HARVARD UNIVERSITY

Graduate School of Arts and Sciences

DISSERTATION ACCEPTANCE CERTIFICATE

The undersigned, appointed by the

Harvard John A. Paulson School of Engineering and Applied Sciences
have examined a dissertation entitled:

"Integrating Machine Learning and Optimization with Applications in Public Health and Sustainability"

presented by: Kai Wang

Signature _____
Typed name: Professor M. Tambe

Signature _____
Typed name: Professor D. Parkes

Signature _____
Typed name: Professor Y. Chen

Signature _____
Typed name: Professor A. Procaccia

May 4, 2023

Integrating Machine Learning and Optimization with Applications in Public Health and Sustainability

a dissertation presented
by
Kai Wang
to
The School of Engineering and Applied Sciences

in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the subject of
Computer Science

Harvard University
Cambridge, Massachusetts
May 2023

Thesis advisor: Professor Milind Tambe Kai Wang

Integrating Machine Learning and Optimization with Applications in Public Health and Sustainability

Abstract

The field of artificial intelligence (AI) has garnered increasing attention in the realms of public health and conservation due to its potential to characterize complex dynamics and facilitate difficult decision-making. My research focuses on developing AI solutions, utilizing machine learning and optimization techniques, to provide actionable decisions for deployment and create positive social impact. This endeavor necessitates the integration of new algorithmic and learning paradigms, combining machine learning techniques to extract knowledge from data and optimization techniques to leverage domain knowledge and scale up to larger problem sizes. In this thesis, I present methodological and theoretical contributions in the integration of optimization into machine learning problems, including supervised learning, online learning, and multi-agent systems, with the aim of improving learning performance and scalability by harnessing the knowledge encoded in optimization tasks. Notably, this thesis introduces the first decision-focused learning to integrate sequential problems into the learning pipeline to provide feedback from decision-making processes and significantly reduce computation costs, thus enabling applications in large-scale public health problems. The proposed algorithm has been successfully applied in a field study and deployment in a maternal and child health program, marking the first successful implementation of decision-focused learning in the real world. Currently, the proposed algorithm is used by over 100,000 beneficiaries in India to enhance engagement with health information and translate algorithmic contributions into tangible social impact.

Contents

List of Tables

Listing of figures

Acknowledgments

My Ph.D. journey was beyond my expectations, as it taught me and exposed me to more than I could have imagined. I am blessed to have had the opportunity to develop my skills and expertise, explore my research interests, establish connections and friendships, and use what I learned to contribute to others.

PhD Advisor: I feel extremely lucky to have had the opportunity to work with my advisor, Milind Tambe. You are the best mentor, an inspirational leader, and a lifelong friend to me. Your wisdom guides me in the vast academic ocean, and your encouragement cheers me in frustrated moments. I have learned so much from your aspiration, life experiences, research philosophy, and beyond. I will keep all your support, feedback, and every single word in my mind. All of your advice becomes the nutrients to grow my life, to flourish, and to shine. Thank you, Milind.

Thesis Committee: I would also like to express my gratitude to the members of my thesis committee: David Parkes, Yiling Chen, and Ariel Procaccia. I have learned a lot from the conversation and interaction with you. Your feedback and advice have significantly influenced and shaped my research directions and the long-term vision. Thank you for bringing me to the field and guiding me to master it.

Research and Life Mentors: Many other mentors, including Finale Doshi-Velez, David Parkes, Aparna Taneja, Georgios Theocharous, Sridhar Mahadevan, Boaz Barak, Salil Vadhan, Haifeng Xu, Bryan Wilder, and Andrew Perrault, have also provided me with guidance and assistance. I would like to extend a special thanks to a few of them: Haifeng Xu, for showing me how to conduct research when I began my Ph.D. Without your guidance, I would not have made it this far. Bryan Wilder, for being a role model and providing me with with fresh research insights all the time. I thoroughly enjoyed the moment of discussing research and brainstorming with you. Andrew Perrault, for mentoring me and providing feedback to help me become a more mature researcher. I learned and benefited a lot from working with you.

Coauthors and collaborators: I want to thank all my co-authors, labmates, and mentors that I have collaborated with during the past few years. This thesis reflects a significant amount

of collaborative effort with every one of you. I have been fortunate to work with Brandon Amos, Bo An, Elizabeth Bondi, Jeffrey Brantingham, Haipeng Chen, Sarah Cooney, Edward Cranford, Panayiotis Danassis, Bistra Dilkina, Finale Doshi-Velez, Fei Fang, Jessica Finocchiaro, Cleotilde Gonzalez, Qingyu Guo, Umang Gupta, Aparna Hegde, Gauri Jain, Sonja Johnson-Yu, Nitin Kamra, David Kempes, Mykel Kochenderfer, Corine Laan, Christian Lebiere, Yan Liu, Neha Madhiwalla, Sridhar Mahadevan, Aditya Mate, Sara Marie Mc Carthy, Ayan Mukhopadhyay, Thanh Nguyen, Han Ching Ou, Andrew Perrault, Michael Reiter, Sanket Shah, Christoph Siebenbrunner, Arunesh Sinha, Zhao Song, Sze-chuan Suen, Milind Tambe, Aparna Taneja, Georgios Theocharous, Phebe Vayanos, Shresth Verma, Yevgeniy Vorobeychik, Bryan Wilder, Hailey Winetrobe, Haifeng Xu, Lily Xu. I appreciate the opportunity to work with and learn from each of you. In particular, I would like to thank Han-Ching and Aditya for always being there to listen and support me. I also want to especially thank Lily for her invaluable support, encouragement, diligence, and optimism throughout many collaborations with me. I am grateful to have worked with you and shared so many memories.

Labmates and Friends at Harvard and USC: I am also grateful to my academic friends and labmates at Harvard and USC, including Haifeng Xu, Amulya Yadav, Sara Marie McCarthy, Aaron Schlenker, Shahrzad Gholami, Bryan Wilder, Aida Rahmattalabi, Elizabeth Bondi-Kelly, Han-Ching Ou, Lily Xu, Aditya Mate*, Jackson Killian, Sanket Shah, Paula Rodriguez Diaz, Sonja Johnson-Yu, Lucia Gordon, Gauri Jain, Hongjin Lin, Rachel Guo, Shahin Jabbari, Andrew Perrault, Christoph Siebenbrunner, Haipeng Chen, Arpita Biswas, Panayiotis Danassis, Esther Rolf, Jessica Finocchiaro, Juspreet Singh Sandhu, Chi-Ning Chou, James Larisch, Aaron Ferber, Caleb Robinson, Amrita Gupta, Te-Lin Wu, Chou-Chun Wu, Jeremy Hsu, Chen-Yu Wei, Tsung-Yu Lu, Cho-Ying Wu, Wei-Chung Wang, Jeffrey Jow. I feel extremely fortunate to have had the chance to meet so many friends, labmates, and collaborators in Boston and Los Angeles. I am grateful for having you participated in my journey, and I am glad to be part of your journey as well.

Harvard Taiwanese Student Association (HTSA): I would also like to express my gratitude to the friends I met through the HTSA. I had the opportunity to get to know a group of close friends and create memories together. Lois Tang, Ginnie Hu, Vicky Huang, and Ellen Chao, you are the best team to work with, always covering my back and making the cold Boston winters full of laughs and joy. I am also thankful to Yu-Shun Hsiao, Min Tang, Phoebe Tsai, Hsiang Hsu, Chih-Fu Wei, Chia-Rui Chang, Jonathan Chang, Ling Qin, and the entire HTSA for making my life colorful and splendid. Joining the HTSA in 2022 was one of the best decisions I have made. I am also thankful to every one of you for making my life colorful and splendid. Boston becomes much warmer with all of you together.

College Friends and West Coast Life: I am also thankful for my friends since college and high school, who have shared the most joyful times with me in college and the US. Ping-Min Lin, Shih-Yen Tao, Yu-Heng Hsieh, Te-Li Wang, and Yi-Chen Lin, you have been the best friends and

the best team together in college and beyond. We are too close that we do not even need texts to communicate. I sincerely appreciate having you in my life and that we are still chatting frequently to remind me that you are always with me. I want to give a deep thank to Yi Lee for the most cherished and precious memory during the most difficult time. You have taught me many important life lessons and significantly shaped my personality. You are an amazingly thoughtful person and the brightest light to all your friends and me. I wish you all the best on your journey and hope to catch up with you in the future.

