A_t is only decided by D_t, the source cause of the change in D_t is X_t. We can only obtain such information through a causal model, not an associational one.

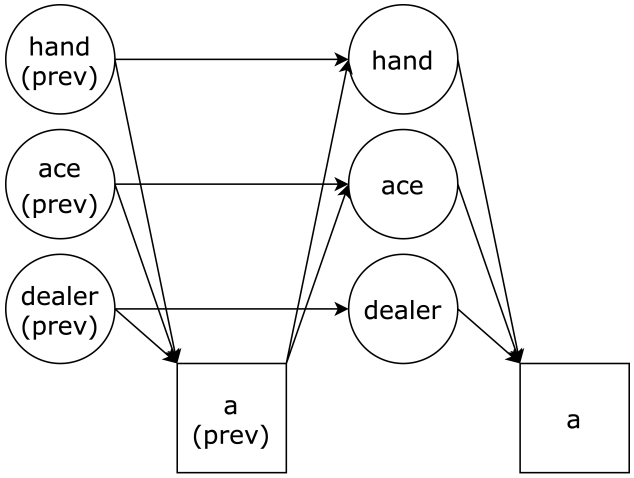


FIGURE 6.7. The skeleton of the Blackjack SCM.

6.6.3. Blackjack. We test our explanation mechanism on a simplified game of Blackjack. The state is defined as [`hand`, `ace`, `dealer`], where `hand` represents the sum of current cards in hand, `ace` represents if the player has a usable ace (an ace that can either be a 1 or an 11), and `dealer`, is the value of the dealer's shown card. There are two possible actions: to `draw` a new card or to `stick` and end the game. We use an on-policy Monte-Carlo control [**8**] agent to test our mechanism. Since the problem dynamic is non-linear, we use a neural network to learn each structural equation. Fig. 6.7 shows the skeleton of the SCM. More details about the rules of the game are explained in Appendix 6.9.1.2. Note that in Blackjack, the exogenous variable U_i of some features can be interpreted as the stochasticity or the "luck" during the input trajectory. e.g., $U_{\texttt{hand},t}$ corresponds to the value of the card drawn at step t if the previous action is `draw`.

Using Q-values as Metric. The solid bars in Fig. 6.8 on the next page show the result of Q-value-based importance based on Eq. (6.4). We interpret the result as follows: (1) The importance of all features are highest at step 1. This is because state 1 is closest to the decision boundary of the policy, and thus applying a perturbation at this step is easier to change the Q-value distribution; (2) The importance of `dealer` and `dealer_prev` are the same throughout the trajectory. This is

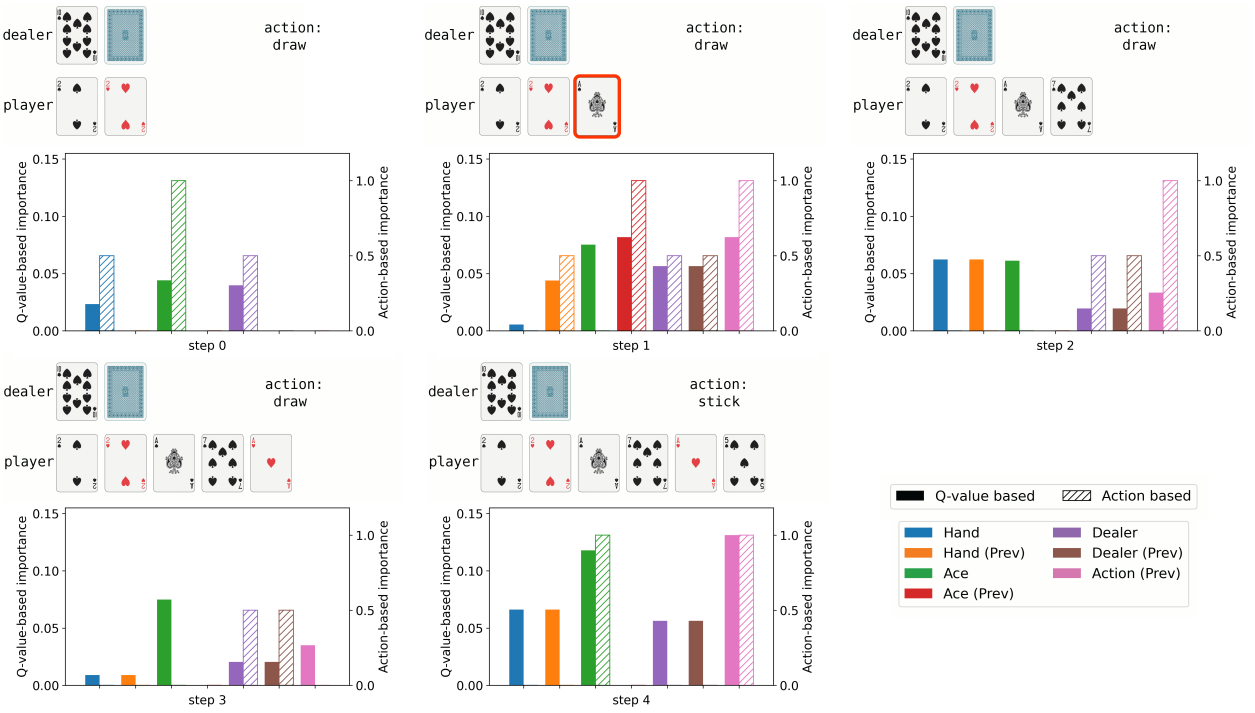


FIGURE 6.8. A trajectory of a blackjack game and the result from running our mechanism using either the Q-values or the action as the metric. In each sub-graph, the top figure shows the state, and the usable ace is highlighted in red if present. The bottom figure shows the importance of each feature. The solid bars are the Q-value-based importance and the hatched bars are the action-based importance. Note that at step 1, the importance for the previous hand, previous dealer, previous ace, and previous action are not applicable since there is no previous state for the first state.

due to the fact that `dealer` and `dealer_prev` are always the same. Thus, applying a perturbation on `dealer_prev` will have the same effect as applying a perturbation on `dealer` assuming changing `dealer_prev` won't incur a change in the previous action; (3) A similar phenomenon can be observed between `hand` and `hand_prev`. Increasing the hand at step $t-1$ by one will have the same outcome as drawing a card with one higher value at t. The occasional difference comes from the change in `hand_prev` causing `a_prev` to change; (4) The importance of `ace` is highest at steps 2 and 5. In both of these two states, changing if the player has an ace or not while keeping other features the same will change the best action and a larger difference in the Q-values, which causes the importance to be higher.

Using Action as Metric. The hatched bars in Fig. 6.8 show the result of action-based importance based on Eq. (6.3). The importance is more "bursty", and features, such as `hand`, have an importance of zero in the majority of the steps since a perturbation of size one could not trigger a change in the action. However, intuitively, `hand` is crucial to the agent's decision-making. Therefore, in this case, we note that the Q-value-based method produces a more reasonable explanation in this example.

Multi-Step Temporal Importance. We cascade the causal graph of blackjack in Fig. 6.7 to estimate the impact of the past states and actions on the current action, and the full SCM is shown in Fig. 6.12 in Appendix 6.9.1.2. Fig. 6.9 shows the results of Q-value-based importance. The importance of A_4 on itself is omitted since it will always be one regardless of any other part of the graph. We interpret the results as follows: (1) The importance of $\texttt{hand}_\tau$ and $\texttt{dealer}_\tau$ is flat over time. As discussed above, perturbing these two features at any given step will mostly change the last state in the same way, resulting in constant importance; (2) The importance of the action a_τ increases as τ gets closer to the last step $t = 4$. An action taken far in the past should generally have a smaller impact on the current action, which corresponds to the increasing importance for $\texttt{a}_\tau$ in our explanation.

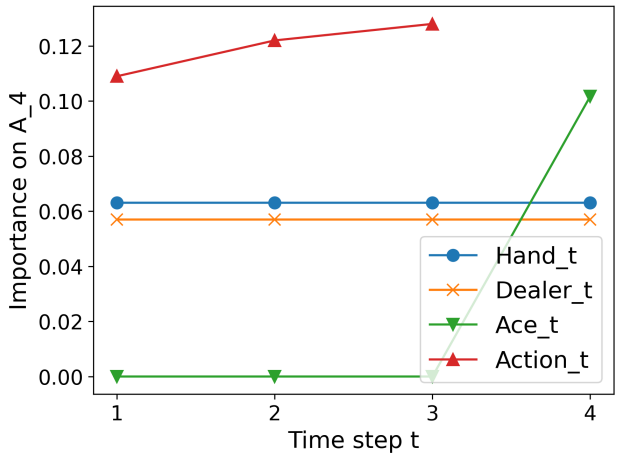


FIGURE 6.9. The Q-value-based temporal importance on A_4 for all state features and actions at past time steps in the Blackjack experiment.

6.6.4. Additional Evaluation. We also evaluate our scheme in a more complex RL environment, Lunar Lander, in Appendix 6.9.3. Lunar Lander is a simulation testing environment

developed by OpenAI Gym [**142**]. The simulation shows that our scheme can explain some specific phases(state) of the spaceship in the landing process.

6.7. Discussions

Our causal importance explanation mechanism is a post-hoc explanation method that uses data collected by an already learned policy. We focus on providing local explanations based on a particular state and action. Counterfactual reasoning is required to recover the exogenous variables and estimate the effect on the given state and action. In this case, the intervention operation is not enough to achieve this goal, as it can only evaluate the average results (population) over the exogenous variables, which is not a local explanation for the given state.

Intra-state Relations. One crucial characteristic of our method is that we consider intra-state relations when computing the importance, which is essential in accurately quantifying the impact of a state feature on the action. Although the MDP defines that a state feature at a certain time step cannot affect another state feature at the same step, it is essential to consider causal relationships within state features when measuring their impact if we use causal intervention or associational perturbation. Since these types of methods require modifying the value of a specific state feature, it should subsequently affect the value of other state features based on real-world causality. For instance, in the collision avoidance problem (Section 6.6.2), the distance to the end (D_t) will change in response to the distance from the start (X_t), and in the crop irrigation problem (Section 6.6.1), the crop weight (C_t) will vary based on the humidity level (H_t). Ignoring the intra-state causality can lead to an invalid state after the intervention, resulting in inaccurate importance estimates for the given state feature. Hence, we formulate the intra-state relations in the SCM to provide more accurate and comprehensive explanations of the problem.

Additive Noise Assumption. With the additive noise assumption in Eq. (6.2), the exogenous variable (noise) can be fully recovered and used for counterfactual reasoning. We note that the full recovery noise assumption can be relaxed for our mechanism. In the case where the exogenous variables have multiple values (not deterministic), we can generalize our definition of importance vector in Eq. (6.3) by replacing the first term with the expectation over different values of exogenous variables using probabilistic counterfactual reasoning [**143**]. Furthermore, the additive

noise assumption is not mandatory. We can use bidirectional conditional GAN [**144**] to model the structure function and use its noise to conduct counterfactual reasoning and obtain the importance vector.

Known SCM Skeleton Assumption. Our approach is based on the assumption that the SCM skeleton is known, which can be obtained either through background knowledge of the problem or learned using causal discovery algorithms. Causal discovery aims to identify causal relations by analyzing the statistical properties of purely observational data. There are several causal discovery algorithms available, including the classical constraint-based PC algorithm [**136**], algorithms based on linear non-Gaussian models [**137**], and algorithms that use the additive noise assumption [**138**, **139**]. These algorithms can be used to learn the SCM skeleton from observational data, which can then be used in our method to quantify the impact of state features and actions on the outcome. There are also existing toolboxes such as [**145**] and [**146**] that can be easily applied directly to data to identify the SCM structure.

Perturbation. In addition to the method we employed in the simulation, which averages the importance derived from both positive and negative δ, maximizing them is also a viable option.To compute the causal importance vector defined in Eq. (6.3), we need to choose a perturbation value δ. As shown in Table 6.1, the importance may depend on δ. Therefore, it is not meaningful to compare importance vectors calculated with different δ. This is a common issue of perturbation-based algorithms, including the saliency map method. In our case, δ should be as small as possible but still be computationally feasible. More detailed sensitivity analysis and normalization on the perturbation value δ can be found in Appendix 6.9.4.

Limitations. Our study has limitations when the state space has high dimensions, for example, in visual RL, where state features are represented as images. Image data is inherently high-dimensional, with multiple features that can interact in complex ways. The SCMs we used may struggle to fully capture the complexity of these interactions, especially when a large number of variables are involved. To address this issue, we suggest utilizing the algorithm of causal discovery in images [**147**] and representation learning [**148**]. Further work is needed to explore this direction.

Another question that might be raised is what will happen if the trained SCM is not perfect. An imperfect SCM will cause the counterfactual reasoning result to be biased, and thus affecting

the final importance. One potential solution is quantifying the uncertainty of the explanation. If the explainer can output its confidence on top of the importance score, users can identify potential out-of-distribution samples where our explanation framework might fail. To achieve this, we need to separate aleatoric uncertainty (which comes from the inherent variability in the environment) and epistemic uncertainty (which represents the imperfection of the model) [**149**]. Our use of SCM may help us to differentiate the two, and this is one of the directions we are currently exploring.

6.8. Conclusion

In this paper, we have developed a causal explanation mechanism that quantifies the causal importance of states on actions and their temporal importance. Our quantitative and qualitative comparisons show that our explanation can capture important factors that affect actions and their temporal importance. This is the first step towards causally explaining RL policies. In future work, it will be necessary to explore different mechanisms to quantify causal importance, relax existing assumptions, build benchmarks, develop human evaluations, and use the explanation to improve evaluation and RL policy training.

6.9. Appendix

6.9.1. Additional Experiments and Details. In this section, we provide additional details regarding the crop irrigation problem, the collision avoidance problem, and the Blackjack experiments. Furthermore, we describe our results on an additional testing environment, Lunar Lander.

All experiments were conducted on a machine with 8 NVIDIA RTX A5000 GPU, an dual AMD EPYC 7662 CPU, and 256 GB RAM.

6.9.1.1. *Crop Irrigation.* This section contains details of the crop irrigation experiment.

System dynamics.

$$\begin{aligned}
\text{Precipitation} &= U(0,1) \\
\text{SolarRadiation} &= U(0,1) \\
\text{Humidity} &= 0.3 \cdot \text{Humidity}_{\text{prev}} + 0.7 \cdot \text{Precipitation} \\
\text{CropWeight} &= \text{CropWeight}_{\text{prev}} \\
&\quad + 0.07 \cdot \left(1 - (0.4 \cdot \text{Humidity} + 0.6 \cdot \text{Irrigation} - \text{Radiation}^2)^2\right) \\
&\quad + 0.03 \cdot U(0,1)
\end{aligned}$$

The change in CropWeight at each step is determined by humidity, irrigation and radiation, and maximum growth is achieved when $0.4 \cdot \text{Humidity} + 0.6 \cdot \text{Irrigation} = \text{Radiation}^2$. An additional exogenous variable is also included in the change of CropWeight. This can be regarded as some unobserved confounders that affect the growth that are not included in the system dynamics, such as $\texttt{CO}_2\texttt{Concentration}$ or the temperature.