Volleyball Friends: I am thankful to meet groups of amazing friends at National Taiwan University, West Coast, and Boston through volleyball. Volleyball constitutes an irreplaceable part of my college and Ph.D. to support my physical and mental health. I want to especially thank Nico Wei, Ying-Hsin Chen, Cyril Wu, Ivy Chen, Ting-Ya Chang, Congcong Zhu, Yi Ke, and Yu Li for constantly sharing your kindness and having fun events together. You made my Boston life not only energetic but also feel like home. Meeting you in Boston is my greatest luck.

Family: To my dearest significant other, Yu-Tien Hsu, I want to express my gratitude for giving me the courage to face new challenges, warming my heart with your unlimited energy and unconditional love, and making every part of my life in Boston better and more memorable. I am grateful to have met you, to have gotten to know you, and to have fallen in love with you. You are such a wonderful person, and I would love to learn more from you. I can't wait to explore the next chapter with you. Finally, I want to thank my parents for providing me with unlimited freedom and support in navigating the future. "Be kind to people and do whatever you want" – this is the only but the most important message you want me to keep in mind, and I am grateful for your years of love, trust, and support. You are always there for me, and home with you is always the best and safest shelter for me to recover and restart my adventure. None of my journey would have been possible without everything you have given me.

1

Introduction

We are confronted with numerous global challenges, particularly in the areas of public health and environmental sustainability, which disproportionately affect the most vulnerable populations. For instance, maternal health and maternal mortality during pregnancy [239,62], recognized as one of the United Nations' sustainability goals in health, poses a significant threat to under-resourced communities in the United States [157,139,67] and the Global South [266,239,7]. Similarly, illegal poaching and illegal wildlife trade have had severe consequences on wildlife population and biodiver-

sity[364,343,28,16,314]. As the poaching and smuggling crisis continues to devastate populations of endangered animals, the implementation of protections for these species becomes of utmost importance[83,220,74,63].

In order to address the magnitude of large-scale societal challenges, scientists have invested a significant amount of effort in finding actionable solutions to act on the problems of interest. For instance, in the field of maternal health, interventions such as iron and folic acid supplements have been shown to reduce the risk of premature birth[277,8] and the likelihood of having a child with spina bifida[138,104,22]. In the realm of conservation, patrolling efforts in national parks serve as a deterrent against poaching and smuggling activities[99,221,336,253]. All of these works showcase the power of intervention and public policy on public health and conservation.

The aforementioned success in customized intervention and public policy motivates scientists to study how to scale up the impact using AI. We have seen how AI has been used in various industrial and societal applications to suggest actionable decisions and maximize desired performance. For example, AI has been used in digital marketing to decide how to allocate limited advertisement resources to maximize revenue under constraints and uncertainty[188,86,61]. AI has also been used in urban planning and smart cities to optimize traffic design and urban development decisions under resource constraints and regulations[352,344,76]. These AI applications leverage machine learning to quantify uncertainty and characterize knowledge based on available data, and formulate optimization and decision-making processes based on domain experts' knowledge to suggest actionable decisions.

However, in public health and environmental sustainability, collecting data is expensive and time-consuming, leading to the issue of limited data in machine learning that is difficult to learn a meaningful model and extract useful knowledge. Furthermore, decision making in public health and environmental sustainability can be entangled by constraints and multiple self-interested agents involved, posing an additional question on how to properly formulate optimization problems to

find optimal decisions using knowledge learned from machine learning. The combined challenges of limited data, complex decision-making processes, and the involvement of multiple agents hinder the integration of AI components for designing data-driven decision-making solutions in public health and environmental sustainability. Therefore, the main research question of this thesis is:

> How to design AI solutions using machine learning and optimization in public health and environmental sustainability?

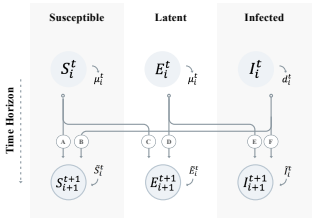
In my thesis, I delve into the study of machine learning algorithms to quantify uncertainty and knowledge based on limited data using machine learning and design scalable algorithms to translate knowledge obtained from data into actionable decisions using optimization. I aim to design effective machine learning and optimization algorithms to address the challenges posed by uncertainty and resource constraints in tackling large-scale societal issues. As shown in Figure 1.1, The public health and environmental sustainability problems studied in this thesis include:

- **Maternal health:** I study improving access and engagement to maternal health information through learning mothers' engagement behavior and allocating limited health workers to provide further assistance. My algorithm on integrating machine learning and optimization led to a real-world deployment to the mobile maternal health program in India used by more than 100,000 mothers with a 30% more improvement on the engagement to health information.

- **Wildlife conservation:** I study predicting poaching risk of different locations in national parks based on terrain features and historical poaching data to determine how to assign limited park rangers and patrol resources with patrol route recommendations. I also study collaboration between park rangers and patrol posts by designing mechanism to incentivize collaboration without communication.

(a) Maternal and child health using AI to predict engagement to health information and schedule health workers to provide assistance. Photo taken by Kai Wang during the field trip in Mumbai, India.

(b) Wildlife conservation using AI to predict poaching risk in national parks and optimize patrol routes to maximize patrol performance. Photo taken by Lily Xu during her field trip in Cambodia.



(c) Epidemiology using AI to forecast adherence and disease transmission parameters in agent-based modeling and compartment models, and solving optimization to recommend intervention strategies.

Figure 1.1: I study maternal health, wildlife conservation, and epidemiology by designing machine learning and optimization solutions in the data-to-deployment pipeline.

- **Epidemiology and tuberculosis adherence:** I study the problem of learning adherence behavior to tuberculosis medication based on historical data and scheduling health workers to call or physically visit patients to encourage adherence.

1.1 Problem Statement

To holistically answer the research question in public health and environmental sustainability, as illustrated in Figure 1.2, I study the data-to-deployment pipeline to develop data-driven decision-making AI solutions. The pipeline involves several key steps. Firstly, data relevant to the targeted problem is utilized to train machine learning algorithms, enabling the construction of an accurate model to characterize the problem of interest. Secondly, leveraging the constructed model and considering limited intervention resources, the resource allocation problem is formulated as an optimization challenge, aimed at optimizing the intervention performance and decision quality. Finally, in collaboration with domain experts and organizations, the suggested intervention decision is thoroughly examined and, upon validation, deployed in the field with multiple parties involved to create

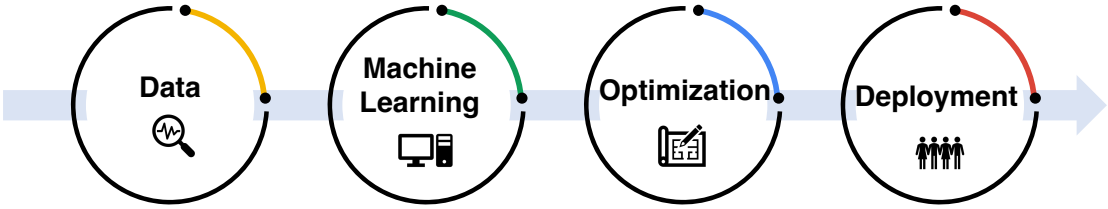


Figure 1.2: The data-to-deployment pipeline that is commonly used in AI, data science, and societal challenges.

social impact.

This data-to-deployment pipeline is widely applicable across various industrial and societal domains. For example, it can be employed in maternal health programs to analyze historical data on mothers' engagement behavior and determine appropriate interventions [211,10,174]. It can also be utilized in wildlife conservation programs to predict and allocate patrol resources for areas at risk of illegal poaching activities [349,253]. In epidemiology modeling, the pipeline can be used to fit disease models with parameters and optimize intervention design [153,118]. Additionally, it can aid in solving routing problems by fitting traffic predictive models and finding optimal routes [317,229], as well as in designing advertisement and recommendation systems that learn user preferences and provide suitable recommendations [275,261].