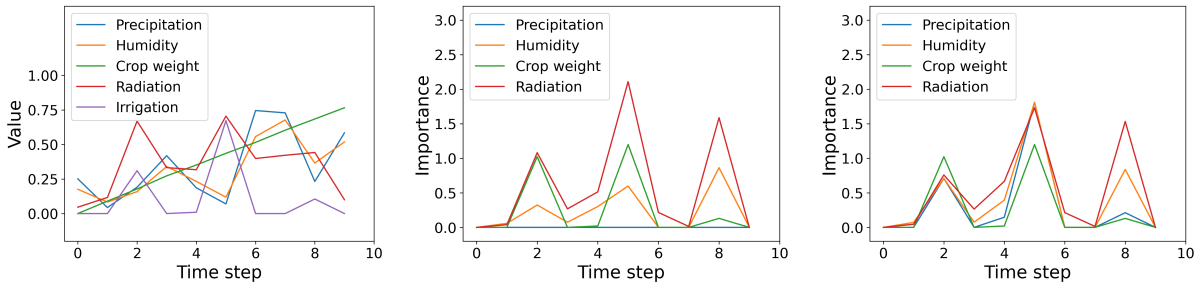
Policy.

$$\text{Irrigation} = (\text{Radiation}^2 - 0.4 \cdot \text{Humidity}) \cdot (1.6 \cdot \text{CropWeight} + 0.2)/0.6$$

The policy we used is a suboptimal policy that multiplies an additional coefficient $1.6 \cdot \text{CropWeight} + 0.2$ on the optimal policy. This will cause the irrigation value to be less than optimal when CropWeight is less than 0.5, and more than optimal and vice versa.

Training. We use a neural network to learn the causal functions in the SCM. The network has three fully-connected layers, each with a hidden size of four. We use Adam with a learning rate of 3×10^{-5} as the optimizer. The training dataset consists of 1000 trajectories (10000 samples) and the network is trained for 50 epochs.

Perturbation. The perturbation value δ used in the intervention is 0.1 w.r.t. the range of each value.



(a) A trajectory of the crop irrigation problem (b) Saliency map method result (c) Causal-based importance result

FIGURE 6.10. Importance vector for state in crop irrigation problem.

6.9.1.2. *Blackjack.* This section contains details and additional figures for the blackjack simulation.

System dynamics. This simulation is done in the blackjack environment in OpenAI Gym [**142**]. The goal is to draw cards such that the sum is close to 21 but never exceeds it. Jack, queen and king have a value of 10, and an ace can be either a 1 or an 11, and an ace is called "usable" when it can be used at an 11 without exceeding 21. We assume the deck is infinite, or equivalently each card is drawn with replacement.

In each game, the dealer starts with a shown card and a face-down card, while the player starts with two shown cards. The game ends if the player's hand exceeds 21, at which the player loses, or if the player chooses to stick, the dealer will reveal the face-down card and draw cards until his sum is 17 or higher. The player wins if the player's sum is closer to 21 or the dealer goes bust.

Policy. We trained the agent using on-policy Monte-Carlo control. Fig. 6.11 shows the policy and the decision boundary.

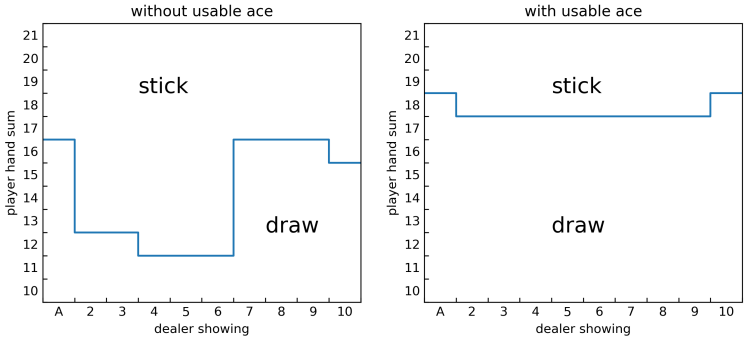


FIGURE 6.11. The policy we use for the blackjack game. The blue line shows the decision boundary.

SCM structure. We assume the blackjack game has a causal structure as shown in Fig. 6.7. Additionally, Fig. 6.12 shows the 5-step cascading SCM we used to test the temporal importance.

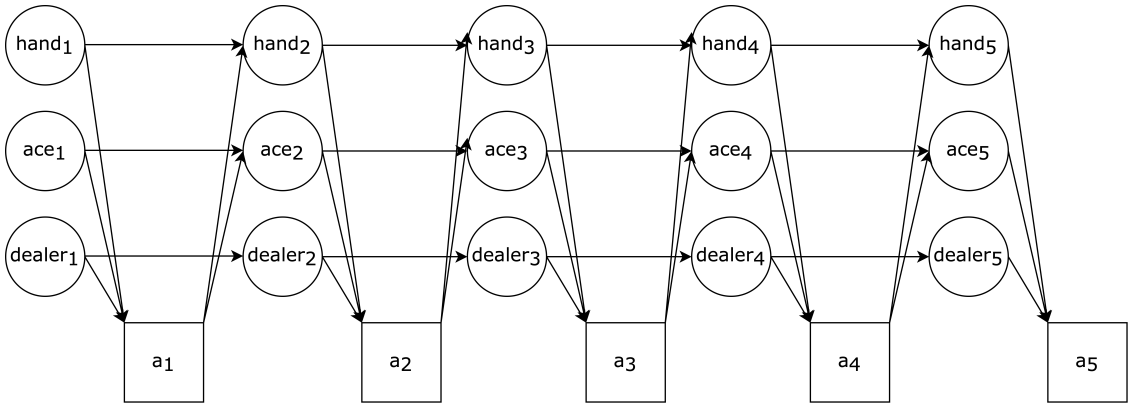


FIGURE 6.12. The skeleton of the cascading SCM for a 5-step blackjack game.

Training. We use a neural network to learn the causal functions in the SCM. The network has three fully-connected layers and each layer has a hidden size of four. We use Adam with a learning rate of 3×10^{-5} as the optimizer. The training dataset consists of 50000 trajectories ($\sim$76000 samples) and the network is trained for 50 epochs.

Perturbation. Since blackjack has a discrete state space, for numerical features "hand" and "dealer", we use a perturbation value $\delta = 1$. For the boolean feature "ace", we flip its value as the perturbation.

6.9.2. Collision Avoidance Problem. We use the collision avoidance problem to further illustrate that our causal method can find a more meaningful importance vector than saliency map, i.e., which state feature is more impactful to decision-making.

System dynamics. The state $\mathbf{S}_t$ includes the distance from the start X_t, the distance to the end D_t, and the velocity V_t of the car, i.e., $\mathbf{S}_t := [V_t, X_t, D_t]$, where $V_t \leq v_{\max}$ and $v_{\max}$ is the maximum speed of the car. The action A_t is the car's acceleration, which is bounded $|A_t| \leq e_{\max}$. The state transition is defined as follows:

$$
\begin{aligned}
V_{t+1} &:= V_t + A_t \Delta t \\
X_{t+1} &:= X_t + V_t \Delta t + \frac{1}{2} A_t \Delta t^2 \\
D_{t+1} &:= X_{\text{goal}} - X_{t+1}
\end{aligned}
$$

The objective of the RL problem is to find a policy π to minimize the traveling time under the condition that the final velocity is zero at the endpoint (collision avoidance).

Policy. An RL agent learns the following **optimal** control policy also known as the bang-bang control (optimal under certain technical conditions) defined as Eq. (6.7)

SCM structure. We use Fig. 6.5(b) as the SCM skeleton and use linear regression to learn the structural equations as the entire dynamics are linear.

Perturbation. The perturbation value δ used in the intervention is 0.1 after normalization.

6.9.3. Lunar Lander.

System dynamics. Lunar lander problem is a simulation testing environment developed by OpenAI Gym [**142**]. The goal is to control a rocket to land on the pad at the center of the surface while conserving fuel. The state space is an 8-dimensional vector containing the horizontal and vertical coordinates, the horizontal and vertical speed, the angle, the angular speed, and if the left/right leg has contacted or not.

The four possible actions are to fire one of its three engines: the main, the left, or the right engine, or to do nothing.

The landing pad location is always at $(0, 0)$. The rocket always starts upright at the same height and position but has a random initial acceleration. The shape of the ground is also randomly generated, but the area around the landing pad is guaranteed to be flat.

Policy. We train our RL policy using DQN [**150**].

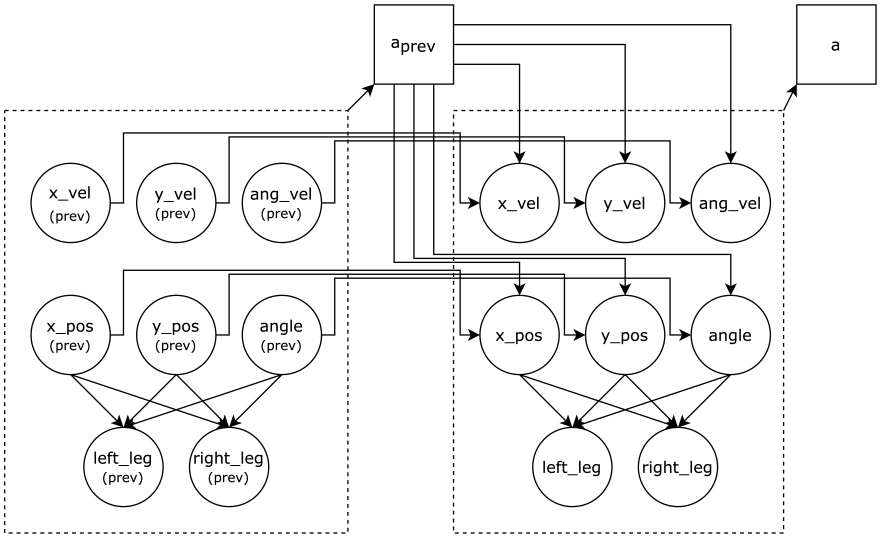
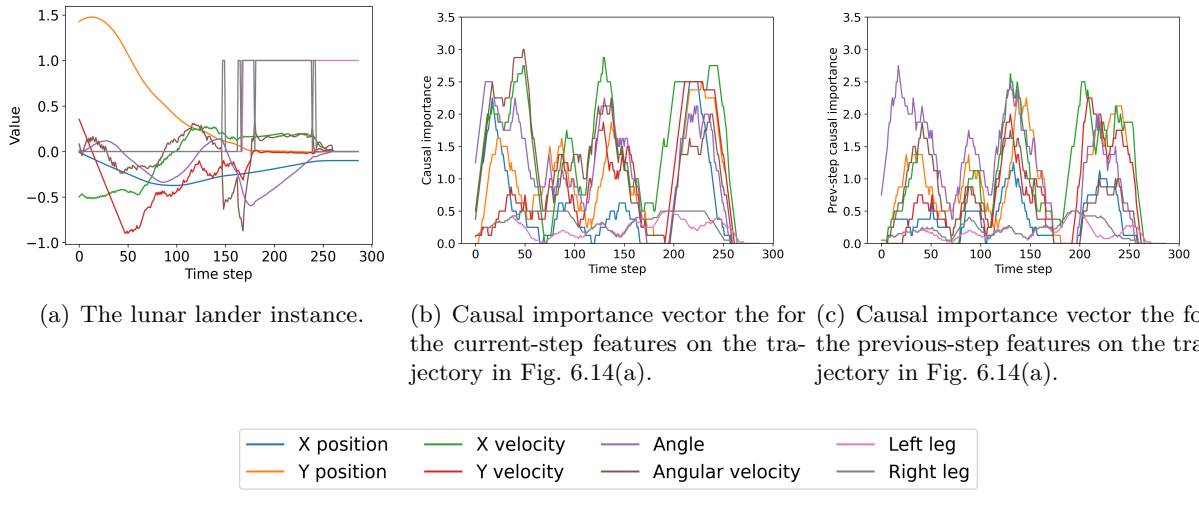


FIGURE 6.13. The causal structure of lunar lander that includes previous state and actions. There should also be edges from each feature to the action at its time step, e.g. edges from `x_pos_prev` to `a_prev`, or from `x_pos` to `a`. These edges are not shown in this graph for simplicity.

SCM structure. We use the Fig. 6.13 as the skeleton of SCM. The structural functions are learned with linear regression using 100 trajectories ($\sim$25000 samples).

Evaluation. Fig. 6.14 shows a trajectory of the agent interacting with the lunar lander environment and the corresponding causal importance using our mechanism. We notice that our mechanism discovers three importance peaks, and we explain this as the agent's decision-making during the landing process consisting of three phases: a "free fall phase", in which the agent mainly falls straight and slightly adjusts its angle to negate the initial momentum; an "adjusting phase", in which the agent mostly fires the main engine to reduce the Y-velocity; and a "touchdown phase", during which the lander is touching the ground and the agent is performing final adjustments to stabilize its angle and speed. Fig. 6.15(a), 6.15(b) and 6.15(c) show our causal importance vector during each of the three phases. We notice that during the "free fall phase", features such as angle, angular velocity and x-velocity are more important since the agent needs to rotate to negate the



(a) The lunar lander instance.

(b) Causal importance vector the for the current-step features on the trajectory in Fig. 6.14(a).

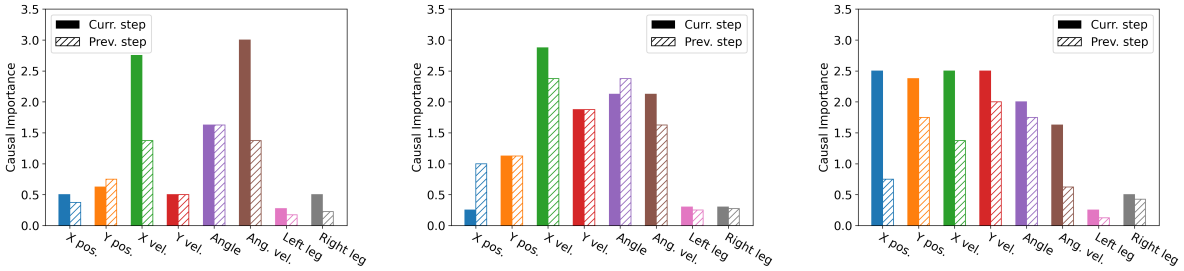
(c) Causal importance vector the for the previous-step features on the trajectory in Fig. 6.14(a).

(d)

FIGURE 6.14. A lunar lander trajectory instance we used to evaluate our algorithm and the corresponding causal importance vector. The "freefall phase" is roughly between steps 0-70, "adjusting phase" is between steps 70-170, and "touchdown phase" is from about step 170 to the end.

initial x-velocity. However, as the rocket approaches the ground during the "adjusting phase", we find an increase in importance for y-velocity since a high vertical velocity is more dangerous to control when the rocket is closer to the ground. In the last "touchdown phase", a large x-position and x-velocity importance can be observed as a change in those features is highly likely to cause the lander to fail to land inside the designated landing zone. Since the lander is already touching the ground, it will take much more effort for the agent to adjust compared to when the lander is still high in the air.

The results are similar to those of saliency-based algorithms [**78**], and Fig. 6.16 shows the difference in importance vector between our algorithm and saliency-based algorithm. Note that differences only occur for the positions and the angle. This is because other features don't have any additional causal paths to the action besides the direct connection. Therefore, the intervention operation is equivalent to the conditioning operation for these features. The features position and angle have an additional causal path through the legs, which causes the difference. Notably, our method captures higher importance for angle, which we interpret as that the landing angle is crucial and is actively managed by the agent.



(a) Importance vector during the "free fall phase" (step 48).

(b) Importance vector during the "adjusting phase" (step 129).

(c) Importance vector during the "touchdown phase" (step 216).

FIGURE 6.15. The importance vector on lunar lander calculated using our method and a comparison with the saliency map method. The solid bars in the first three figures representing the importance of the current-step features and the shaded bars are for the previous-step features.

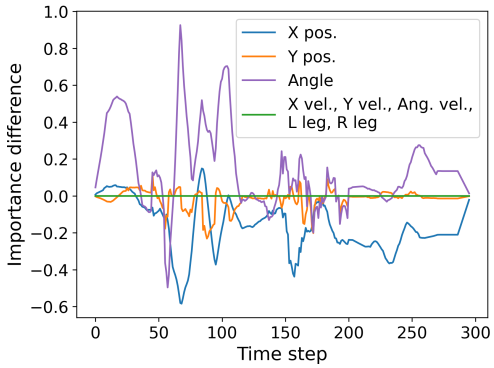


FIGURE 6.16. Difference between our method and the saliency map method for current-step features.

We are also able to compute the importance of the features in the previous steps, and Fig. 6.14(c) and the shaded bars in Fig. 6.15 represent such importance vectors. The previous-step importances are rather similar to those of the current-step features since the size of the time step is comparatively small. However, our algorithm captures that during the "adjusting phase", the previous-step importance for the angle is in general higher than the current-step importance, as changing the previous angle may have a cascading effect on the trajectory and is especially important to the agent when it is actively adjusting the angle.

6.9.4. Sensitivity Analysis. This section performs a sensitivity analysis on how the perturbation amount affects the result of our explanation.

For action-based importance, too small of a perturbation may not yield a meaningful result. This is due to the fact that, depending on the environment and the policy, a too small perturbation may fail to trigger a noticeable change in the action, resulting in a zero importance. This differs from the zero importance case where the policy disregards the feature when making decisions. In our experiments, we use 0.01 with respect to the range of the features for continuous features and the smallest unit for discrete features.

In general, using different perturbation amounts δ on the same state in the same SCM may result in different importance vectors, and vectors calculated using different δ cannot be meaningfully compared. However, if we desire the importance of using different δ to be more on the same level, we suggest finding the highest importance across all features and all time steps and normalizing all results by said number. Section 6.9.4.2 contains an example comparing the importance score with and without the aforementioned normalization.

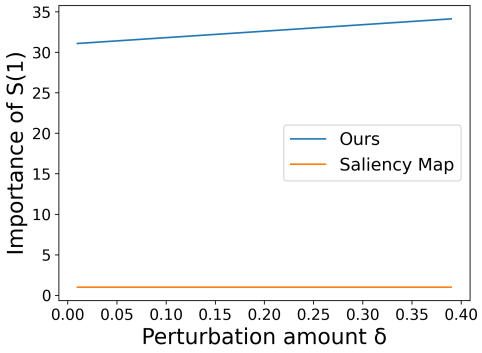


FIGURE 6.17. The importance vector of $\mathbf{S}^{(1)}$ from both our method and the saliency map method with respect to the perturbation amount.

6.9.4.1. *One-step MDP.* As we demonstrated in the example of one-step MDP in Fig. 6.3 and Table 6.1, our importance vector will sometimes be affected by the perturbation amount. For this experiment, we use Fig. 6.3 as the skeleton and the following settings. The constants are

$$c_1 = 1, c_2 = -2, c_3 = 3, c_{12} = 2, c_p = -1$$

We use unit Gaussian distributions as the exogenous variables and the values are

$$u_1 = 0.50, u_2 = -0.14, u_3 = 0.65, u_p = 1.52, u_a = -0.23$$

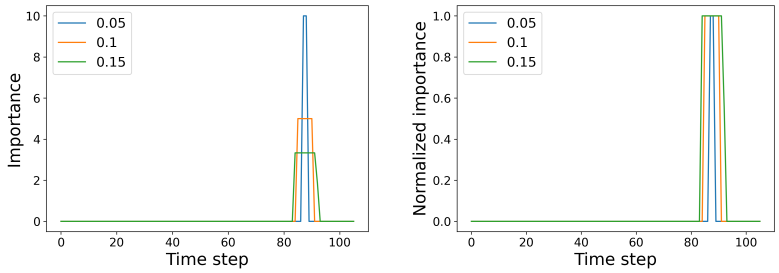
The state value and the corresponding action are then

$$\mathbf{s}^{(1)} = 0.50, \mathbf{s}^{(2)} = 0.86, \mathbf{s}^{(3)} = -0.88, v_p = 1.52, a = 3.83$$

The result of running our method and the saliency map method on the feature $\mathbf{S}^{(1)}$ is shown in Fig. 6.17. Same as in Table 6.1. Our algorithm is linear w.r.t. δ while the saliency map result is constant. The increased importance comes from the causal link $\mathbf{S}^{(1)} \to \mathbf{S}^{(2)} \to A$, which also introduces the linear relationship.

6.9.4.2. *Collision Avoidance.* Fig. 6.18 shows the importance vector of X_t in the collision avoidance problem and different color lines correspond to different perturbation amounts. Note that similar to the result shown in Fig. 6.6(b), the importance of D_t is the same as X_t, and X_{t-1} is the same but off by one time step. Other features have negligible importance.

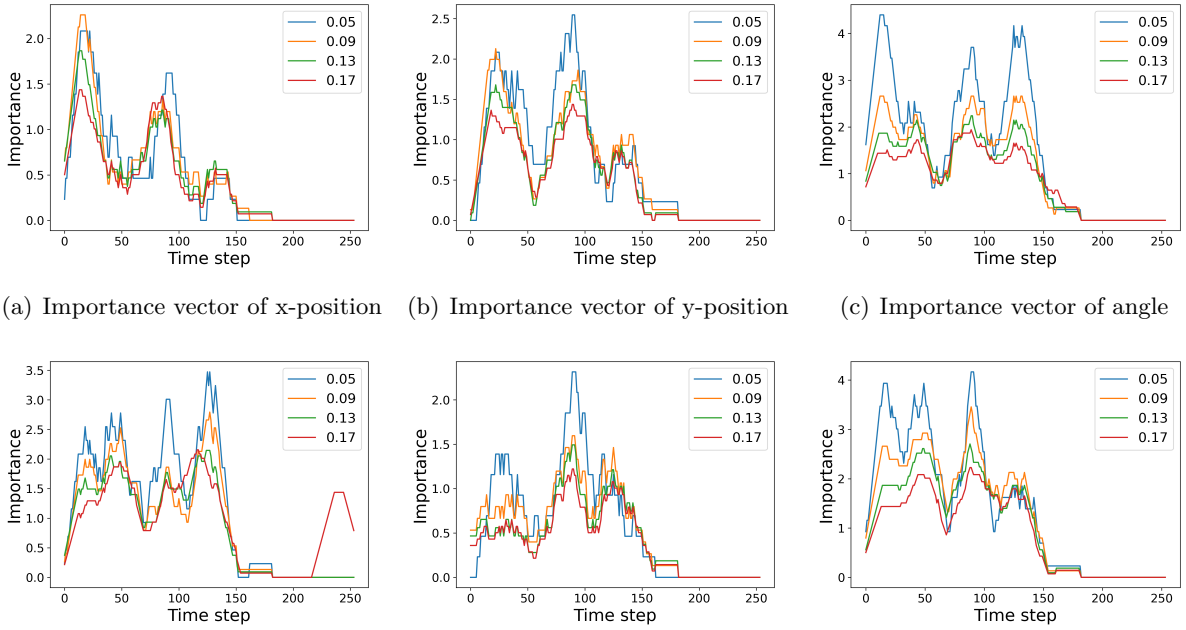
There are two effects of using different perturbation amounts: 1) The number of steps with non-zero importance is increasing as δ increases since a larger δ will cause states further away from the decision boundary to cross the boundary after the perturbation; 2) The value of peak importance is lower. Since we use the action-based importance and the action is essentially binary, the difference in importance solely comes from the normalization we applied on δ (the denominator in Eq. (6.3). If this is undesirable, one way to combat this is to normalize the result using the highest importance across all features and time steps. The normalized result is shown in Fig. 6.18(b), in which the peak value will be one regardless of δ.



(a) Importance vector of X_t (b) Normalized importance vector of X_t

FIGURE 6.18. Sensitivity analysis on the collision avoidance problem.

6.9.5. Lunar Lander. Fig. 6.19 shows the sensitivity analysis on lunar lander and the different color lines correspond to different perturbation amounts. Binary features including left and right leg are not included. The general trend of the result is the same while the value and the exact shape of the curve vary slightly when different δ is used and our result is robust w.r.t. δ.



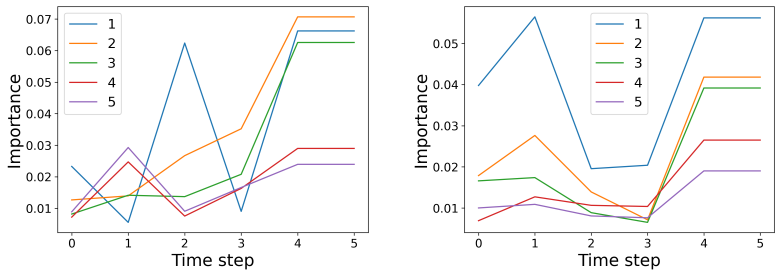
(a) Importance vector of x-position (b) Importance vector of y-position (c) Importance vector of angle

(d) Importance vector of x-velocity (e) Importance vector of y-velocity (f) Importance vector of angular velocity

FIGURE 6.19. Sensitivity analysis on the lunar lander environment.

6.9.5.1. *Blackjack.* Fig. 6.20 shows the sensitivity analysis for blackjack, with different color lines representing different perturbation amounts. The binary feature `ace` is not included. In blackjack, since the smallest legal perturbation amount is one and the range of the value is at most 21, increasing δ has a much larger effect on the result. However, we can observe that the general shape of the curves is similar, indicating the robustness of our method.

6.9.6. Action-based Importance versus Q-value-based Importance. This section discusses the comparison between the action-based importance method and the Q-value-based importance method. It demonstrates that the Q-value-based method sometimes fails to reflect the features in the state that the policy relies on.



(a) Importance vector of `hand` (b) Importance vector of `dealer`

FIGURE 6.20. Sensitivity analysis on the Blackjack environment.

Consider a one-step MDP with the SCM shown in Fig. 6.21, where the state $\mathbf{S} = [S_1, S_2]$, $S_i \in [-1, 1]$, $i = 1, 2$, and the action $a \in [-1, 1]$. The reward is defined as $R(\mathbf{S}, a) = 100 \times S_2 + a \times S_1$. Under this setting, the optimal policy is:

$$A = \begin{cases} -1 & S_1 < 0 \\ 1 & \text{otherwise} \end{cases}$$

Intuitively, the policy selects the minimum value in the action space when S_1 is negative , and the maximum value otherwise.

The action-based importance method correctly identifies S_1 as more important, as the policy only depends on S_1. However, the Q-value-based method produces a different result. In a one-step MDP, the Q-function is the same as the reward function. As the coefficient in the Q(reward) function is larger for S_2, the Q-value-based method finds S_2 more important, which is different from the features that the policy relies on.

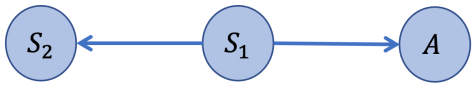


FIGURE 6.21. The skeleton of SCM of the one step MDP.

7

Causal Path-Specific Importance in SCM

7.1. Introduction

Path-specific effect analysis, which quantifies the strength of pathways linking a decision to its outcome, is an important topic in statistical causal inference. This approach has been widely adopted across various disciplines, including social psychology [**151**], medicine [**152**], and economics [**153**], proving to be a valuable tool for analyzing complex systems. Recently, with the advancement of artificial intelligence, path-specific effect analysis has been employed to provide explanations for decision-making mechanisms in models and design algorithms that promote fair predictions [**6**, **154**, **155**].

As an illustration of the significance of the path-specific effect, consider the structural causal model (SCM) in Figure 7.1, which describes a hiring process based on an applicant's gender, number of children, physical strength, and qualifications [**156**]. The effect of gender (X) on the hiring decision (Y) can be decomposed into three different pathways starting from X and ending in Y: the direct impact of gender (unfair) $\pi_1 : X \to Y$, the impact mediated by the number of children $\pi_2 : X \to C \to Y$ (unfair, as it also discriminates against women who have children), and the impact mediated by physical strength $\pi_3 : X \to M \to Y$ (fair). It is essential to measure each path's effect in the SCM to analyze the hiring process's fairness. The importance of the path-specific effect is also demonstrated in other domains, such as protein signaling networks, which provides insight into how signaling molecules influence subsequent molecules in the cascade [**152**], and biological pathways of symptoms, where it elucidates the impact of a disease on symptoms [**157**]. Identifying and differentiating specific pathways can contribute to a deeper understanding of complex diseases and may lead to new insights into disease mechanisms.

However, using the classical definition of the path-specific effect [**158**] has limitations in assessing the fairness of SCM. This is because it only provides an average effect estimate at the population

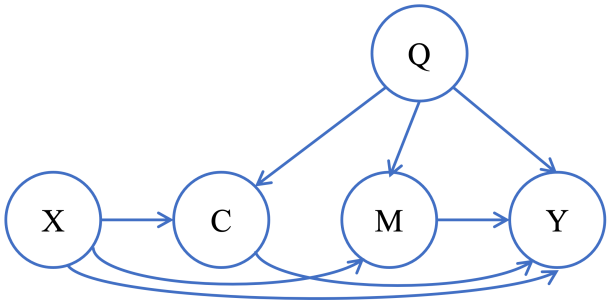


FIGURE 7.1. The causal graph of job hiring for the physical demanding job. X: gender, C: the number of children, M: physical strength, Q: qualification, Y: hiring probability

(type) level without considering individual-level impact (token). For example, the impact of gender on hiring ($X \to Y$) can be very small on average for the population, but it can be large for an individual.

To overcome this limitation, we present a novel definition of the causal counterfactual path-specific importance score. The proposed definition has three major advantages. Firstly, it can quantify individual-level impact through a specific pathway from the source vertex to the target vertex. Secondly, for paths with multiple edges, the effect can be decomposed into individual edges, making it cognitively intuitive for humans to understand. Finally, the computation of the proposed definition is efficient compared to the classical path-specific effect calculation, which is computationally demanding due to the need to evaluate each path independently. In complex causal graphs where the number of pathways grows exponentially with the number of edges, evaluating all pathways from the source vertex to the target vertex can be expensive. Our proposed path-specific importance score has desirable mathematical properties and can be efficiently computed through our designed algorithm.

In summary, we define a causal counterfactual path specific importance score, which quantifies the individual-level impact of a specific path from the source variable to the target variable. We show that our metric has desirable mathematical properties, including adherence to chain rules and consistency. Additionally, we present an algorithm that can efficiently compute the scores of all paths from the source vertex to the target vertex and identify the k-most significant paths in a causal graph with the highest scores.

The structure of this paper is as follows. Section 2 introduces preliminary knowledge related to causality. Section 3 presents our definition of path-specific counterfactual importance score. In Section 4, we establish the mathematical properties of the importance score. In Section 5, we propose an efficient algorithm to identify the most important path. Section 6 provides the evaluation of our method, while Section 7 presents related works in the field. Section 8 offers a detailed discussion of our findings. We summarize our work in Section 9. Lastly, the appendix provides proof of theorems and lemmas.

7.2. Preliminaries

We introduce the notation used to express concepts and variables. Capital letters, such as X, are utilized to represent random variables, while small letters, such as x, are utilized to denote the realizations of these variables. Bold letters, such as $\mathbf{X}$, are used to denote vectors of random variables. Calligraphic letters, such as $\mathcal{X}$, are used to represent sets. For a given natural number n, the $[n]$ is defined as the set $\{1, 2, \cdots, n\}$.

Causal Graph and Skeleton. Causal graphs are probabilistic graphical specifically constructed to depict data-generating processes [**132**]. In the graph, each vertex represents a variable. With a given variable set denoted as $\mathcal{V} = \{V_i, i \in [n]\}$, a directed edge from variable V_j to V_i implies that V_i reacts to modifications in V_j, with all other variables maintaining a constant value. Direct causes of V_i, or its parent variables, are defined as those linked to V_i through directed edges, denoted by the set $\mathcal{P}a_i$. The underlying structure of a causal graph, commonly referred to as its skeleton, is determined by the graph's overall topology.