1.2 Challenges

Designing algorithms and individual components for the data-to-deployment pipeline poses a number of technical challenges. I summarize a list of topics studied in this thesis below:

- **Learning in the presence of optimization:** In the realm of public health and environmental conservation, machine learning and optimization play vital roles in transforming data into actionable decisions for implementation. However, traditional supervised learning techniques primarily rely on comparing predictions with ground truth labels to define accuracy metrics or loss functions. In contrast, optimization and decision-making processes

prioritize the quality of proposed decisions than accuracy metrics. This discrepancy in objectives between machine learning and optimization introduces a potential gap in the data-to-deployment pipeline. This objective mismatch issue is pervasive in various AI and data science challenges that involve the joint utilization of machine learning and optimization to convert data into actionable decisions.

- **Data exploration and exploitation:** In addition to machine learning and optimization, there is often an opportunity to collect new data during the deployment of new decisions. This access to fresh data enables exploration of decisions and features that may not be adequately represented in the training data. However, it is also essential to strike a balance between exploration and exploitation, as we strive to ensure that the selected decisions result in good overall performance, rather than being purely exploratory. This tradeoff between exploration and exploitation arises in the data-to-decision pipeline, involving various optimization and decision-making processes. Our objective is to comprehensively understand how to design online learning algorithms that effectively incorporate both the machine learning and optimization components.

- **Optimization in multi-agent systems:** In real-world societal challenges, decision-making processes frequently entail the involvement of multiple self-interested agents. It is crucial to thoroughly investigate the interactions among these agents and the optimization problems that arise in multi-agent systems. In particular, the development of scalable solutions to effectively address optimization challenges in multi-agent systems is a key area of focus.

The main objective of this thesis is to comprehensively investigate the impact of optimization on various components within the data-to-decision pipeline, and to develop scalable algorithms that seamlessly integrate optimization with machine learning and data collection processes. While individual artificial intelligence components in the data-to-deployment pipeline have been thor-

Figure 1.3: I collaborated with ARMMAN to deploy my *decision-focused learning* algorithm that integrates machine learning and optimization to the mobile health program with ARMMAN, and the algorithm is used by more than 100,000 people in India with a 30% more improvement in health information engagement. During the field study, I visited the ARMMAN office (left) and the region where ARMMAN operates the maternal mobile health program to serve under-resourced communities in Mumbai, India (middle). I followed the health workers to physically visit the families enrolled in the health program (right). The health workers talked to mothers and provided preventive care information and assistance to increase access to health information and reduce maternal and child mortality/morbidity.

oughly researched in diverse applications, the holistic integration of these components has received relatively less attention. By incorporating optimization into machine learning, data collection, and multi-agent systems, domain-specific solutions can be designed to effectively handle diverse constraints and knowledge in different application domains, such as public health and environmental sustainability. This thesis addresses the potential challenges and essential components involved in creating data-driven decision-making solutions for deployment in these fields. From a technical standpoint, the integration of optimization poses new challenges in effectively and efficiently blending optimization techniques with machine learning, data collection, and multi-agent systems. This thesis establishes the algorithmic foundations for machine learning and other AI techniques in the presence of various optimization problems and decision-making processes. As a whole, this thesis summarizes the pivotal role of optimization in the data-to-deployment pipeline and showcases its applications in public health and environmental sustainability.

1.3 Thesis Outline

This thesis is divided into three parts, which corresponds to three different gaps in the data-to-deployment pipeline in Figure 1.4.

- **Part I (learning in the presence of optimization)** studies the integration of machine learning and optimization problems to produce actionable and quality-aware solutions in public health and wildlife conservation.
- **Part II (optimization in online learning)** studies using optimization to design online learning algorithms to simultaneously collect data and ensure better theoretical guarantees in public health challenges.
- **Part III (optimization in multi-agent systems)** studies decision making in multi-agent systems via Stackelberg games to design scalable and approximate solutions for wildlife conservation.

The chapters in this thesis are based on materials in the publications [321,323,320,324,327,311,322,325,319,326]. Each chapter includes a related work section to summarize the prior work on the related topics. Figure 1.4 and below summarize the contributions of chapters covered in each part.

1.3.1 Part I: Learning in the Face of Optimization

Effective data-driven decision making requires alignment of machine learning and optimization in the data-to-decision pipeline, but unfortunately most use cases consider the two steps separately. For example, in my collaboration on the maternal health challenge with an Indian non-government organization ARMMAN, we first predict the behavior of expecting and new mothers from historical data, and then optimize limited number of service calls from ARMMAN's call center to boost engagement with their health information program. However, machine learning and optimization

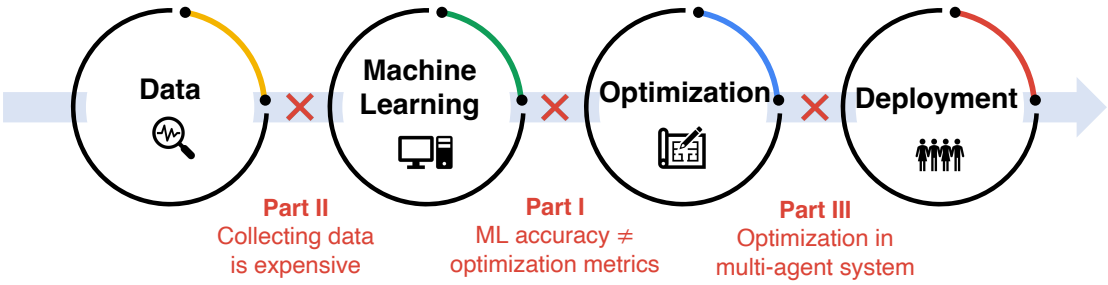


Figure 1.4: The figure summarizes the gaps in the data-to-deployment pipeline. Part I discusses the objective mismatch gap between machine learning and optimization. Part II discusses the challenge of using optimization to design data collection algorithms to improve online learning performance. Part III discusses the scalability challenges of optimization in multi-agent systems and how to design scalable approximate algorithms.

operate separately, and their objectives are often misaligned: machine learning seeks to maximize predictive accuracy, while optimization seeks to optimize decision quality. Improved predictive accuracy does not necessarily result in better decision quality, producing a mismatch, as we truly care about producing the best decisions possible. My research focuses on fixing this misalignment of objectives by integrating machine learning and optimization problems via *decision-focused learning* (DFL) as shown in Figure 1.5. Specifically, (i) **Chapter 2** and **Chapter 3** generalize decision-focused learning to sequential decision problems, followed by a field study result summarized in **Chapter 4**, and (ii) **Chapter 5** and **Chapter 6** highlight and alleviate the scalability issue in decision-focused learning.

Decision-focused learning in sequential decision problems

Many public health problems involve sequential decision making to maximize long-term performance. However, the existing decision-focused learning algorithms only work for non-sequential optimization problems with explicit optimality conditions.

Chapter 2 delves into the implicit optimality conditions in sequential decision problems, treating them as implicit fixed-point equations. This leads to a novel technique for differentiating through

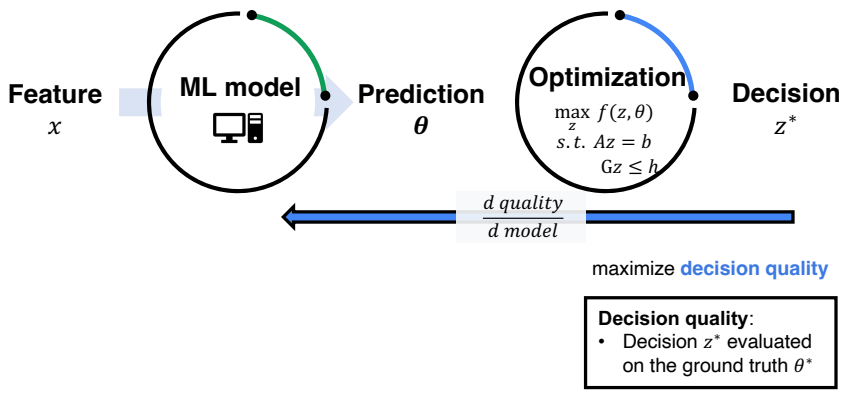


Figure 1.5: Decision-focused learning uses the decision quality of the optimization problem to train the machine learning model.

optimal solutions of sequential decision problems using the implicit function theorem and the policy gradient theorem from reinforcement learning literature, establishing the differentiability of sequential problems and decision-focused learning in such scenarios.

Chapter 3 focuses on restless multi-armed bandits (RMABs), a specific category of sequential decision problems used to model the effect of sequential treatments in public health. Integrating RMABs into the learning tasks for decision-focused learning in public health is challenging due to the inherent computational complexity of solving RMABs optimally. However, this chapter resolves this computational issue by using an approximate index-based solution that can be solved in polynomial time. Furthermore, this chapter demonstrates the differentiability of the index-based solution, successfully enabling decision-focused learning in RMAB problems.

Field study in maternal and child health

Chapter 4 covers the real-world field study result of decision-focused learning in the maternal and child health challenge formulated as a restless multi-armed bandit problem. I have collaborated with ARMMAN to study the maternal health challenge and boost engagement with their mobile health information program by optimizing service calls. We conducted a field study to compare the

proposed decision-focused learning algorithm with other non-decision-focused learning algorithms on a cohort of 9000 beneficiaries registered between April 2022 to June 2022. Our decision-focused learning algorithm significantly outperforms the non-decision-focused learning algorithm. This result has led to the first real-world deployment of decision-focused learning with ARMMAN; with estimated 100,000 beneficiaries in under-resourced communities benefiting from using decision-focused learning in boosting engagement with ARMMAN's health program.

Scalability of decision-focused learning

Decision-focused learning was previously proposed to train predictive models that maximize decision quality in downstream optimization tasks. However, integrating optimization into the learning process requires repeatedly solving and backpropagating through the optimization problem at every gradient step, which can quickly become computationally intractable as the problem size grows even in the non-sequential setting.

Chapter 5 proposes subsampling decision variables of optimization problems to reduce the dimensionality of the optimization problem in decision-focused learning, with an approximation guarantee on gradient estimate.

Chapter 6 presents a method that uses a lower-dimensional surrogate problem constructed from the original problem with a closed-form expression to reduce the dimensionality and the optimization cost.

All of these methods effectively reduce the computation cost of decision-focused learning, enabling real-world applications by achieving cubic reduction in computation overhead.

1.3.2 Part II: Optimization in Online Learning

Data collection plays a crucial role in the performance of wildlife conservation and public health efforts. In my research, I investigate various types of multi-armed bandit (MAB) problems, where the learner repeatedly queries to learn and optimize the rewards from interactions. Specifically, I focus on stochastic MABs and restless MABs, which are motivated by the domains of wildlife conservation and public health, respectively.

Chapter 7 investigates a stochastic MAB problem with a fixed budget, where multiple arms are pulled to receive feedback at each time step. Unlike standard combinatorial MABs, in this scenario, we observe additive feedback from each arm, which contributes to the final reward metric. For example, in wildlife conservation, we observe rewards from each patrol location, or in public health, we observe the tuberculosis treatment effect in individual districts of a large state. I demonstrate that the additive decomposed feedback helps reduce uncertainty in Gaussian process regression and enables faster convergence. This leads to the development of an online algorithm called decomposed-GP-UCB for stochastic MAB problems with continuous pulling actions.

Chapter 8 studies restless multi-armed bandits (RMABs) as an extension of MABs to understand the impact of sequential decisions in public health. I propose an online algorithm that leverages the temporal dependency in RMABs to learn the unknown transition dynamics, such as treatment effects and long-term health impacts. The algorithm yields a frequentist regret bound of $O(\sqrt{T \log T})$, which generalizes the state-of-the-art Bayesian regret bound to a broader range of RMAB problems.

Chapter 9 extends the concept of decomposed feedback in MAB problems to non-additive decomposed feedback in online combinatorial optimization problems. I propose an online algorithm that uses a predictive model to achieve a sublinear regret guarantee in online combinatorial optimization problems. The result highlights the benefits of utilizing decomposed feedback in online

combinatorial optimization, a generalized version of MAB problems, to improve regret bounds.

1.3.3 Part III: Optimization in Multi-agent Systems

Real-world challenges often involve multiple roles with different interests and require sequential decision making. For instance, in wildlife conservation, patrollers in national parks must choose a patrol strategy to protect endangered wildlife, while poachers respond to the patrol plan to launch attacks on animals. Part III focuses on using Stackelberg games to understand sequential decision making in multi-agent systems and design scalable algorithms for efficient computation of near-optimal equilibria.

Chapter 10 studies Stackelberg games with multiple followers, each having their own interests. I propose a technique to differentiate through the equilibrium reached by multiple followers, estimating the gradient of the leader's payoff obtained from the equilibrium. This method results in the first gradient-based algorithm for solving Stackelberg games with multiple followers, which outperforms the standard bilevel formulation for solving Stackelberg games.

Chapter 11 focuses on Stackelberg games with different response models for the followers, and develops algorithms for defending against attackers with varying behaviors. I propose equilibrium refinement algorithms for Stackelberg games with arbitrary resource constraints, which identifies robust solutions against potential uncertainty in the response behavior of followers. This algorithm can be applied to applications such as scheduling security resources to protect vulnerable targets.

Part I

Decision-focused Learning: Learning in the Face of Optimization

2

Decision-focused Learning in Sequential Decision Problems

2.1 Introduction

Predict-then-optimize[91,33] is a framework for solving optimization problems with unknown parameters. Given such a problem, we first train a predictive model to predict the missing parameters from

problem features. Our objective is to maximize the resulting decision quality when the optimization problem is subsequently solved with the predicted parameters[274,237]. Recent work on the *decision-focused learning* approach[80,338] embeds the optimization problem[12,3,32] into the training pipeline and trains the predictive model end-to-end to optimize the final decision quality. Compared with a more traditional "two-stage" approach which maximizes the predictive accuracy of the model (rather than the final decision quality), the decision-focused learning approach can achieve a higher solution quality and generalize better to unseen tasks.

This paper studies the predict-then-optimize framework in *sequential* decision problems, formulated as Markov decision processes (MDPs), with unknown parameters. In particular, at training time, we are given trajectories and environment features from "training MDPs." Our goal is to learn a predictive model which maps from environment features to missing parameters based on these trajectories that generalizes to unseen test MDPs that have features, but not trajectories. The resulting "predicted" training and test MDPs are solved using deep reinforcement learning (RL) algorithms, yielding policies that are then evaluated by offline off-policy evaluation (OPE) as shown in Figure 2.1. This fully offline setting is motivated by real-world applications such as wildlife conservation and tuberculosis treatment where no simulator is available. However, such domains offer past ranger patrol trajectories and environmental features of individual locations from conservation parks for generalization to other unpatrolled areas. These settings differ from those considered in transfer-RL[234,299,187,276] and meta-RL[318,82,100,363,330] because we generalize across different MDPs by explicitly predicting the mapping function from features to missing MDPs parameters, while transfer/meta RL achieve generalization by learning hidden representation of different MDPs implicitly with trajectories.

The main contribution of this paper is to extend the decision-focused learning approach to MDPs with unknown parameters, embedding the MDP problems in the predictive model training pipeline. To perform this embedding, we study two common types of optimality conditions in

MDPs: a Bellman-based approach where mean-squared Bellman error is minimized, and a policy gradient-based approach where the expected cumulative reward is maximized. We convert these optimality conditions into their corresponding Karush–Kuhn–Tucker (KKT) conditions, where we can backpropagate through the embedding by differentiating through the KKT conditions. However, existing techniques from decision-focused learning and differentiating through KKT conditions do not directly apply as the size of the KKT conditions of sequential decision problems grow linearly in the number of states and actions, which are often combinatorial or continuous and thus become intractable.

We identify and resolve two computational challenges in applying decision-focused learning to MDPs, that come up in both optimality conditions: (i) the large state and action spaces involved in the optimization reformulation make differentiating through the optimality conditions intractable and (ii) the high-dimensional policy space in MDPs, as parameterized by a neural network, makes differentiating through a policy expensive. To resolve the first challenge, we propose to sample an estimate of the first-order and second-order derivatives to approximate the optimality and KKT conditions. We prove such a sampling approach is unbiased for both types of optimality conditions. Thus, we can differentiate through the approximate KKT conditions formed by sample-based derivatives. Nonetheless, the second challenge still applies—the sampled KKT conditions are expensive to differentiate through due to the dimensionality of the policy space when model-free deep RL methods are used. Therefore, we propose to use a low-rank approximation to further approximate the sample-based second-order derivatives. This low-rank approximation reduces both the computation cost and the memory usage of differentiating through KKT conditions.