Structural Causal Models (SCM). The Structural Causal Model (SCM) introduces a framework in which the value of each variable V_i can be determined as a function of both its parent variables and exogenous variables within a causal graph. This formalization proceeds as follows: let $\mathcal{V} = \{V_i, i \in [n]\}$ denote a set of endogenous (observed) variables, and let $\mathcal{U} = \{U_i, i \in [n]\}$ stand for a set of exogenous (unobserved) variables. An SCM is then established through a series of structural equations [**132**] :

$$V_i = f_i(\mathcal{P}a_i, U_i), \mathcal{P}a_i \subset \mathcal{V}, U_i \subset \mathcal{U}, i \in [n], \tag{7.1}$$

where function f_i signifies the causal mechanism of V_i, thereby determining the value of V_i contingent upon its parent variables and the exogenous variables.

Intervention and Do-operation. The intervention of a variable on an SCM, denoted by the $do(\cdot)$ operator, refers to assigning a constant value v to V_i, irrespective of its structural equation in the SCM. $do(V_i = v)$ means setting the value of V_i to a constant v regardless of its structural equation in the SCM, i.e., ignoring the edges into the vertex V_i. This equates to disregarding the edges leading into the vertex V_i in the causal graph. After intervention $do(X = x)$, the distribution of Y is denoted by $P(Y_{X=x} = y)$, where the variable $Y_{X=x}$ is the value of Y after intervention $do(X = x)$.

Path. Given a graph G, a path from vertex X to vertex Y in G is a finite sequence of edges $\pi = \{XV_1, V_1V_2, \cdots V_{m-2}V_{m-1}, V_{m-1}Y\}$ in which any two consecutive edges are adjacent, and all vertices are distinct.

Counterfactual Reasoning. Counterfactual reasoning enables the exploration of hypothetical "what if" scenarios. Consider an observation O referring to the realization of variables in the causal graph, and $X = x$ being given, the counterfactual question is asking the potential outcome if X were assigned a different value x' [**132**]. The counterfactual outcome of Y is represented as $Y_{X=x'}|X = x, O = o$. With a given SCM, deterministic counterfactual reasoning can be accomplished through intervention, as follows:

(1) Recover the exogenous variable value U as u, via the structural functions and the values $X = x$, $O = o$;

(2) Calculate the counterfactual outcome $Y|do(X = x'), U = u$. Specifically, within the SCM, assign the value of X to x' and substitute all exogenous variable values U as u on the right side of each structural function to obtain the value of Y.

The two steps are called **deterministic counterfactual reasoning** [**143**], because the value of U can be solved uniquely from f when X and O are given. If the exogenous variable U in step 1 has multiple solutions, then nondeterminism should be involved in causal models by assigning a prior probability $P(U = u)$. In the **nondeterministic counterfactual reasoning** [**143**], step 1 is to update $P(U)$ as $P(U|X = x, O = o)$, and the counterfactual outcome in step 2 is defined as the expectation over the posterior distribution of U.

7.3. Causal Effect along Different Pathways

We consider an SCM that represents the causal relationships among a set of variables $\mathcal{V} = \{V_i, i \in [n]\}$ that are given. All structural functions $f_i, i \in [n]$ are assumed to be continuous, with bounded derivatives on their respective domains. Additionally, we assume the corresponding causal graph G is a directed acyclic graph (DAG), as an example depicted in Figure 7.1.

Our primary objective is to examine the influence of a source vertex X on a target vertex Y, where $X, Y \in \mathcal{V}$. The most prevalent definition of a causal effect is applied within the medical field for binary cases, where the outcome variable Y depends on whether a patient accepts the diagnosis D or not. In this context, the causal effect is defined as the expected difference between these two scenarios within the population, denoted by $\mathbb{E}[Y_{D=1}] - \mathbb{E}[Y_{D=0}]$. Building upon this concept, to evaluate the causal effect at an individual level, allowing the algorithm to quantify the impact for a specific data point, we introduce the total counterfactual importance score described in Definition 2. This method incorporates counterfactual reasoning and a perturbation value δ in the denominator of Eq. 7.2.

Definition 2. *(Total Counterfactual Importance Score) Given a factual observation* $O : \{V_i = v_i | i = 1, \ldots, n\}$*, which is a realization of all endogenous variables, for* $\forall X, Y \in \mathcal{V}$*, the counterfactual importance score of* X *on* Y *is*

$$w_{X \to Y|O} = \lim_{\delta \to 0} \frac{\mathbb{E}[Y_{X=x+\delta}|O] - \mathbb{E}[Y_{X=x}|O]}{\delta}, \tag{7.2}$$

where $X = x$ *is the factual observation from* O*.*

The numerator represents the difference between two counterfactual outcomes of Y in the situations X is setup to X or $x+\delta$ when given the observation O. By incorporating the denominator δ and the limit operation in Eq. 7.2, this definition intuitively captures how Y responds to the change on X given the current observation O. A larger value of effect scores for a path indicates greater significance for that path.

Next, we will discuss how to perform an intervention on a variable along a path. Furthermore, we introduce the concept of the path-specific counterfactual importance score for a given path π, which starts from variable X and ends at variable Y. This score quantifies the causal effect from

the source variable to the target variable along a specific path. A direct edge can be considered a special case of a path.

DEFINITION 3. *(Intervention a variable along a path) For an SCM M, given a path (or an edge, a subgraph) π, we can partition each vertex V_i 's parents into two parts $Pa_i = Pa_i(\pi) \cup Pa_i(\tilde{\pi})$, where $Pa_i(\pi)$ represents those members of Pa_i that are linked to V_i in π, $Pa_i(\tilde{\pi})$ represents the complementary set. The operation of intervention of $X = x$ on the path π is defined as: we replace the structural equation f_i as $f_i(Pa_i, u)^{\pi} = f_i(Pa_i(\pi)_{X=x}, Pa_i(\tilde{\pi}), u)$, where $Pa_i(\pi)_{X=x}$ represents it takes the value when $X = x$ is enforced. The outcome of Y after the intervention of $X = x$ on the path π can be represented as $Y_{X=x|\pi}$.*

DEFINITION 4. *(Path-Specific Counterfactual Importance Score) Given a factual observation $O : \{V_i = v_i | i = 1, \ldots, n\}$, which is a realization of all endogenous variables, and a causal path π in graph G, the path-specific counterfactual importance score of source vertex X on target vertex Y along path π is defined as follows*

$$w_{X \xrightarrow{\pi} Y|O} = \lim_{\delta \to 0} \frac{\mathbb{E}[Y_{X=x+\delta|\pi, X=x|\tilde{\pi}}|O] - \mathbb{E}[Y_{X=x}|O]}{\delta}. \tag{7.3}$$

Here, $X = x$ is the factual observation. The term $\tilde{\pi} = \Pi_{X \to Y} \backslash \pi$ represents the set of all paths from X to Y excluding path π, where $\Pi_{X \to Y}$ is the set of all paths from X to Y. When the path is a direct edge from X to Y, the notation $w_{X \xrightarrow{\pi} Y|O}$ can be simplified as $w_{l_{XY}|O}$. The first term in the numerator represents the counterfactual outcome of Y when X is fixed at $x + \delta$ if X is part of path π; otherwise, it remains fixed at x.

The numerator captures the effect of X on Y through the given π. Definition 4 can quantify how Y responds to the change on X only through a path π given the current observation O. In our model, the direct effect is defined as the effect propagated only through the edge directly connecting the source and target, which can be regarded as a special case of path-specific effect.

Intuitively, Definition 4 uses the intervention operation on a path and only considers the effect of the changing of X on Y propagating from the given path π. We will keep X as its original value for other edges in the graph. Intuitively, if the change caused by the intervention is larger, the causal strength of X to Y on these paths is larger. The operation of intervention on a path

isolates the impact on Y by X on the given path. The normalization and limitation operation on the intervention value provides some "independence" among edges of the causal effect in a path. These exhibit desirable properties.

The numerator is similar to the definition of the counterfactual path-specific effect in [**155**]. However, they focus on the discrete case. Although [**155**] can be directly extended to the individual level, it doesn't have the math properties we propose in the next section. Our definition adds the normalization of the intervention value in the denominator and focuses on the strength of the local effect by limitation operation. Intuitively, these modifications provide some "independence" of the causal effect in the graph. So, our method has more straightforward decomposition and fast calculation properties (discussed in the next section), which are important for explaining and evaluating the causal effect along each path.

The limit format of the perturbation value δ has a similar format as incremental causal effect and marginal treatment effect [**94**, **95**, **96**]. Ours are defined on counterfactual reasoning and path-specific effect. However, they focus on the total or direct effect.

7.4. Properties of Path-Specific Counterfactual Importance Score

As previously stated in the introduction, the traditional definition of path-specific effect encounters difficulties in decomposing its effect to individual edges and computing it efficiently. This section delves into the mathematical properties of our path-specific counterfactual importance score, which addresses these limitations.

In detail, we discuss the connection between our definition and the partial derivative involved in the calculation, as well as the consistency between the total importance score and path-specific score. First, we introduce an assumption and a lemma pertinent to the direct effect, which refers to an edge that directly connects the source and target vertex within the graph.

LEMMA 7. *For an SCM M, given an edge from X to Y, and the structural causal equation of Y: $Y = f_Y(X, \mathbf{Z_{yx}}, U_y)$, where $\mathbf{Z_{yx}}$ represents all parents of Y excluding X. Given a factual observation O, the counterfactual importance score of X on Y along the path/link $\pi = l_{XY} = \{XY\}$*

is the partial derivative of Y with respect to X at the point of the observation. Formally,

$$
\begin{aligned}
w_{l_{XY}|O} &= \mathbb{E}_{U_y|O}\left[\left.\frac{\partial Y}{\partial X}\right|_{O}\right] \\
&= \mathbb{E}_{U_y|O}\left[\left.\frac{\partial Y}{\partial X}\right|_{X=x,\mathbf{Z_{yx}}=\mathbf{z_{yx}},U_y=u_y}\right] \\
&= \int \left.\frac{\partial Y}{\partial X}\right|_{X=x,\mathbf{Z_{yx}}=\mathbf{z_{yx}},U_y=u_y} P(u_y|O)du_y \qquad (7.4)
\end{aligned}
$$

The Lemma 7 does not require the exogenous variables $\mathcal{U}$ to be deterministically recovered (have a unique solution when f, X, O are given). The derivation is based on nondeterministic counterfactual reasoning. It shows that the value of the path-specific counterfactual importance score in an edge case is the partial derivative of Y concerning X in the counterfactual reasoning data point weighted by the posterior probability of $\mathcal{U}$. Additionally, if exogenous variables $\mathcal{U}$ can be perfectly recovered as $\mathbf{u}$, given a factual observation $O : \{V_j = v_j | j = 1, \ldots, n\}$, then equation (7.4) in Lemma 7 can be simplified as eq. (7.5). The detailed proof of Lemma 7 can be found in Appendix 7.10.

$$
w_{l_{XY}|O} = \left.\frac{\partial Y}{\partial X}\right|_{O} \qquad (7.5)
$$

We introduce the following assumption about exogenous variables $\mathcal{U}$.

ASSUMPTION 1. ***(Independent exogenous variables assumption)*** *All endogenous variables $\mathcal{U}$ are independent.*

Based on Lemma 7, when Assumption 1 holds true, we can derive the following theorem for a general path in the causal graph, which shows that the calculation of the importance score for each edge in the graph is independent, and the importance score for a path is the multiplication of the importance score for each edge inside the path. Our path-specific score can be directly decomposed to individual edges. This is helpful to understand the effect of these edges on the outcome and design effective algorithms for calculating the importance score in the graph. The detailed proof of Theorem 7.4.1 is in Appendix 7.10.

THEOREM 7.4.1. ***(Chain rule: path-specific counterfactual importance score and partial derivative)*** *For an SCM M, when Assumption 1 holds true, given a path $\pi = \{XV_1, V_1V_2, \cdots V_{m-2}V_{m-1}, V_{m-1}Y\}$ with length m and a factual observation O, the path-specific counterfactual importance score of the source vertex X on target vertex Y along the path π can be represented as:*

$$\begin{aligned} w_{X \xrightarrow{\pi} Y|O} &= w_{l_{XV_1}|O} \left(\prod_{i=1}^{m-2} w_{l_{V_iV_{i+1}}|O} \right) w_{l_{V_{m-1}}Y|O} \\ &= \mathbb{E}_{U_1|O} \left[\left. \frac{\partial V_1}{\partial X} \right|_O \right] \left(\prod_{i=1}^{m-2} \mathbb{E}_{U_{i+1}|O} \left[\left. \frac{\partial V_{i+1}}{\partial V_i} \right|_O \right] \right) \\ &\quad \mathbb{E}_{U_Y|O} \left[\left. \frac{\partial Y}{\partial V_Y} \right|_O \right]. \end{aligned}$$

We also show the consistency of our definition in Theorem 7.4.2. The sum of counterfactual path-specific scores among all paths from source to target is the same as the total counterfactual importance score from source to target. It provides the connection between Definition 2 and Definition 4. This makes the addition and comparison among different paths' scores more meaningful. The detailed proof is in Appendix 7.10.

THEOREM 7.4.2. *(**Consistency**) Given an SCM M and a factual observation O, when Assumption 1 holds true, the total counterfactual importance score of X on Y equals the summation of path-specific counterfactual importance score among all the possible paths from X to Y:*

$$w_{X \to Y|O} = \sum_{\pi \in \Pi_{X \to Y}} w_{X \xrightarrow{\pi} Y|O}.$$

7.5. Efficient Algorithm for the Most Important Path

In reality, given an SCM, people are more interested in finding the most important path that has the largest impact. We define the **most important paths** from the source to the target as the path with the largest absolute value of the Importance score. We introduce algorithm Algorithm 13 to find the most important path effectively. We note that this algorithm can be easily extended to the k-most important path, by using a priority queue.

Based on Theorem 7.4.1, the calculation for the importance score for each edge is independent. So, we can evaluate each edge's score first and then calculate the path's score. Then, we notice that

finding the most important path in the original causal graph can be reduced into finding the longest path in a DAG with different edge weights. In detail, based on Theorem 7.4.1, the importance score for a path is the multiplication of the importance score for each edge inside the path. We denote the importance score of all edges in a graph as $w_1, \cdots, w_n$, where n is the number of edges. Because of the monotonicity of the log function, $\prod_{i=1}^{n} |w_i|$ achieves the largest value as the $log(\prod_{i=1}^{n} |w_i|)$. Then we can transfer it from the multiplication to the summation as follows:

$$log(\prod_{i=1}^{n} |w_i|) = \sum_{i=1}^{n} log(|w_i|)$$

Next, we define a new graph G' with the same skeleton as the original graph G, but the length for edge i is $\log(|w_i|)$, rather than w_i, $i \in [n]$. For each path in G' with a length of $\sum_{i=1}^{n} \log(|w_i|)$, there exists a corresponding path in G with an absolute importance score of $\prod_{i=1}^{n} |w_i|$. Given the monotonicity of logarithms, the most important path in the original graph G corresponds to the longest path in the transformed graph G'. Finally, Algorithm 13 calls a subroutine Algorithm 14 to calculate the longest path in a DAG [**159**]. Because of the acyclic property, Algorithm 14 can find the longest path in polynomial time by topological sort and dynamic programming.

Our methodology retains its applicability even when $|w_i|$ is less than 1. If $|w_i|$ is found within the range of 0 to 1, then the associated edge length $\log(|w_i|)$ in the graph G' is a negative value. As we presuppose the causal graph to be acyclic, issues related to negative cycles are avoided, and the validity of our longest path computation is preserved. Intuitively, when $|w_i|$ lies between 0 and 1, in the original graph G, the expansion of a path by an edge results in a decrease in the path's importance score. Correspondingly, in the graph G', a negative path $\log(|w_i|)$ contributes to a decrease in path length. Consequently, the consistent correlation between these observations validates our approach.

The time complexity of Algorithm 13 is in the linear scale of number of edges and vertices, $O(|E| + |V|)$ [**160**]. Compared with the case using the classical definition, where the calculations for each path are independent, and the number of paths is exponential to the edge, our algorithm significantly improves finding the most important path in the causal graph.