We empirically test our decision-focused algorithms on three settings: a grid world with unknown rewards, and snare-finding and Tuberculosis treatment problems where transition probabilities are unknown. Decision-focused learning achieves better OPE performance in unseen test MDPs than two-stage approach, and our low-rank approximations significantly scale-up decision-focused

learning.

2.2 Related Work

Differentiable optimization Amos et al.[11] propose using a quadratic program as a differentiable layer and embedding it into deep learning pipeline, and Agrawal et al.[3] extend their work to convex programs. Decision-focused learning[80,338] focuses on the predict-then-optimize[91,33] framework by embedding an optimization layer into training pipeline, where the optimization layers can be convex[80], linear[338,206], and non-convex[247,324]. Unfortunately, these techniques are of limited utility for sequential decision problems because their formulations grow linearly in the number of states and actions and thus differentiating through them quickly becomes infeasible. Amos et al.[12] avoid this issue by studying model-predictive control but limited to quadratic-form actions, reducing the dimensionality. Karkus et al.[165] differentiate through an algorithm by unrolling and relaxing all the strict operators by soft operators. Futoma et al.[111] deal with large optimality conditions by differentiating through the last step of the value-iteration algorithm only. Instead, our approach does not rely on any MDP solver structure. We combine sampling and a low-rank approximation to form an unbiased estimate of the optimality conditions to differentiate through, and show that the approach of Futoma et al.[111] is included in ours as a special case.

Predict-then-optimize and offline reinforcement learning The idea of planning under a predicted MDP arises in model-based RL as *certainty equivalence*[183]. It has been extended to offline settings[167,355], who learn a pessimistic MDP before solving for the policy. Our setting differs because of the presence of features and train-test split—our test MDPs are completely fresh *without any trajectories*. Our setting also resembles meta RL (e.g.,[318,82,100,363,330]) and transfer RL (e.g.,[234,299,187,276].) Meta RL focuses on training a "meta policy" for a distribution of tasks (MDPs), leveraging trajectories for each. Transfer RL works by extracting transferable knowledge from source

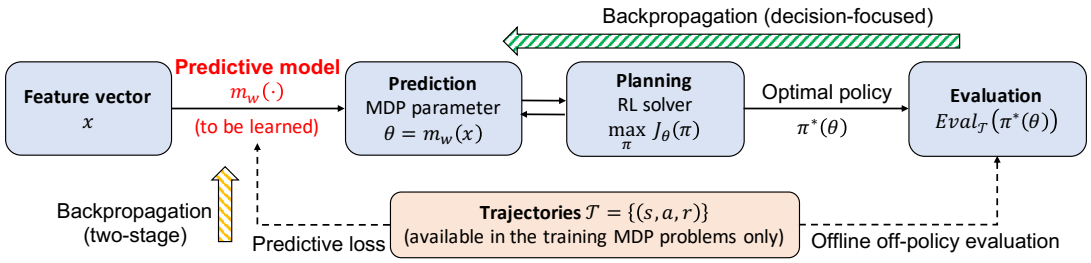


Figure 2.1: We consider learning a predictive model to map from features to unknown MDP parameters and obtaining a policy by solving the predicted MDP with RL. Two-stage learning learns the predictive model by minimizing a predictive loss function, whereas decision-focused learning is trained end-to-end to maximize the final off-policy evaluation performance.

MDPs to target MDPs using trajectories. In contrast to these two paradigms, ours explicitly trains a predictive model (which maps problem features to missing MDP parameters) to generalize knowledge learned from the training set to the testing set using *problem features, not trajectories*.

2.3 Problem Statement

In this paper, we consider learning a predictive model to infer the missing parameters in a sequential decision-making task (formulated as MDPs) using the predict-then-optimize framework. Each MDP is defined by a tuple $(\mathcal{S}, \boldsymbol{s}_0, \mathcal{A}, T, R)$ with an initial state $\boldsymbol{s}_0$, a possibly infinite state space $\mathcal{S}$ and possibly infinite action space $\mathcal{A}$. We assume some parameters are missing in each MDP, which could be any portion of the transition function T and the reward function R. We denote the missing parameters vector by θ^*. Additionally, we assume there are problem features x associated with each MDP, where (θ^*, x) is correlated and drawn from the same unknown, but fixed, distribution*.

We are given a set of training MDPs and a set of test MDPs, each with missing parameters θ^* and

*Examples of the missing parameters θ^* include the poaching risk of different locations in wildlife conservation and the transition probability of patients' healthiness in healthcare problems, where the corresponding problem features are terrain features of different locations and the characteristics of different patients that are correlated to the missing parameters, respectively. These correlated features allow us to predict the missing parameters even if we do not have any trajectories of the MDP.

features x. Each training MDP is accompanied by a set of trajectories $\mathcal{T}$ performed by a behavior policy π_{beh}, consisting of a sequence of states, actions and rewards. In the test MDPs, trajectories from the behavior policy are hidden at test time. These testing MDPs are considered fresh instances that we have to generate a policy without using any trajectories. Thus, at training time, we learn a predictive model m_w to map from features to missing parameters; at test time, we apply the same model to make predictions and plan accordingly without using trajectories.

Formally, our goal is to learn a predictive model m_w to predict the missing parameters $\theta = m_w(x)$. The predicted parameters are used to solve the test MDPs, yielding the policy $\pi^*(m_w(x))$. Lastly, we use an offline evaluation metric to measure the performance of the new policy. The evaluation metric is known as offline off-policy evaluation (OPE)[257,303] and counterfactual inference in sequential experiments[85,68] to evaluate the treatment effect of a new policy. The framework of the entire process is illustrated in Figure 2.1.

Offline off-policy evaluation We evaluate a policy π in a fully offline setting with trajectories $\mathcal{T} = \{\tau_i\}, \tau_i = (s_{i1}, a_{i1}, r_{i1}, \cdots, s_{ih}, a_{ih}, r_{ih})$ generated from the MDP using behavior policy π_{beh}. We use an OPE metric used by Futoma et al.[111] — we evaluate a policy π and trajectories $\mathcal{T}$ as:

$$\text{Eval}_{\mathcal{T}}(\pi) := V^{\text{CWPDIS}}(\pi) - \frac{\lambda_{\text{ESS}}}{\sqrt{\text{ESS}(\pi)}} \tag{2.1}$$

where $V^{\text{CWPDIS}}(\pi) := \sum_{t=1}^{h} \gamma^t \frac{\sum_i r_{it}\rho_{it}(\pi)}{\sum_i \rho_{it}(\pi)}$ and $\text{ESS}(\pi) := \sum_{t=1}^{h} \frac{(\sum_i \rho_{it})^2}{\sum_i \rho_{it}^2}$, and $\rho_{it}(\pi)$ is the ratio of the proposed policy and the behavior policy likelihoods up to time t: $\rho_{it}(\pi) := \prod_{t'=1}^{t} \frac{\pi(a_{it'}|s_{it'})}{\pi_{\text{beh}}(a_{it'}|s_{it'})}$.

Optimization formulation Given a set of training features and trajectories D_{train} denoted by $\{(x_i, \mathcal{T}_i)\}_{i \in I_{\text{train}}}$, our goal is to learn a predictive model m_w to optimize the training performance:

$$\max_{w} \quad \mathop{\mathbb{E}}_{(x,\mathcal{T}) \in D_{\text{train}}} [\text{Eval}_{\mathcal{T}}(\pi^*(m_w(x)))] \tag{2.2}$$

The testing performance is evaluated on the unseen test set $D_{\text{test}} = \{(x_i, \mathcal{T}_i)\}_{i \in I_{\text{test}}}$ with trajectories hidden from training, and only used for evaluation: $\mathop{\mathbb{E}}_{(x,\mathcal{T}) \in D_{\text{test}}} [\text{Eval}_{\mathcal{T}}(\pi^*(m_w(x)))]$.

2.4 Two-stage Learning

To learn the predictive model m_w from trajectories, the standard approach is to minimize an expected predictive loss, which is defined by comparing the prediction $\theta = m_w(x)$ with the trajectories $\mathcal{T}$:

$$\min_{w} \quad \mathop{\mathbb{E}}_{(x,\mathcal{T}) \sim \mathcal{D}_{\text{train}}} \mathcal{L}(\theta, \mathcal{T}) \qquad \text{where} \quad \mathcal{L}(\theta, \mathcal{T}) = \mathop{\mathbb{E}}_{\tau \sim \mathcal{T}} \ell_\theta(\tau), \quad \theta = m_w(x) \tag{2.3}$$

For example, when the rewards are missing, the loss could be the squared error between the predicted rewards and the actual rewards we see in the trajectories for each individual step. When the transition probabilities are missing, the loss could be defined as the negative log-likelihood of seeing the trajectories in the training set.