Because of the math property that the edge's scores are independent, we can easily formulate our definition of the path-specific score into the existing algorithm that is based on graph theorem

Algorithm 13 Fast Calculation for the Most Important Path

Inputs: Graph G(V, E), source vertex X, target vertex Y
Outputs: Path P
Calculate the path-specific importance score for each edge in the graph: $w_1, w_2, \cdots, w_n$, where $n = |E|$
Transfer the importance score to log scale, $log(|w_1|), log(|w_2|), \cdots, log(|w_n|)$
Define a graph G' that maintains same skeleton as graph G, with edge weight $log(|w_1|), log(|w_2|), \cdots, log(|w_n|)$
Path P = LongestPathDAG(G', X, Y)
return P

Algorithm 14 Longest Path in DAG

Inputs:Weighted DAG graph $G(V, E)$, source vertex X, target vertex Y
Outputs: Path P
Linearized order $\mathcal{V}$ = Topologically sort G
for each vertex $v \in \mathcal{V}$ in linearized order from X to Y **do**
    $dist(v) = \max_{u,v \in E}(dist(u) + w(u, v))$
    $v_{previous} = argmax_{u,v \in E}(dist(u) + w(u, v))$
end for
Recursively find the path using $v_{previous}$ starting from X to Y
return P

to design an efficient algorithm. For example, we can use the algorithm in [**161**] to find the paths with the top k largest importance score in a causal graph.

7.6. Evaluation

We demonstrate the performance of our method for path-specific analysis using two datasets, one synthetic and the other real.

7.6.1. Job Hiring Problem. We assume the following SCM for the physically demanding job hiring process discussed in the introduction (illustrated in Figure 7.1).

$$X = U_x, U_x \sim Bernoulli(0.6)$$

$$Q = U_Q, U_Q \sim \mathcal{N}(2, 5^2)$$

$$C = X + 0.5QU_c, U_c \sim Tr\mathcal{N}(2, 1^2, 0.1, 3.0)$$

$$M = 3X + 0.4QU_M, U_M \sim Tr\mathcal{N}(3, 2^2, 0.1, 3.0)$$

$$Y = h(X, Q, C, M) = \zeta(-10 + 5X + C + Q + M)$$

where $Bernoulli$, $\mathcal{N}$, and Tr$\mathcal{N}$ represent the Bernoulli, Gaussian, and truncated Gaussian distributions, respectively, and $\zeta(k) = 1/(1 + exp(-k))$ denotes the standard sigmoid function. A similar data generation process was employed in [**156**].

The simulation results are presented in Figure 7.2. The first three figures display population-level results. Specifically, they illustrate the distribution of importance scores for three distinct paths, originating from gender and ending in hiring, which quantify the impact of X on Y. For the majority of individuals (over 80%), the effect of these three paths is small, indicating that the model appears fair on average. This result aligns with the classical definition at the population level.

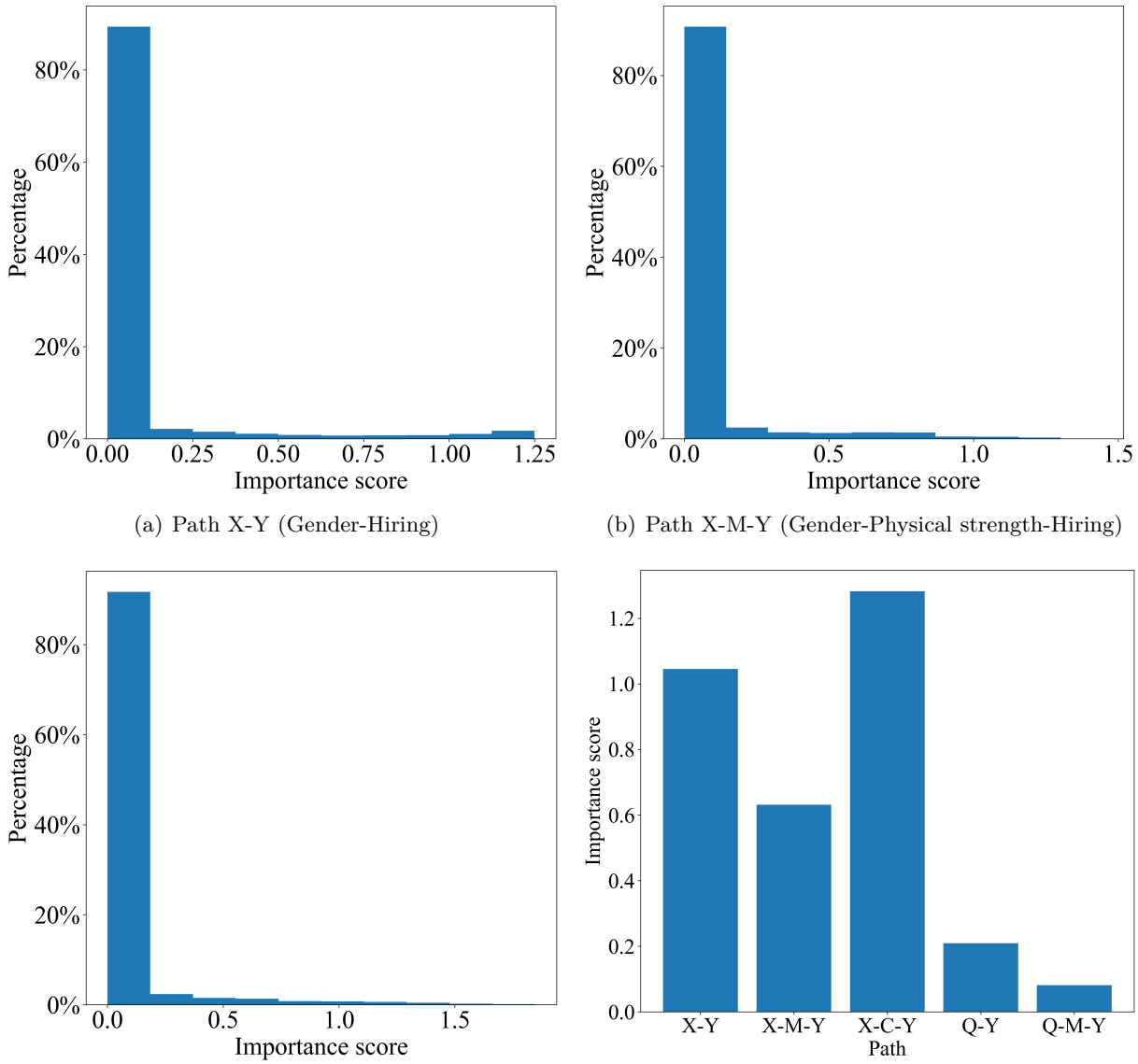
In the final subfigure, we examine a specific individual and assess the impact of all paths related to hiring decisions of Figure 7.1 for this particular individual. We observe that the hiring process is unfair for this individual, as the impact of the unfair paths $X \to Y$ and $X \to C \to Y$ possess importance scores comparable to other paths. This example highlights how our approach can be utilized to quantify the influence of paths for individuals.

To further investigate the variations in importance scores for paths within the population and at the individual level, we introduce a weight coefficient, w_c corresponding to the 'number of children' feature, into the equation for Y. An increase in the value of w_c signifies an amplified impact of feature C on Y.

$$Y = h(X, Q, C, M) = \zeta(-10 + 5X + w_c C + Q + M)$$

Our analysis results are depicted in Figure 7.3. The first figure presents the distribution of importance scores for path $X - C - Y$. The second figure relates to the individual-level importance scores. We note that as the value of w_c increases, the importance score for path $X - C - Y$ increases correspondingly, while the scores for other paths remain almost unchanged. This indicates an increased level of unfairness associated with this specific path, in line with the changes in w_c.

7.6.2. Smoking Impact Problem. We also evaluate our method using a real dataset - the Framingham Heart Study dataset [**162**]. The original Framingham cohort dataset comprises two years of examination data for 5,209 participants aged between 30 and 62 years. Our objective is to



(a) Path X-Y (Gender-Hiring)

(b) Path X-M-Y (Gender-Physical strength-Hiring)

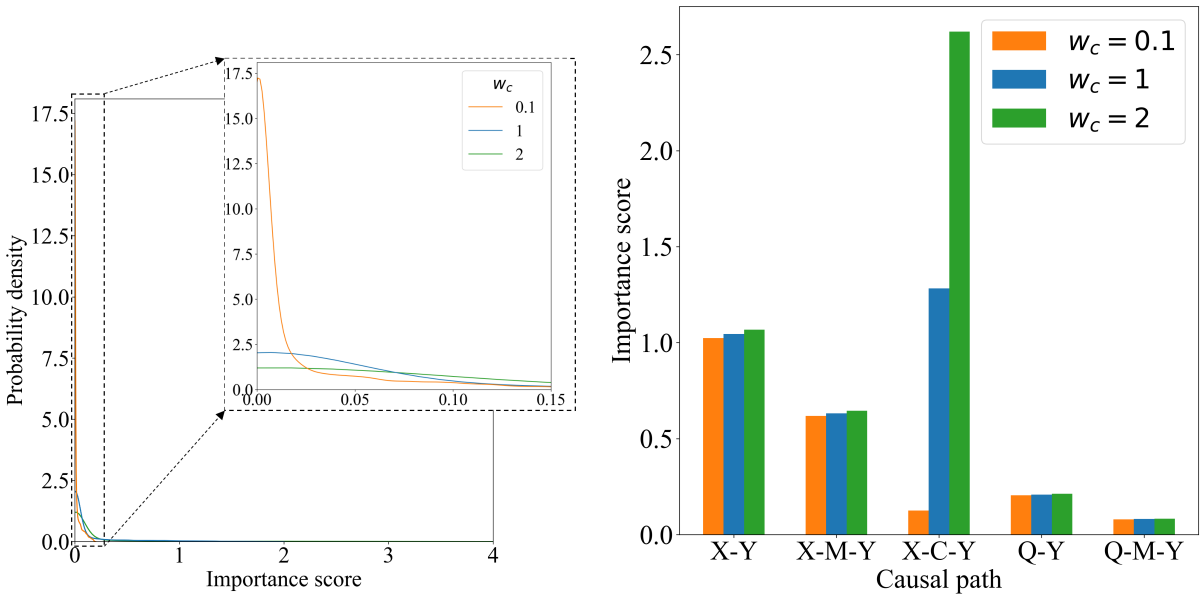
(c) Path X-C-Y (Gender-Num of children-Hiring)

(d) Important score for an individual

FIGURE 7.2. The causal counterfactual path-specific importance score in job hiring problem.

investigate the causal mechanisms of smoking behavior on systolic blood pressure (SBP) mediated by cholesterol levels and body weight (depicted in Figure 7.4). Our SCM is formulated as a linear regression model with some covariant terms, as recommended by [**163**].

Figure 7.5 presents the average effect of the paths that begin with smoking and end with at the population level by using our method. Our method reveals that the direct impact of smoking on



(a) The distribution of importance score for path X-C-Y with different values of w_c

(b) Importance score for an individual with different values of w_c

FIGURE 7.3. The causal counterfactual path-specific importance score in job hiring problem with different values of w_C

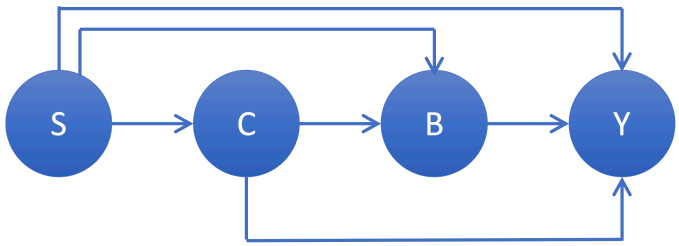


FIGURE 7.4. The causal graph of smoking effect on blood pressure. S: smoking behavior, C: cholesterol level, B: body weight, Y: systolic blood pressure (SBP).

blood pressure, denoted as $(S \to Y)$, has the highest average importance score in comparison to the other three paths. Consequently, the direct path's influence is the most dominant factor relative to other paths. Notably, the ranking for the paths sorted by their importance based on our method is consistent with the conclusion drawn in [**163**]. We believe the difference in the magnitude of the results is due to the different perturbation values used by the two methods.

The first four subfigures of Figure 7.6 display the distribution of importance scores along the paths that begin with smoking and end with SBP at the population level. Additionally, the last two subfigures show the individual-level importance of these paths for two specific individuals.

Interestingly, we find that for person A, both the direct path from smoking to blood pressure ($S \to Y$) and the path of smoking affecting blood pressure through body weight ($S \to B \to Y$) are significant. However, for person B, the direct impact of smoking on blood pressure ($S \to Y$) is the dominant factor. This type of individual-level path analysis provided by our method can help an individual know each path's influence for the particular individual.

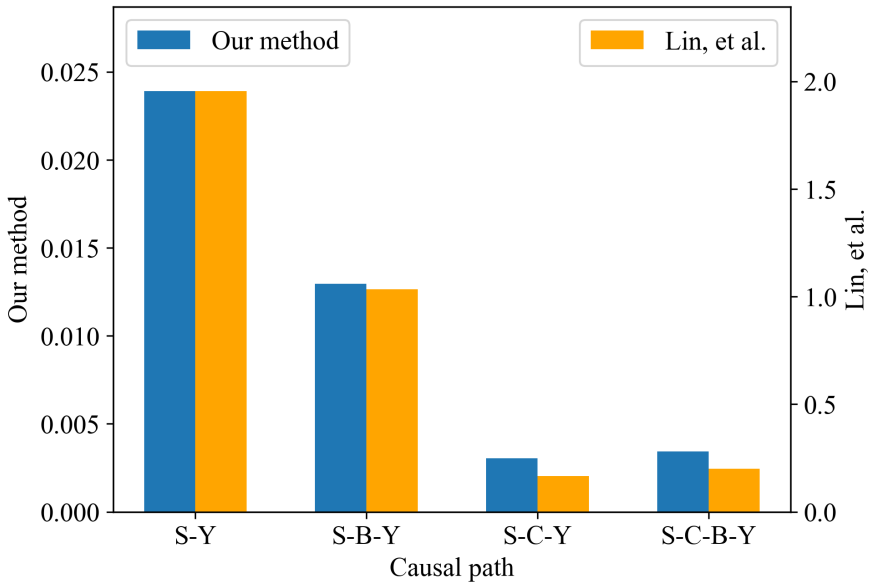
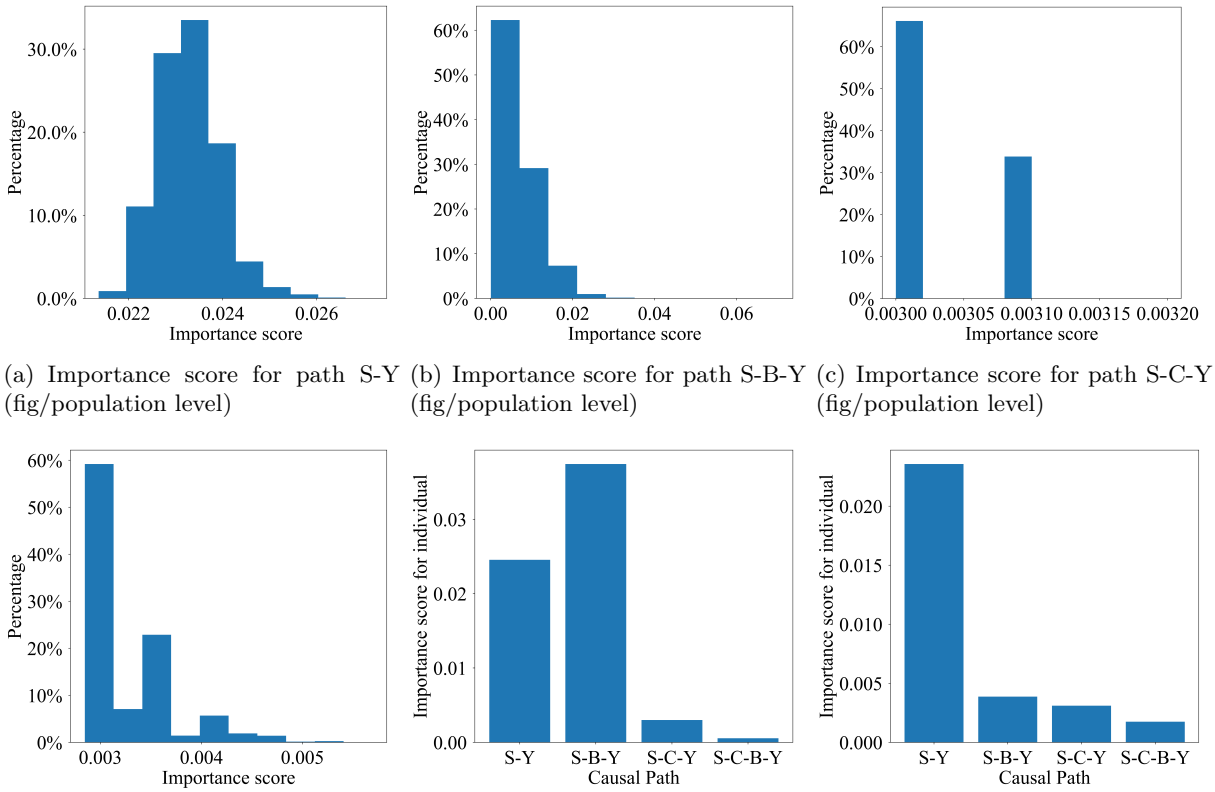


FIGURE 7.5. Average effect for each path at the population level.

7.7. Related Works

The study of path-specific analysis has garnered significant interest within the academic community. A majority of the existing research has centered around non-parametric settings, identifiability, or informational decomposition of SCMs [**91**, **92**, **93**]. These approaches diverge from our work, which is grounded in a known SCM model. Our method enables the quantification of individual-level path-specific effects and the development of efficient algorithms to identify the most crucial paths.

Several related studies have proposed causal effect definitions such as incremental causal effects and marginal treatment effects [**94**, **95**, **96**]. These definitions employ a similar constraint format



(a) Importance score for path S-Y (fig/population level)

(b) Importance score for path S-B-Y (fig/population level)

(c) Importance score for path S-C-Y (fig/population level)

(d) Importance score for path S-C-B-Y (fig/population level)

(e) Importance score for individual (person A)

(f) Importance score for individual (person B)

FIGURE 7.6. The causal counterfactual path-specific importance score in smoking impact problem.

for the perturbation value δ. However, while their focus lies on total effects and population-level analysis, our work concentrates on path-specific effects and offers individual-level quantification.

We elucidate the relationship between our work and existing literature concerning population-level causal effects. In particular, the studies by Janzing et al. [**97**] and Wang et al. [**99**] both define and measure causal strength (effect) using Shapley values. Janzing et al. [**98**] assess causal influence based on the distributional change resulting from removing a "causal arrow". Furthermore, Janzing's concept of intrinsic causal contribution [**97**] allows for the separation of intrinsic information added by each vertex from the information obtained from its ancestors. Wang et al. [**99**] allocate credit to edges and provide an interpretation of the entire causal graph. In contrast, our method builds upon the classical definition of average treatment effect, which is grounded in intervention operations, and extends this definition to continuous and individual cases. Our approach

boasts advantages such as the clear decomposition of causal effects along paths and rapid calculations for identifying the most important paths. This transparent decomposition proves critical for interpreting the entirety of the SCM.

7.8. Discussions

Motivation for Path-specific Effects: In certain applications, the analysis of path effects yields vital information that is not captured by total effects. Notable examples include the fairness analysis of a model as discussed in the introduction, protein signaling networks (which examine the influence of signaling molecules on subsequent molecules within the cascade) [**152**], and the biological pathway of symptoms (exploring the impact of a disease on its symptoms) [**157**]. Identifying and distinguishing specific pathways can enhance our understanding of complex diseases and potentially offer novel insights into disease mechanisms.

Linear Case: Based on our definition, when all structural equations are linear, the individual-level effect is equivalent to the population-level effect, as the difference between the counterfactual after perturbation and the original is independent of $\mathcal{U}$. Consequently, for each edge, the path-specific score is the partial derivative of the structural function, which corresponds to the coefficient of the linear function. Simultaneously, using Theorem 1, the path-specific score for a given path is determined by the product of each variable's coefficient within its respective linear model along that path.

Assumptions: We do not make the assumption that $\mathcal{U}$ is perfectly recovered, which is only necessary for deterministic counterfactual reasoning and facilitates the calculation of the expectation over $\mathcal{U}$. The only requirement is the independence of $\mathcal{U}$ (Assumption 1) . Without perfectly recovered of $\mathcal{U}$, our definitions and theorems remain valid, and our method can still be employed based on nondeterministic counterfactual reasoning, as evidenced by our experiments.

Single-point Derivatives and Potential for Severe Errors: Our objective is to quantify the local effect at a specific point; hence, a method that concentrates on a single point preserves generality. We recognize that the product of effects along a causal path may lead to error propagation and amplification, a common issue when estimating a path's strength. This arises because the

target vertex can be represented as a multi-layer composition function of the source vertex through the causal graph structure. We intend to explore this challenge in future work.

7.9. Conclusion

In this study, we present a novel metric for path-specific effect analysis, termed the causal counterfactual path-specific importance score. This score quantifies path-specific effects at the individual level. We demonstrate that our metric exhibits desirable mathematical properties, such as compliance with chain rules and preservation of consistency. These properties facilitate the decomposition of the path-specific score into each edge within the path, enabling the development of an efficient algorithm to identify the most critical path in a causal graph in linear time with respect to the number of edges. Our simulations indicate that our innovative definition concurs with the classical definition at the population (type) level while also delineating path-specific effects at the individual (token) level.

7.10. Appendix

We first introduce Lemma 8, which pertains to the properties based on our assumption of the structural function. Lemma 8 demonstrates that if f is continuously differentiable and possesses a bounded derivative of Y with respect to X, then the expectation operation (integration) and limit (derivative) on f are exchangeable.

LEMMA 8. *Let $U \in \mathbb{U}$ be a random variable. Suppose that $f(x, U)$ is continuously differentiable with respect to x for all $U \in \mathbb{U}$ and integrable on U for all $x \in \mathbb{R}$. If the partial derivative $|\frac{\partial}{\partial x} f(x, U)|$ is bounded by a constant B for all $x \in \mathbb{R}, U \in \mathbb{U}$,*

$$\left| \frac{\partial}{\partial x} f(x, U) \right| \leq B,$$

then

$$\begin{aligned}\frac{\partial}{\partial x}\mathbb{E}\big[f(x,U)\big] &= \frac{\partial}{\partial x}\int_U f(x,U)dF(U)\\ &= \int_U \frac{\partial}{\partial x}f(x,U)dF(U)\\ &= \mathbb{E}\bigg[\frac{\partial}{\partial x}f(x,U)\bigg],\end{aligned}$$

where F is the cumulative distribution function of U.

PROOF. From the mean value theorem, there exists $x' \in (x, x+\delta)$ such that $\frac{f(x+\delta,U)-f(x,U)}{\delta} = \frac{\partial}{\partial x}f(x',U)$. Then

$$\begin{aligned}\frac{\partial}{\partial x}\mathbb{E}\big[f(x,U)\big] &= \frac{\partial}{\partial x}\int_U f(x,U)dF(U)\\ &= \lim_{\delta\to 0}\int_U \frac{f(x+\delta,U)-f(x,U)}{\delta}dF(U)\\ &= \lim_{\delta\to 0}\int_U \frac{\partial}{\partial x}f(x',U)dF(U).\end{aligned}$$

Since $|\frac{\partial}{\partial x}f(x',U)| \le B$ for all $x' \in (x, x+\delta)$, from dominated convergence theorem [**164**], we have

$$\begin{aligned}\frac{\partial}{\partial x}\mathbb{E}\big[f(x,U)\big] &= \lim_{\delta\to 0}\int_U \frac{\partial}{\partial x}f(x',U)dF(U)\\ &= \int_U \lim_{\delta\to 0}\frac{\partial}{\partial x}f(x',U)dF(U)\\ &= \int_U \frac{\partial}{\partial x}f(x,U)dF(U)\\ &= \mathbb{E}\bigg[\frac{\partial}{\partial x}f(x,U)\bigg].\end{aligned}$$

□

7.10.1. Proof of Lemma 1. According to our assumption about f and U, we can have following results by using Lemma 8 in Definition 4.

PROOF.

$$
\begin{aligned}
w_{l_{XY}|O} &= \lim_{\delta\to 0} \frac{\mathbb{E}[Y_{X=x+\delta|\pi, X=x|\Pi_{X\to Y}\setminus\pi}|O] - \mathbb{E}[Y_{X=x}|O]}{\delta} \\
&= \lim_{\delta\to 0} \frac{\mathbb{E}[f_Y(x+\delta, \mathbf{z_{yx}}, U_y)|O - f_Y(x, \mathbf{z_{yx}}, U_y)|O]}{\delta} \\
&= \lim_{\delta\to 0} \frac{\int \left(f_Y(x+\delta, \mathbf{z_{yx}}, u_y) - f_Y(x, \mathbf{z_{yx}}, u_y)\right) P(u_y|O) du_y}{\delta} \\
&= \int \lim_{\delta\to 0} \frac{f_Y(x+\delta, \mathbf{z_{yx}}, u_y) - f_Y(x, \mathbf{z_{yx}}, u_y)}{\delta} P(u_y|O) du_y \\
&= \int \left.\frac{\partial Y}{\partial X}\right|_{X=x, \mathbf{Z_{yx}}=\mathbf{z_{yx}}, U_y=u_y} P(u_y|O) du_y \\
&= \mathbb{E}_{U_y|O}\left[\left.\frac{\partial Y}{\partial X}\right|_{X=x, \mathbf{Z_{yx}}=\mathbf{z_{yx}}, U_y=u_y}\right]
\end{aligned}
$$

□

7.10.2. Proof of Theorem 1.

PROOF. We first show Theorem 7.4.1 holds for the path with two edges. Consider a path $\pi' = \{XV_1, V_1V_2\}$, we can show that the impact score of a path can be decomposed to the score of the edge inside the page. The calculation of each edge's score is independent. From the assumption of f_1, f_2 and U_1, U_2,

$$
\begin{aligned}
w_{X\xrightarrow{\pi'}V_2|O} &= \lim_{\delta\to 0} \frac{\mathbb{E}[V_{2X=x+\delta|\pi, X=x|\Pi_{X\to V_2}\setminus\pi}|O] - \mathbb{E}[Y_{X=x}|O]}{\delta} \\
&= \lim_{\delta\to 0}(\mathbb{E}[f_{V_2}(f_{V_1}(x+\delta, \mathbf{z_{v_1x}}, U_1), \mathbf{z_{v_2v_1}}, U_2)|O \\
&\quad - f_{V_2}(f_{V_1}(x, \mathbf{z_{v_1x}}, U_1), \mathbf{z_{v_2v_1}}, U_2)|O])/\delta \\
&= \lim_{\delta\to 0} \mathbb{E}_{U_1,U_2|O}[(f_{V_2}(f_{V_1}(x+\delta, \mathbf{z_{v_1x}}, U_1), \mathbf{z_{v_2v_1}}, U_2) \\
&\quad - f_{V_2}(f_{V_1}(x, \mathbf{z_{v_1x}}, U_1), \mathbf{z_{v_2v_1}}, U_2))/\delta|O] \\
&= \mathbb{E}_{U_1,U_2|O}\Big[\lim_{\delta\to 0}(f_{V_2}(f_{V_1}(x+\delta, \mathbf{z_{v_1x}}, U_1), \mathbf{z_{v_2v_1}}, U_2) \\
&\quad - f_{V_2}(f_{V_1}(x, \mathbf{z_{v_1x}}, U_1), \mathbf{z_{v_2v_1}}, U_2))/\delta|O\Big].
\end{aligned} \tag{7.6}
$$

Next we prove the chain rule $g'(h(x)) = g'(h(x))h'(x)$ for continuously differentiable function g and h. Define

$$\tilde{g}(y) = \begin{cases} \dfrac{g(y) - g(h(x))}{y - h(x)}, & y \neq h(x), \\ g'(h(x)), & y = h(x), \end{cases}$$

which is continuous around x due to the differentiability of g. It's easy to verify that $\forall \delta \neq 0$,

$$\frac{g(h(x+\delta)) - g(h(x))}{\delta} = \tilde{g}(h(x+\delta))\frac{h(x+\delta) - h(x)}{\delta}.$$

Since $\tilde{g}$ is continuous and h is continuously differentiable, $\lim_{\delta\to 0} \tilde{g}(h(x+\delta))$ and $\lim_{\delta\to 0} \frac{h(x+\delta)-h(x)}{\delta}$ exist, which implies that $\lim_{\delta\to 0} \frac{g(h(x+\delta))-g(h(x))}{\delta}$ exists and

$$\lim_{\delta\to 0} \frac{g(h(x+\delta)) - g(h(x))}{\delta} = g'(h(x))h'(x).$$

Applying chain rule into (7.6),

$$\begin{aligned} w_{X \xrightarrow{\pi'} V_2|O} &= \mathbb{E}_{U_1,U_2|O}\Big[\lim_{\delta\to 0}(f_{V_2}(f_{V_1}(x+\delta, \mathbf{z_{v_1x}}, U_1), \mathbf{z_{v_2v_1}}, U_2) \\ &\quad - f_{V_2}(f_{V_1}(x, \mathbf{z_{v_1x}}, U_1), \mathbf{z_{v_2v_1}}, U_2))/\delta|O\Big]. \qquad \text{(Step 1)} \\ &= \mathbb{E}_{U_1,U_2|O}\left[\frac{\partial V_2}{\partial V_1}|_{V_1=v_1, \mathbf{Z_{v_2v_1}}=\mathbf{z_{v_2v_1}}, U_2=u_2}\right. \\ &\quad \left. \times \frac{\partial V_1}{\partial X}|_{X=x, \mathbf{Z_{v_1x}}=\mathbf{z_{v_1x}}, U_1=u_1}\right] \qquad \text{(Step 2)} \\ &= \mathbb{E}_{U_2|O}\left[\frac{\partial V_2}{\partial V_1}|_{V_1=v_1, \mathbf{Z_{v_2v_1}}=\mathbf{z_{v_2v_1}}, U_2=u_2}\right] \\ &\quad \times \mathbb{E}_{U_1|O}\left[\frac{\partial V_1}{\partial X}|_{X=x, \mathbf{Z_{v_1x}}=\mathbf{z_{v_1x}}, U_1=u_1}\right] \qquad \text{(Step 3)} \\ &= w_{l_{XV_1}|O} w_{l_{V_2V_1}|O} \qquad \text{(Step 4).} \end{aligned}$$

The transition from step 2 to step 3 is valid because the first term in step 2 is independent of U_1 and the second term in step 2 is independent of U_2. Therefore, their expectations can be decomposed and calculated independently, which is shown in step 3.