In the first stage, to train the predictive model, we run gradient descent to minimize the loss function defined in Equation (2.3) and make prediction on the model parameter $\theta = m_w(x)$ of each problem. In the second stage, we solve each MDP problem with the predicted parameter θ using an RL algorithm to generate the optimal policy $\pi^*(\theta)$. However, predictive loss and the final evaluation metric are commonly misaligned especially in deep learning problems with imbalanced data [142,185,155,51]. This motivates us to learn the predictive model end-to-end and therefore avoid the

misalignment.

2.5 Decision-focused Learning in Sequential Decision Problems

In this section, we present our main contribution, decision-focused learning in sequential decision problems, as illustrated in Figure 2.1. Decision-focused learning integrates an MDP problem into the training pipeline to directly optimize the final performance. Instead of relying on a predictive loss to train the predictive model m_w, we can directly optimize the objective in Equation (2.2) by running end-to-end gradient descent to update the predictive model m_w:

$$\frac{d\,\text{Eval}(\pi^*)}{dw} = \frac{d\,\text{Eval}(\pi^*)}{d\pi^*}\frac{d\pi^*}{d\theta}\frac{d\theta}{dw} \tag{2.4}$$

We assume the policy $\pi^*(\theta)$ is either stochastic and smooth with respect to the change in the parameter θ, which is common in settings with continuous state or action spaces, or that an appropriate regularization term [127,128] is used to improve the smoothness of the policy. More discussions about the smoothness can be found in Appendix A.2.1.

This gradient computation requires us to back-propagate from the final evaluation through the MDP layer to the predictive model m_w that we want to update. The major challenge in Equation (2.4) is to compute $\frac{d\pi^*}{d\theta}$, which involves differentiating through an MDP layer solved by an RL algorithm. In the following section, we first discuss two different optimality conditions in MDPs, which are later used to convert into KKT conditions and differentiate through to compute $\frac{d\pi^*}{d\theta}$. We then discuss two computational challenges associated with the derivative computation.

2.5.1 Optimality Conditions in MDPs

When the predicted model parameter $\theta = m_w(x)$ is given, the MDP can be solved by any RL algorithm to get an optimal policy π^*. Here we discuss two common optimality conditions in MDPs,

differing by the use of policy gradient or Bellman equation:

Definition 1 (Policy gradient-based optimality condition)**.** *Defining $J_\theta(\pi)$ to be the expected cumulative reward under policy π, the optimality condition of the optimal policy π^* is:*

$$\pi^* = \arg\max_{\pi} J_\theta(\pi) \qquad \textit{where} \quad J_\theta(\pi) := \mathop{\mathbb{E}}_{\tau\sim\pi,\theta} G_\theta(\tau) \tag{2.5}$$

where $G_\theta(\tau)$ is the discounted value of trajectory τ given parameter θ, and the expectation is taken over the trajectories following the optimal policy and transition probability (as part of θ).

Definition 2 (Bellman-based optimality condition)**.** *Defining $J_\theta(\pi)$ to be the mean-squared Bellman error*[†] *under policy π, the optimality condition of the optimal policy π^* (valuation function) is:*

$$\pi^* = \arg\min_{\pi} J_\theta(\pi) \qquad \textit{where} \quad J_\theta(\pi) := \mathop{\mathbb{E}}_{\tau\sim\pi,\theta} \frac{1}{2}\delta_\theta^2(\tau,\pi) \tag{2.6}$$

where $\delta_\theta(\tau,\pi) = \sum_{(s,a,s')\in\tau} Q_\pi(s,a) - R_\theta(s,a) - \gamma \mathop{\mathbb{E}}_{a'\sim\pi} Q_\pi(s',a')$ is the total Bellman error of a trajectory τ, and each individual term $\delta_\theta(\tau,\pi)$ can depend on the parameter θ because the Bellman error depends on the immediate reward R_θ, which can be a part of the MDP parameter θ. The expectation in Equation (2.6) *is taken over all the trajectories generated from policy π and transition probability (as part of θ).*

2.5.2 Backpropagating Through Optimality and KKT Conditions

To compute the derivative of the optimal policy $\pi^*(\theta)$ in an MDP with respect to the MDP parameter θ, we differentiate through the KKT conditions of the corresponding optimality conditions:

[†] We use the same notation J to denote both the expected cumulative reward and the expected Bellman error to simplify the later analysis of decision-focused learning.

Definition 3 (KKT Conditions). *Given objective $J_\theta(\pi)$ in an MDP problem, since the policy parameters are unconstrained, the necessary KKT conditions can be written as:* $\nabla_\pi J_\theta(\pi^*) = 0$.

In particular, computing the total derivative of KKT conditions gives:

$$\begin{aligned} 0 = \frac{d}{d\theta}\nabla_\pi J_\theta(\pi^*) &= \frac{\partial}{\partial\theta}\nabla_\pi J_\theta(\pi^*) + \frac{\partial}{\partial\pi}\nabla_\pi J_\theta(\pi^*)\frac{d\pi^*}{d\theta} = \nabla^2_{\theta\pi} J_\theta(\pi^*) + \nabla^2_\pi J_\theta(\pi^*)\frac{d\pi^*}{d\theta} \\ \implies \frac{d\pi^*}{d\theta} &= -(\nabla^2_\pi J_\theta(\pi^*))^{-1}\nabla^2_{\theta\pi} J_\theta(\pi^*) \end{aligned} \tag{2.7}$$

We can use Equation (2.7) to replace the term $\frac{d\pi^*}{d\theta}$ in Equation (2.4) to compute the full gradient to back-propagate from the final evaluation to the predictive model parameterized by w:

$$\frac{d\,\mathrm{Eval}(\pi^*)}{dw} = -\frac{d\,\mathrm{Eval}(\pi^*)}{d\pi^*}(\nabla^2_\pi J_\theta(\pi^*))^{-1}\nabla^2_{\theta\pi} J_\theta(\pi^*)\frac{d\theta}{dw} \tag{2.8}$$

2.5.3 Computational Challenges in Backward Pass

Unfortunately, although we can write down and differentiate through the KKT conditions analytically, we cannot explicitly compute the second-order derivatives $\nabla^2_\pi J_\theta(\pi^*)$ and $\nabla^2_{\theta\pi} J_\theta(\pi^*)$ in Equation (2.8) due to the following two challenges:

Large state and action spaces involved in optimality conditions The objectives $J_\theta(\pi^*)$ in Definition 1 and Definition 2 involve an expectation over all possible trajectories, which is essentially an integral and is intractable when either the state or action space is continuous. This prevents us from explicitly verifying optimality and writing down the two derivatives $\nabla^2_\pi J_\theta(\pi^*)$ and $\nabla^2_{\theta\pi} J_\theta(\pi^*)$.

High-dimensional policy space parameterized by neural networks In MDPs solved by model-free deep RL algorithms, the policy space $\pi \in \Pi$ is often parameterized by a neural

network, which has a significantly larger number of variables than standard optimization problems. This large dimensionality makes the second-order derivative $\nabla^2_\pi J_\theta(\pi^*) \in \mathbb{R}^{\dim(\pi)\times\dim(\pi)}$ intractable to compute, store, or invert.

2.6 Sampling Unbiased Derivative Estimates

In both policy gradient–based and Bellman–based optimality conditions, the objective is implicitly given by an expectation over all possible trajectories, which could be infinitely large when either state or action space is continuous. This same issue arises when expressing such an MDP as a linear program — there are infinitely many constraints, making it intractable to differentiate through.