Then, given a path contains multiple edges $\pi = \{XV_1, V_1V_2, \cdots V_{m-2}V_{m-1}, V_{m-1}Y\}$, because the impact score for each edge is independent, we can show that the calculation of path-specific

counterfactual importance score can be written as the format similar to the chain rule in calculus. The path-specific counterfactual importance score of the source vertex X on target vertex Y along the path π can be represented as:

$$\begin{aligned} w_{X \xrightarrow{\pi} Y|O} &= \mathbb{E}_{U_1|O}\left[\left.\frac{\partial V_1}{\partial X}\right|_O\right]\left(\prod_{i=1}^{m-2}\mathbb{E}_{U_{i+1}|O}\left[\left.\frac{\partial V_{i+1}}{\partial V_i}\right|_O\right]\right) \\ &\quad \mathbb{E}_{U_Y|O}\left[\left.\frac{\partial Y}{\partial V_Y}\right|_O\right] \\ &= w_{l_{XV_1}|O}\left(\prod_{i=1}^{m-2} w_{l_{V_iV_{i+1}}|O}\right) w_{l_{V_{m-1}Y}|O} \end{aligned}$$

□

7.10.3. Proof of Theorem 2.

Proof. We denote the set of all paths from X to Y as $\Pi_{X\to Y} = \{\pi = (XV_{\pi,1}, V_{\pi,1}V_{\pi,2}, \ldots, V_{\pi,m_\pi-1}Y) \mid X \in Pa(V_{\pi,1}), V_{\pi,m_\pi-1} \in Pa(Y), V_{\pi,i} \in Pa(V_{\pi,i+1}), \forall i = 1, \ldots, m_\pi - 2\}$, where m_π is the number of vertex in path π. We use m to denote the number of vertex in the largest path in the DAG (which means that if we start from Y going backward along its parents, we will reach the root at most m steps). Then from the definition of the total derivative, we have

$$\begin{aligned} w_{X\to Y|O} &= \lim_{\delta\to 0}\frac{\mathbb{E}[Y_{X=x+\delta}|O] - \mathbb{E}[Y_{X=x}|O]}{\delta} \\ &= \left.\frac{\partial Y}{\partial X}\right|_{X=x} \\ &= \sum_{V_{m-1}\in Pa(Y)\setminus\{X\}}\ \sum_{V_{m-2}\in Pa(V_{m-1})\setminus\{X\}} \cdots \sum_{V_1\in Pa(V_2)\setminus\{X\}} \\ &\quad \mathbb{E}_{\mathcal{U}|O}\left[\left.\frac{\partial Y}{\partial V_{m-1}}\right|_O\left(\prod_{i=1}^{m-2}\left.\frac{\partial V_{i+1}}{\partial V_i}\right|_O\right)\left.\frac{\partial V_1}{\partial X}\right|_O\right] \\ &= \sum_{\pi\in\Pi_{X\to Y}} \mathbb{E}_{U_Y|O}\left[\left.\frac{\partial Y}{\partial V_{\pi,m_\pi-1}}\right|_O\right] \\ &\quad \left(\prod_{i=1}^{m_\pi-2}\mathbb{E}_{U_{i+1}|O}\left[\left.\frac{\partial V_{\pi,i+1}}{\partial V_{\pi,i}}\right|_O\right]\right)\mathbb{E}_{U_1|O}\left[\left.\frac{\partial V_{\pi,1}}{\partial X}\right|_O\right] \\ &= \sum_{\pi\in\Pi_{X\to Y}} w_{X\xrightarrow{\pi}Y|O} \end{aligned}$$

The step of combining a sequence of summation operations to one summation over $\pi \in \Pi_{X \to Y}$ holds, because the change of X will not affect the value of V_{m-1} if there is not exist a path $(XV_{m-1}, V_{m-2}V_{m-1}, \ldots, V_1 Y)$. $\square$

8

Conclusion and Future Work

8.1. Conclusion

In this dissertation, we addressed significant challenges faced by reinforcement learning in real-world settings, including data efficiency, exploration-exploitation trade-off, and explainability, through a set of algorithms and solutions presented in three distinct parts.

First, we aimed to enhance data efficiency in multi-agent/task situations by developing online-learning-based joint optimization and kernel-based multi-task contextual bandits algorithms. By exploiting similarities among tasks or agents while optimizing their performance with limited data, our proposed algorithms demonstrated their potential for improving data utilization in multi-agent/task scenarios. As an application area, we considered cellular network configuration and showed that the algorithms significantly contributed to the optimization of such networks, outperforming conventional methods in settings where obtaining extensive training data is challenging.

Second, we proposed opportunistic reinforcement learning algorithms - AdaLinUCB, OppUCRL2, and OppPSRL - designed to adaptively balance exploration and exploitation when operating under varying exploration costs. These algorithms dynamically adjusted their optimism according to an external variation factor, resulting in superior performance in opportunistic learning scenarios when compared to existing contextual bandit and conventional RL algorithms. Our theoretical regret-bound analyses provided guarantees of their performance, highlighting their applicability and benefits in real-world environments where exploration costs fluctuate.

Finally, we tackled RL's lack of explainability by introducing a causal explanation mechanism that quantified the causal influence of states on actions and their temporal impact, providing powerful insights into RL policies. Our approach, which involved formulating MDP as SCM, was able to identify and quantify pivotal factors in RL policies that influence actions and their temporal relevance. To further leverage causality for improved explainability, we proposed a novel definition

of the causal counterfactual path-specific importance score for general SCMs, which enabled a more comprehensive and efficient understanding of the causal influence in decision chains.

Collectively, our research contributes to the advancement of RL by addressing and overcoming critical challenges in real-world settings, paving the way for the development of safer, more explainable, and efficient RL applications in the future. Each proposed algorithm and solution demonstrated effectiveness in various domains, particularly enhancing data efficiency in multi-agent/task scenarios, adaptively balancing exploration-exploitation trade-offs, and providing causal explanations for RL policies.

8.2. Future Work and Limitations

8.2.1. Multi-agent/Task Optimization with Application IN Cellular Network Configuration. The introduced approach for multi-agent or task optimization within cellular network configuration may conceivably be extrapolated to comparable network configuration challenges, including the configuration of handoff thresholds, antenna adjustments, and the allocation of transmission power. Moreover, conducting evaluative experiments within actual field conditions would be a considerable contribution towards research development in this domain. The proposed algorithm may prove beneficial in several application domains requiring the effective use of limited data and the careful navigation of the exploration-exploitation trade-off.

8.2.2. Opportunistic Reinforcement Learning. The present exploration of opportunistic learning may pave the way for subsequent research, particularly within the field of deep reinforcement learning, where maintaining an optimal balance between exploration and exploitation remains a persistent challenge. The theoretical regret bound is another salient element. In our existing work, given a square wave variation factor, both the OppUCRL2 and UCRL2 methods achieve equivalent bounds. Nevertheless, this is not universally applicable, with potential scenarios where the algorithm may not perform as effectively as simulation results might imply. In light of potential disparities between exploration strengths and the regret bound, it is recommended that subsequent research delve into sophisticated methods and other reinforcement learning environments, such as weakly communicating MDP and the efficiency of computational and statistical elements.

8.2.3. Causal Interpretability in Reinforcement Learning. The causal importance explanation mechanism proposed herein is a post-hoc method that utilizes data harvested from a previously learned policy. This renders it somewhat limited in addressing high-dimensional state spaces, particularly in visual reinforcement learning, where state features are manifested as images. To surmount this hurdle, the application of algorithms such as causal discovery in images and representation learning might prove instrumental. Moreover, if the trained SCM is imperfect, there may be potential issues. An imperfect SCM would yield skewed counterfactual reasoning results, subsequently leading to biases in final importance. Upcoming research should investigate methods of quantifying the uncertainty of the explanation, thereby enabling users to discern potential out-of-distribution samples where our explanation framework may falter.

In conclusion, the potential for enhancements and augmentations in subsequent research is vast, covering areas such as experimental evaluations, algorithmic improvements, and rectifying constraints in causal interpretability. The exploration of these research avenues promises to yield fresh insights, thereby empowering reinforcement learning approaches to better navigate real-world applications.

Bibliography

[1] Xueying Guo, George Trimponias, Xiaoxiao Wang, Zhitang Chen, Yanhui Geng, and Xin Liu. Cellular network configuration via online learning and joint optimization. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1295–1300. IEEE, 2017.

[2] Xueying Guo, George Trimponias, Xiaoxiao Wang, Zhitang Chen, Yanhui Geng, and Xin Liu. Learning-based joint configuration for cellular networks. *IEEE Internet of Things Journal*, 5(6):4283–4295, 2018.

[3] Xiaoxiao Wang, Xueying Guo, Jie Chuai, Zhitang Chen, and Xin Liu. Kernel-based multi-task contextual bandits in cellular network configuration. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1517–1526. IEEE, 2019.

[4] Xueying Guo, Xiaoxiao Wang, and Xin Liu. Adalinucb: Opportunistic learning for contextual bandits. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 2420–2427. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[5] Xiaoxiao Wang, Nader Bouacida, Xueying Guo, and Xin Liu. Opportunistic episodic reinforcement learning. *arXiv preprint arXiv:2210.13504*, 2022.

[6] Xiaoxiao Wang, Fanyu Meng, Xin Liu, Zhaodan Kong, and Xin Chen. Causal explanation for reinforcement learning: Quantifying state and temporal importance. *Applied Intelligence*, pages 1–19, 2023.

[7] Xiaoxiao Wang, Minda Zhao, Fanyu Meng, Xin Liu, Zhaodan Kong, and Xin Chen. Quantifying causal path-specific importance in structural causal model. *Computation*, 11(7):133, 2023.

[8] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, Cambridge, 2018.

[9] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[10] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[11] TP Imthias Ahamed, PS Nagendra Rao, and PS Sastry. A reinforcement learning approach to automatic generation control. *Electric power systems research*, 63(1):9–26, 2002.

[12] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[14] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

[15] Jie Chuai, Zhitang Chen, Guochen Liu, Xueying Guo, Xiaoxiao Wang, Xin Liu, Chongming Zhu, and Feiyi Shen. A collaborative learning based approach for parameter configuration of cellular networks. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1396–1404. IEEE, 2019.

[16] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.*, 3:397–422, 2002.

[17] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *AISTATS*, pages 208–214, 2011.

[18] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, pages 2312–2320, 2011.

[19] John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 817–824. Curran Associates, Inc., 2008.

[20] Apostolos N Burnetas and Michael N Katehakis. Optimal adaptive policies for markov decision processes. *Mathematics of Operations Research*, 22(1):222–255, 1997.

[21] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[22] Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *The Journal of Machine Learning Research*, 99:1563–1600, 2010.

[23] Osband Ian, Van Roy Benjamin, and Russo Daniel. (More) efficient reinforcement learning via posterior sampling. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, pages 3003–3011. Curran Associates Inc., 2013.

[24] Wei Ding and Di Yuan. A decomposition method for pilot power planning in umts systems. In *Digital Information and Communication Technology and it's Applications (DICTAP), Second International Conference,*, pages 42–47. IEEE, 2012.

[25] Kimmo Valkealahti, Albert Hõglund, Jyrki Parkkinen, and A Hamalainen. Wcdma common pilot power control for load and coverage balancing. In *Personal, Indoor and Mobile Radio Communications. The 13th IEEE International Symposium,*, volume 3, pages 1412–1416. IEEE, 2002.

[26] Adrian Agustin, Sandra Lagen, and Josep Vidal. Energy efficient cell load-aware coverage optimization for small-cell networks. In *Communications (ICC), IEEE International Conference,*, pages 2036–2041. IEEE, 2015.

[27] Imran Ashraf, Holger Claussen, and Lester TW Ho. Distributed radio coverage optimization in enterprise femtocell networks. In *IEEE International Conference Communications(ICC)*, pages 1–6, 2010.

[28] Rouzbeh Razavi and Holger Claussen. Self-configuring switched multi-element antenna system for interference mitigation in femtocell networks. In *Personal Indoor and Mobile Radio Communications (PIMRC), IEEE 22nd International Symposium on*, pages 237–242. IEEE, 2011.

[29] Cong Shen, Ruida Zhou, Cem Tekin, and Mihaela van der Schaar. Generalized global bandit and its application in cellular coverage optimization. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):218–232, 2018.

[30] X. Chen, J. Wu, Y. Cai, H. Zhang, and T. Chen. Energy-Efficiency Oriented Traffic Offloading in Wireless Networks: A Brief Survey and a Learning Approach for Heterogeneous Cellular Networks. *IEEE Journal on Selected Areas in Communications*, 33(4):627–640, April 2015.

[31] Liang Xiao, Yan Li, Jinliang Liu, and Yifeng Zhao. Power control with reinforcement learning in cooperative cognitive radio networks against jamming. *The Journal of Supercomputing*, 71(9):3237–3257, Sep 2015.

[32] L. Xiao, Y. Li, C. Dai, H. Dai, and H. V. Poor. Reinforcement Learning-Based NOMA Power Allocation in the Presence of Smart Jamming. *IEEE Transactions on Vehicular Technology*, 67(4):3377–3389, April 2018.

[33] Ching-Yu Liao, Fei Yu, Victor CM Leung, and Chung-Ju Chang. A novel dynamic cell configuration scheme in next-generation situation-aware cdma networks. *IEEE Journal on Selected Areas in Communications,*, 24(1):16–25, 2006.

[34] Carlos Guestrin, Michail G. Lagoudakis, and Ronald Parr. Coordinated reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 227–234, 2002.

[35] Jelle R. Kok and Nikos Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *J. Mach. Learn. Res.*, 7:1789–1828, December 2006.

[36] Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Machine Learning*, 5(1):1–122, 2012.

[37] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[38] Olivier Chapelle and Lihong Li. An empirical evaluation of Thompson Sampling. In *Advances in Neural Information Processing Systems*, pages 2249–2257, 2011.

[39] Lihong Li, Wei Chu, John Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *ACM International Conference on World Wide Web*, pages 661–670, 2010.

[40] Wenzhuo Ouyang, A Eryilmaz, and N. B. Shroff. Asymptotically optimal downlink scheduling over Markovian fading channels. In *Proceedings of IEEE INFOCOM*, pages 1224–1232, March 2012.

[41] Xueying Guo, Rahul Singh, P. R. Kumar, and Zhisheng Niu. Optimal energy-efficient regular delivery of packets in cyber-physical systems. In *IEEE ICC*, 2015.

[42] Rahul Singh, Xueying Guo, and P. R. Kumar. Index policies for optimal mean-variance trade-off of inter-delivery times in real-time sensor networks. In *Proceedings of IEEE INFOCOM*, 2015.

[43] Yuxuan Sun, Xueying Guo, Sheng Zhou, Zhiyuan Jiang, Xin Liu, and Zhisheng Niu. Learning-based task offloading for vehicular cloud computing systems. In *IEEE ICC*, 2018.

[44] Yuxuan Sun, Jinhui Song, Sheng Zhou, Xueying Guo, and Zhisheng Niu. Task replication for vehicular edge computing: a combinatorial multi-armed bandit based approach. In *IEEE GLOBECOM*, 2018.

[45] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[46] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2249–2257. Curran Associates, Inc., 2011.

[47] Rajeev Agrawal. Sample mean based index policies with o(log n) regret for the multi-armed bandit problem. *Advances in Applied Probability*, 27(4):1054–1078, 1995.

[48] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *the 19th International Conference on World Wide Web (WWW)*, 2010.

[49] Michal Valko, Nathaniel Korda, Rémi Munos, Ilias Flaounas, and Nelo Cristianini. Finite-time analysis of kernelised contextual bandits. *arXiv preprint arXiv:1309.6869*, 2013.

[50] Qingyun Wu, Huazheng Wang, Quanquan Gu, and Hongning Wang. Contextual bandits in a collaborative environment. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 529–538. ACM, 2016.

[51] Huazheng Wang, Qingyun Wu, and Hongning Wang. Learning hidden features for contextual bandits. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1633–1642. ACM, 2016.

[52] Huazheng Wang, Qingyun Wu, and Hongning Wang. Factorization bandits for interactive recommendation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, 2017.

[53] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[54] Sofía S Villar, Jack Bowden, and James Wason. Multi-armed bandit models for the optimal design of clinical trials: benefits and challenges. *Statistical science: a review journal of the Institute of Mathematical Statistics*, 30(2):199, 2015.

[55] Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[56] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):9, 2016.

[57] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM, 2004.

[58] Chang Wang and Sridhar Mahadevan. Heterogeneous domain adaptation using manifold alignment. In *International Joint Conference on Artificial Intelligence*, volume 22, page 1541, 2011.

[59] Lixin Duan, Dong Xu, and Ivor Tsang. Learning with augmented features for heterogeneous domain adaptation. *arXiv preprint arXiv:1206.4660*, 2012.

[60] Rita Chattopadhyay, Qian Sun, Wei Fan, Ian Davidson, Sethuraman Panchanathan, and Jieping Ye. Multisource domain adaptation and its application to early detection of fatigue. *ACM Transactions on Knowledge Discovery from Data*, 6(4):18, 2012.

[61] Sinno Jialin Pan, James T Kwok, and Qiang Yang. Transfer learning via dimensionality reduction. In *AAAI*, volume 8, pages 677–682, 2008.

[62] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011.

[63] Edwin V Bonilla, Kian M Chai, and Christopher Williams. Multi-task gaussian process prediction. In *Advances in neural information processing systems*, pages 153–160, 2008.

[64] Edwin V Bonilla, Felix V Agakov, and Christopher KI Williams. Kernel multi-task learning using task-specific features. In *Artificial Intelligence and Statistics*, pages 43–50, 2007.

[65] Aniket Anand Deshmukh, Urun Dogan, and Clay Scott. Multi-task learning for contextual bandits. In *Advances in Neural Information Processing Systems*, pages 4851–4859, 2017.

[66] Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 263–272. PMLR, 2017.

[67] Peter L Bartlett and Ambuj Tewari. REGAL: A regularization based algorithm for reinforcement learning in weakly communicating MDPs. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 35–42. AUAI Press, 2009.

[68] Christoph Dann and Emma Brunskill. Sample complexity of episodic fixed-horizon reinforcement learning. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, pages 2818–2826. MIT Press, 2015.

[69] Alexander L. Strehl, Lihong Li, and Michael L. Littman. Reinforcement learning in finite MDPs: PAC analysis. *The Journal of Machine Learning Research*, 10:2413–2444, 2009.

[70] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, pages 2249–2257. Curran Associates Inc., 2011.

[71] Malcolm Strens. A Bayesian framework for reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning*, pages 943–950, 2000.

[72] Ian Osband and Benjamin Van Roy. Why is posterior sampling better than optimism for reinforcement learning? In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 2701–2710. PMLR, 2017.

[73] Huasen Wu, Xueying Guo, and Xin Liu. Adaptive exploration-exploitation tradeoff for opportunistic bandits. In *ICML*, 2018.

[74] Djallel Bouneffouf, Amel Bouzeghoub, and Alda Lopes Gançarski. A contextual-bandit algorithm for mobile context-aware recommender system. In *International conference on Neural Information Processing (ICONIP)*, pages 324–331, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[75] Erika Puiutta and Eric Veith. Explainable reinforcement learning: A survey. In *International cross-domain conference for machine learning and knowledge extraction*, pages 77–95. Springer, 2020.

[76] Alexandre Heuillet, Fabien Couthouis, and Natalia Díaz-Rodríguez. Explainability in deep reinforcement learning. *Knowledge-Based Systems*, 214:106685, 2021.

[77] Lindsay Wells and Tomasz Bednarz. Explainable ai and reinforcement learning—a systematic review of current approaches and trends. *Frontiers in artificial intelligence*, 4:550030, 2021.

[78] Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding atari agents. In *International Conference on Machine Learning*, pages 1792–1801. PMLR, 2018.

[79] Nikaash Puri, Sukriti Verma, Piyush Gupta, Dhruv Kayastha, Shripad Deshmukh, Balaji Krishnamurthy, and Sameer Singh. Explain your move: Understanding agent actions using specific and relevant feature attribution. *arXiv preprint arXiv:1912.12191*, 2019.

[80] Zoe Juozapaitis, Anurag Koul, Alan Fern, Martin Erwig, and Finale Doshi-Velez. Explainable reinforcement learning via reward decomposition. In *IJCAI/ECAI Workshop on Explainable Artificial Intelligence*, 2019.

[81] Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning*, pages 5045–5054. PMLR, 2018.

[82] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[83] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017.

[84] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[85] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*, 2017.

[86] Patrick Schwab and Walter Karlen. Cxplain: Causal explanations for model interpretation under uncertainty. *arXiv preprint arXiv:1910.12336*, 2019.

[87] Aditya Chattopadhyay, Piyushi Manupriya, Anirban Sarkar, and Vineeth N Balasubramanian. Neural network attributions: A causal perspective. In *International Conference on Machine Learning*, pages 981–990. PMLR, 2019.

[88] Anupam Datta, Shayak Sen, and Yair Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE symposium on security and privacy (SP)*, pages 598–617. IEEE, 2016.

[89] Prashan Madumal, Tim Miller, Liz Sonenberg, and Frank Vetere. Explainable reinforcement learning through a causal lens. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2493–2500, 2020.

[90] Matthew L Olson, Roli Khanna, Lawrence Neal, Fuxin Li, and Weng-Keen Wong. Counterfactual state explanations for reinforcement learning agents via generative deep learning. *Artificial Intelligence*, 295:103455, 2021.

[91] Daniel Malinsky, Ilya Shpitser, and Thomas Richardson. A potential outcomes calculus for identifying conditional path-specific effects. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3080–3088. PMLR, 2019.

[92] Junzhe Zhang and Elias Bareinboim. Non-parametric path analysis in structural causal models. In *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence*, 2018.

[93] Heyang Gong and Ke Zhu. Path-specific effects based on information accounts of causality. *arXiv preprint arXiv:2106.03178*, 2021.

[94] Dominik Rothenhäusler and Bin Yu. Incremental causal effects. *arXiv preprint arXiv:1907.13258*, 2019.

[95] Xiang Zhou and Yu Xie. Marginal treatment effects from a propensity score perspective. *Journal of Political Economy*, 127(6):3070–3084, 2019.

[96] Xiang Zhou and Aleksei Opacic. Marginal interventional effects. *arXiv preprint arXiv:2206.10717*, 2022.

[97] Dominik Janzing, Patrick Blöbaum, Lenon Minorics, and Philipp Faller. Quantifying causal contributions via structure preserving interventions. *arXiv preprint arXiv:2007.00714*, 2020.

[98] Dominik Janzing, David Balduzzi, Moritz Grosse-Wentrup, and Bernhard Schölkopf. Quantifying causal influences. 2013.

[99] Jiaxuan Wang, Jenna Wiens, and Scott Lundberg. Shapley flow: A graph-based approach to interpreting model predictions. In *International Conference on Artificial Intelligence and Statistics*, pages 721–729. PMLR, 2021.

[100] Iana Siomina and Di Yuan. Soft handover overhead control in pilot power management in wcdma networks. In *Vehicular Technology Conference. VTC 2005-Spring*, volume 3, pages 1875–1879. IEEE, 2005.

[101] X. Cheng, L. Fang, X. Hong, and L. Yang. Exploiting Mobile Big Data: Sources, Features, and Applications. *IEEE Network*, 31(1):72–79, January 2017.

[102] X. Cheng, L. Fang, L. Yang, and S. Cui. Mobile Big Data: The Fuel for Data-Driven Wireless. *IEEE Internet of Things Journal*, 4(5):1489–1516, Oct 2017.

[103] Suzhi Bi, Rui Zhang, Zhi Ding, and Shuguang Cui. *Big data aware wireless communication: challenges and opportunities*, pages 180–216. Cambridge University Press, 2016.

[104] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer New York Inc., New York, NY, USA, 2017 printed.

[105] Pierre Brémaud. *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*, volume 31. Springer Science & Business Media, 2013.

[106] S. C. Borst, M. G. Markakis, and I. Saniee. Nonconcave utility maximization in locally coupled systems, with applications to wireless and wireline networks. *IEEE/ACM Transactions on Networking*, 22(2):674–687, April 2014.

[107] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[108] F. P. Kelly. *Reversibility and Stochastic Networks*. Cambridge University Press, New York, NY, USA, 2011.

[109] Cisco Visual Networking Index. Cisco visual networking index: global mobile data traffic forecast update, 2014–2019. *Tech. Rep*, 2015.

[110] X. Guo, Z. Niu, S. Zhou, and P. R. Kumar. Delay-constrained energy-optimal base station sleeping control. *IEEE Journal on Selected Areas in Communications*, 34(5):1073–1085, May 2016.

[111] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[112] Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

[113] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[114] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[115] Fuzhen Zhang. *The Schur complement and its applications*, volume 4. Springer Science & Business Media, 2006.

[116] Carl Edward Rasmussen. Gaussian processes for machine learning. MIT Press, 2006.

[117] Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A Hilbert space embedding for distributions. In *ALT*, 2007.

[118] Andreas Krause and Cheng S Ong. Contextual gaussian process bandit optimization. In *Advances in Neural Information Processing Systems*, pages 2447–2455, 2011.

[119] Joris Walraevens, Bart Steyaert, and Herwig Bruneel. Performance analysis of a single-server atm queue with a priority scheduling. *Computers & Operations Research*, 30(12):1807 – 1829, 2003.

[120] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 297–306. ACM, 2011.

[121] Sarah Filippi, Olivier Cappe, Aurélien Garivier, and Csaba Szepesvári. Parametric bandits: The generalized linear case. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 586–594. Curran Associates, Inc., 2010.

[122] Rossman. *Rossmann Store sales data*, 2015. `https://www.kaggle.com/c/rossmann-store-sales/data`.

[123] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced Lectures on Machine Learning*, pages 63–71. Springer, 2004.

[124] Wei Chu, Seung-Taek Park, Todd Beaupre, Nitin Motgi, Amit Phadke, Seinjuti Chakraborty, and Joe Zachariah. A case study of behavior-driven conjoint analysis on yahoo!: Front page today module. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 1097–1104, 2009.

[125] Christoph Dann, Lihong Li, Wei Wei, and Emma Brunskill. Policy certificates: Towards accountable reinforcement learning. In *International Conference on Machine Learning*, pages 1507–1516, 2019.

[126] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[127] Mohammad Ghavamzadeh, Alessandro Lazaric, and Matteo Pirotta. Exploration in reinforcement learning, 2020.

[128] Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

[129] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. 2014.

[130] Rahul Iyer, Yuezhang Li, Huao Li, Michael Lewis, Ramitha Sundar, and Katia Sycara. Transparency and explanation in deep reinforcement learning neural networks. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 144–150, 2018.

[131] Alex Mott, Daniel Zoran, Mike Chrzanowski, Daan Wierstra, and Danilo J Rezende. Towards interpretable reinforcement learning using attention augmented agents. *arXiv preprint arXiv:1906.02500*, 2019.

[132] J. Pearl. *Causality*. Causality: Models, Reasoning, and Inference. Cambridge University Press, Cambridge, 2009.

[133] Ruth MJ Byrne. Counterfactuals in explainable artificial intelligence (xai): Evidence from human reasoning. In *IJCAI*, pages 6276–6282, 2019.

[134] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

[135] Denis Hilton. Causal explanation: From social perception to knowledge-based causal attribution. 2007.

[136] Peter Spirtes, Clark N Glymour, and Richard Scheines. *Causation, prediction, and search*. MIT Press, 2000.

[137] Shohei Shimizu, Patrik O Hoyer, Aapo Hyvärinen, and Antti Kerminen. A linear non-Gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7(Oct):2003–2030, 2006.

[138] Patrik Hoyer, Dominik Janzing, Joris M Mooij, Jonas Peters, and Bernhard Schölkopf. Nonlinear causal discovery with additive noise models. *Advances in neural information processing systems*, 21:689–696, 2008.

[139] Jonas Peters, Joris M Mooij, Dominik Janzing, and Bernhard Schölkopf. Causal discovery with continuous additive noise models. 2014.

[140] JR Williams, CA Jones, JR Kiniry, and Deborah A Spanel. The epic crop growth model. *Transactions of the ASAE*, 32(2):497–0511, 1989.

[141] Arthur Earl Bryson. *Applied optimal control: optimization, estimation and control*. CRC Press, Boca Raton, 1975.

[142] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[143] Madelyn Glymour, Judea Pearl, and Nicholas P Jewell. *Causal inference in statistics: A primer*. John Wiley & Sons, Hoboken, 2016.

[144] Ayush Jaiswal, Wael AbdAlmageed, Yue Wu, and Premkumar Natarajan. Bidirectional conditional generative adversarial networks. In *Asian Conference on Computer Vision*, pages 216–232. Springer, 2018.

[145] Diviyan Kalainathan and Olivier Goudet. Causal discovery toolbox: Uncover causal relationships in python. *arXiv preprint arXiv:1903.02278*, 2019.

[146] Keli Zhang, Shengyu Zhu, Marcus Kalander, Ignavier Ng, Junjian Ye, Zhitang Chen, and Lujia Pan. gcastle: A python toolbox for causal discovery. *arXiv preprint arXiv:2111.15155*, 2021.

[147] David Lopez-Paz, Robert Nishihara, Soumith Chintala, Bernhard Scholkopf, and Léon Bottou. Discovering causal signals in images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6979–6987, 2017.

[148] Mengyue Yang, Furui Liu, Zhitang Chen, Xinwei Shen, Jianye Hao, and Jun Wang. Causalvae: Disentangled representation learning via neural structural causal models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9593–9602, 2021.

[149] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342*, 2021.

[150] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[151] Reuben M Baron and David A Kenny. The moderator–mediator variable distinction in social psychological research: Conceptual, strategic, and statistical considerations. *Journal of personality and social psychology*, 51(6):1173, 1986.

[152] Karen Sachs, Omar Perez, Dana Pe'er, Douglas A Lauffenburger, and Garry P Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529, 2005.

[153] James H Stock and Mark W Watson. Identification and estimation of dynamic causal effects in macroeconomics using external instruments. *The Economic Journal*, 128(610):917–948, 2018.

[154] Silvia Chiappa. Path-specific counterfactual fairness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7801–7808, 2019.

[155] Yongkai Wu, Lu Zhang, Xintao Wu, and Hanghang Tong. Pc-fairness: A unified framework for measuring causality-based fairness. *Advances in Neural Information Processing Systems*, 32, 2019.

[156] Yoichi Chikahara, Shinsaku Sakaue, Akinori Fujino, and Hisashi Kashima. Learning individually fair classifier with path-specific causal-effect constraint. In *International Conference on Artificial Intelligence and Statistics*, pages 145–153. PMLR, 2021.

[157] Hongkai Li, Zhi Geng, Xiaoru Sun, Yuanyuan Yu, and Fuzhong Xue. A novel path-specific effect statistic for identifying the differential specific paths in systems epidemiology. *BMC genetics*, 21(1):1–12, 2020.

[158] Chen Avin, Ilya Shpitser, and Judea Pearl. Identifiability of path-specific effects. 2005.

[159] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[160] Dieter Jungnickel and D Jungnickel. *Graphs, networks and algorithms*, volume 3. Springer, 2005.

[161] Steve HC Yen, David Hung-Chang Du, and Subbarao Ghanta. Efficient algorithms for extracting the k most critical paths in timing analysis. In *26th ACM/IEEE Design Automation Conference*, pages 649–654. IEEE, 1989.

[162] Emelia J Benjamin, Daniel Levy, Sonya M Vaziri, Ralph B D'Agostino, Albert J Belanger, and Philip A Wolf. Independent risk factors for atrial fibrillation in a population-based cohort: the framingham heart study. *Jama*, 271(11):840–844, 1994.

[163] Sheng-Hsuan Lin and Tyler VanderWeele. Interventional approach for path-specific effects. *Journal of Causal Inference*, 5(1), 2017.

[164] R.G. Bartle. *The Elements of Integration and Lebesgue Measure*. Wiley Classics Library. Wiley, 1995.