Inspired by the policy gradient theorem, although we cannot compute the exact gradient of the objective, we can sample a set of trajectories $\tau = \{s_1, a_1, r_1, \ldots, s_h, a_h, r_h\}$ from policy π and model parameter θ with finite time horizon h. Denoting $p_\theta(\tau, \pi)$ to be the likelihood of seeing trajectory τ, we can compute an unbiased derivative estimate for both optimality conditions:

Theorem 1 (Policy gradient-based unbiased derivative estimate)**.** *We follow the notation of Definition 1 and define* $\Phi_\theta(\tau, \pi) = \sum_{i=1}^{h}\sum_{j=i}^{h} \gamma^j R_\theta(s_j, a_j) \log \pi(a_i|s_i)$. *We have:*

$$\nabla_\pi J_\theta(\pi) = \mathop{\mathbb{E}}_{\tau\sim\pi,\theta}\left[\nabla_\pi \Phi_\theta(\tau, \pi)\right] \implies \begin{aligned} \nabla^2_\pi J_\theta(\pi) &= \mathop{\mathbb{E}}_{\tau\sim\pi,\theta}\left[\nabla_\pi \Phi_\theta \cdot \nabla_\pi \log p_\theta^\top + \nabla^2_\pi \Phi_\theta\right] \\ \nabla^2_{\theta\pi} J_\theta(\pi) &= \mathop{\mathbb{E}}_{\tau\sim\pi,\theta}\left[\nabla_\pi \Phi_\theta \cdot \nabla_\theta \log p_\theta^\top + \nabla^2_{\theta\pi} \Phi_\theta\right] \end{aligned} \tag{2.9}$$

Theorem 2 (Bellman-based unbiased derivative estimate)**.** *We follow the notation in Definition 2 to define* $J_\theta(\pi) = \frac{1}{2}\mathop{\mathbb{E}}_{\tau\sim\pi,\theta}\left[\delta^2_\theta(\tau, \pi)\right]$. *We have:*

$$\nabla_\pi J_\theta(\pi) = \mathop{\mathbb{E}}_{\tau\sim\pi,\theta}\left[\delta \nabla_\pi \delta + \frac{1}{2}\delta^2 \nabla_\pi \log p_\theta\right] \implies \nabla^2_\pi J_\theta(\pi) = \mathop{\mathbb{E}}_{\tau\sim\pi,\theta}\left[\nabla_\pi \delta \nabla_\pi \delta^\top + O(\delta)\right]$$

$$\nabla^2_{\theta\pi} J_\theta(\pi) = \mathop{\mathbb{E}}_{\tau\sim\pi,\theta}\left[\nabla_\pi \delta \nabla_\theta \delta^\top + \left(\nabla_\pi \delta \nabla_\theta \log p_\theta^\top + \nabla_\pi \log p_\theta \nabla_\theta \delta^\top + \nabla^2_{\theta\pi}\delta\right)\delta + O(\delta^2)\right] \quad (2.10)$$

For the latter, we apply the fact that at the near-optimal policy, the Bellman error is close to 0 and thus each individual component $\delta(\tau, \pi)$ is small to simplify the analysis. Refer to the appendix for the full derivations of Equations (2.9) and (2.10).

Equations (2.9) and (2.10) provide a sampling approach to compute the second-order derivatives, avoiding computing an expectation over the large trajectory space. We can use the optimal policy derived in the forward pass and the predicted parameters θ to run multiple simulations to collect a set of trajectories. These trajectories from predicted parameters can be used to compute each individual derivative in Equations (2.9) and (2.10).

2.7 Resolving High-dimensional Derivatives by Low-rank Approximation

Section 2.6 provides sampling approaches to compute an unbiased estimate of second-order derivatives. However, since the dimensionality of the policy space $\dim(\pi)$ is often large, we cannot explicitly expand and invert $\nabla^2_\pi J_\theta(\pi^*)$ to compute $\nabla^2_\pi J_\theta(\pi^*)^{-1}\nabla^2_{\theta\pi} J_\theta(\pi^*)$, which is an inevitable step toward computing the full gradient of decision-focused learning in Equation (2.8). In this section, we discuss various ways to approximate $\nabla^2_\pi J_\theta(\pi^*)$ and how we use low-rank approximation and Woodbury matrix identity[340] to efficiently invert the sampled Hessian without expanding the matrices. Let $n := \dim(\pi)$ and $k \ll n$ to be the number of trajectories we sample to compute the derivatives.

2.7.1 Full Hessian Computation

In Equations (2.9) and (2.10), we can apply auto-differentiation tools to compute all individual derivatives in the expectation. However, this works only when the dimensionality of the policy space

$\pi \in \Pi$ is small because the full expressions in Equations (2.9) and (2.10) involve computing second-order derivatives , e.g., $\nabla^2_\pi \Phi_\theta$ in Equation (2.10), which is still challenging to compute and store when the matrix size $n \times n$ is large. The computation cost is $O(n^2 k) + O(n^\omega)$ dominated by computing all the Hessian matrices and the matrix inversion with $2 \leq \omega \leq 2.373$ the complexity order of matrix inversion.

2.7.2 Approximating Hessian by Constant Identity Matrix

One naive way to approximate the Hessian $\nabla^2_\pi J_\theta(\pi^*)$ is to simply use a constant identity matrix cI. We choose $c < 0$ for the policy gradient–based optimality in Definition 1 because the optimization problem is a maximization problem and thus is locally concave at the optimal solution, whose Hessian is negative semi-definite. Similarly, we choose $c > 0$ for the Bellman–based optimality in Definition 2. This approach is fast, easily invertible. Moreover, in the Bellman version, Equation (2.8) is equivalent to the idea of differentiating through the final gradient of Bellman error as proposed by Futoma et al. [111][‡]. However, this approach ignores the information provided by the Hessian term, which can often lead to instability as we later show in the experiments. In this case, the computation complexity is dominated by computing $\nabla^2_{\theta\pi} J_\theta(\pi^*)$, which requires $O(nk)$.

2.7.3 Low-rank Hessian Approximation and Application of Woodbury Matrix Identity

A compromise between the full Hessian and using a constant matrix is approximating the second-order derivative terms in Equations (2.9) and (2.10) by constant identity matrices, while computing the first-order derivative terms with auto-differentiation. Specifically, given a set of k sampled

[‡]The gradient of Bellman error can be written as $\nabla_\pi J_\theta(\pi^*)$ where the policy π is the parameters of the value function approximator and J is defined as the expected Bellman error. The derivative of the final gradient can be written as $\nabla_w(\nabla_\pi J_\theta(\pi^*)) \quad = \quad \nabla^2_{\theta\pi} J_\theta(\pi^*) \frac{d\theta}{dw}$ by chain rule, which matches the last three terms in Equation 2.8 when the Hessian is approximated by an identity matrix.

trajectories $\{\tau_1, \tau_2, \cdots, \tau_k\}$, Equations (2.9) and (2.10) can be written and approximated in the following form:

$$\nabla_\pi^2 J_\theta(\pi) \approx \frac{1}{k}\sum_{i=1}^{k}\left(u_i v_i^\top + H_i\right) \approx \frac{1}{k}\sum_{i=1}^{k}\left(u_i v_i^\top + cI\right) = UV^\top + cI \tag{2.11}$$

where $U = [u_1, u_2, \cdots, u_k]/\sqrt{k} \in \mathbb{R}^{n\times k}$, $V = [v_1, v_2, \cdots, v_k]/\sqrt{k} \in \mathbb{R}^{n\times k}$ and $u_i, v_i \in \mathbb{R}^n$ correspond to the first-order derivatives in Equations (2.9) and (2.10), and H_i corresponds to the remaining terms that involve second-order derivatives. We use a constant identity matrix to approximate H_i, while explicitly computing the remaining parts to increase accuracy.

However, we still cannot explicitly expand $UV^\top \in \mathbb{R}^{n\times n}$ since the dimensionality is too large. Therefore, we apply Woodbury matrix identity[34o] to invert Equation (2.11):

$$(\nabla_\pi^2 J_\theta(\pi))^{-1} \approx (UV^\top + cI)^{-1} = \frac{1}{c}I - \frac{1}{c}U(cI - V^\top U)^{-1}V^\top \tag{2.12}$$

where $V^\top U \in \mathbb{R}^{k\times k}$ can be efficiently computed with much smaller $k \ll n$. When we compute the full gradient for decision-focused learning in Equation (2.8), we can then apply matrix-vector multiplication without expanding the full high-dimensional matrix, which results in a computation cost of $O(nk + k^\omega)$ that is much smaller than the full computation cost $O(n^2k + n^\omega)$.

The full algorithm for decision-focused learning in MDPs is illustrated in Algorithm 1[§].

2.8 Example MDP Problems with Missing Parameters

Gridworld with unknown cliffs We consider a Gridworld environment with a set of training and test MDPs. Each MDP is a 5×5 grid with a start state located at the bottom left corner and

[§]The implementation of Algorithm 1 can be found in https://github.com/guaguakai/decision-focused-RL

Algorithm 1: Decision-focused Learning for MDP Problems with Missing Parameters

1 **Parameter:** Training set $\mathcal{D}_{\text{train}} = \{(x_i, \mathcal{T}_i)\}_{i \in I_{\text{train}}}$, learning rate α, regularization $\lambda = 0.1$
2 **Initialization:** Initialize predictive model $m_w : \mathcal{X} \to \Theta$ parameterized by w
3 **for** *epoch* $\in \{1, 2, \dots\}$, *each training instance* $(x, \mathcal{T}) \in \mathcal{D}_{train}$ **do**
4 **Forward:** Make prediction $\theta = m_w(x)$. Compute two-stage loss $\mathcal{L}(\theta, \mathcal{T})$. Run model-free RL to get an optimal policy $\pi^*(\theta)$ on MDP problem using parameter θ.
5 **Backward:** Sample a set of trajectories under θ, π^* to compute $\nabla^2_{\pi} J_\theta(\pi^*)$, $\nabla^2_{\theta\pi} J_\theta(\pi^*)$
6 **Gradient step:** Set $\Delta w = \frac{d\ \text{Eval}_{\mathcal{T}}(\pi^*)}{dw} - \lambda \frac{d\mathcal{L}(\theta, \mathcal{T})}{dw}$ by Equation (2.8) with predictive loss $\mathcal{L}$ as regularization. Run gradient ascent to update model: $w \leftarrow w + \alpha \Delta w$
7 **Return:** Predictive model m_w.

a safe state with reward drawn from $\mathcal{N}(5, 1)$ located at the top right corner. Each intermediate state has a reward associated with it, where most of them give the agent a reward drawn from $\mathcal{N}(0, 1)$ but 20% of the them are cliffs and give $\mathcal{N}(-10, 1)$ penalty to the agent. The agent has no prior information about the reward of each grid cell (i.e., the reward functions of the MDPs are unknown), but has a feature vector per grid cell correlated to the reward, and a set of historical trajectories from the training MDPs. The agent learns a predictive model to map from the features of a grid cell to its missing reward information, and the resulting MDP is used to plan. Since the state and action spaces are both finite, we use tabular value-iteration [295] to solve the MDPs.

Partially observable snare-finding problems with missing transition function

We consider a set of *synthetic* conservation parks, each with 20 sites, that are vulnerable to poaching activities. Each site in a conservation park starts from a *safe* state and has an unknown associated probability that a poacher places a snare at each time step. Motivated by [349], we assume a ranger who can visit one site per time step and observes whether a snare is present. If a snare is present, the

ranger removes it and receives reward 1. Otherwise, the ranger receives reward of -1. The snare can stay in the site if the ranger does not remove it, which makes the snare-finding problem a sequential problem rather than a multi-armed bandit problem. As the ranger receives no information about the sites that they do not visit, the MDP belief state is the ranger's belief about whether a snare is present. The ranger uses the features of each site and historical trajectories to learn a predictive model of the missing transition probability of a snare being placed. Since the belief state is continuous and the action space is discrete, given a predictive model of the missing transition probability, the agent uses double deep Q-learning (DDQN)[309] to solve the predicted MDPs.

Partially observable patient treatment problems with missing transition probability We consider a version of the Tuberculosis Adherence Problem explored in[210]. Given that the treatment for tuberculosis requires patients to take medications for an extended period of time, one way to improve patient adherence is Directly Observed Therapy, in which a healthcare worker routinely checks in on the patient to ensure that they are taking their medications. In our problem, we consider 5 *synthetic* patients who have to take medication for 30 days. Each day, a healthcare worker chooses one patient to intervene on. They observe whether that patient is adhering or not, and improve the patient's likelihood of adhering on that day, where we use the number of adherent patients as the reward to the healthcare worker. Whether a patient actually adheres or not is determined by a transition matrix that is randomly drawn from a fixed distribution inspired by[170]. The aim of the prediction stage is to use the features associated with each patient, e.g., patient characteristics, to predict the missing transition matrices. The aim of the RL stage is then to create an intervention strategy for the healthcare worker such that the sum of patient adherence over the 30-day period is maximised. Due to partial observability, we convert the problem to its continuous belief state equivalence and solve it using DDQN.

Please refer to Appendix A.3 for more details about problem setup in all three domains.

Table 2.1: OPE performances of different methods on the test MDPs averaged over 30 independent runs. Decision-focused learning methods consistently outperform two-stage approach, with some exception using identity matrix based Hessian approximation which may lead to high gradient variance.

	Gridworld		Snare		Tuberculosis	
Trajectories	Random	Near-optimal	Random	Near-optimal	Random	Near-optimal
TS	-12.0 ± 1.3	4.2 ± 0.8	0.8 ± 0.3	3.7 ± 0.3	35.8 ± 1.5	38.7 ± 1.6
PG-Id	-11.7 ± 1.2	5.7 ± 0.8	-0.1 ± 0.3	3.6 ± 0.3	38.4 ± 1.5	40.7 ± 1.7
Bellman-Id	-9.6 ± 1.4	4.6 ± 0.7	0.7 ± 0.4	3.6 ± 0.3	39.1 ± 1.7	40.8 ± 1.7
PG-W	-11.2 ± 1.2	5.5 ± 0.8	1.2 ± 0.4	4.8 ± 0.3	38.4 ± 1.5	40.8 ± 1.7
Bellman-W	-11.3 ± 1.4	4.8 ± 0.8	1.5 ± 0.4	4.3 ± 0.3	38.6 ± 1.6	41.1 ± 1.7

2.9 Experimental Results and Discussion

In our experiments, we compare two-stage learning (**TS**) with different versions of decision-focused learning (**DF**) using two different optimality conditions, policy gradient (**PG**) and Bellman equation-based (**Bellman**), and two different Hessian approximations (**Identity**, **Woodbury**) defined in Section 2.7. Computing the **full** Hessian (as in Section 2.7.1) is computationally intractable. Across all three examples, we use 7 training MDPs, 1 validation MDP, and 2 test MDPs, each with 100 trajectories. The predictive model is trained on the training MDP trajectories for 100 epochs. Performance is evaluated under the Off-Policy Evaluation (OPE) metric of Equation (2.1) with respect to the withheld test trajectories. In the following, we will discuss *how* DF variants work compared with TS methods, and explore *why* some methods are better. We use two different trajectories, **random** and **near-optimal**, in the training MDP to model different imbalanced information given to train the predictive model. The results are shown in Table 2.1.

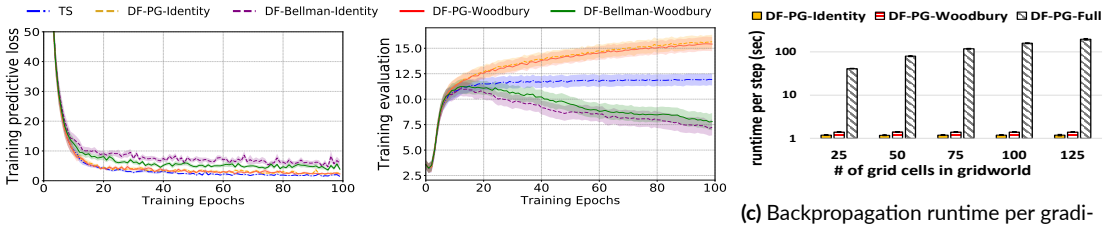
Decision-focused learning with the Woodbury matrix identity outperforms two-stage learning Table 2.1 summarizes the average OPE performance on the test MDPs. We can see that in all of the three problem settings, the best performances are all achieved by decision-focused learning. However, when Hessian approximation is not sufficiently accurate, decision-

focused learning can sometimes perform even worse than two-stage (e.g., PG-Id and Bellman-Id in the snare problem). In contrast, decision-focused methods using a more accurate low-rank approximation and Woodbury matrix identity (i.e., PG-W and Bellman-W), as discussed in Section 2.7.3, dominate two-stage performance in the test MDPs across all settings.

Low predictive loss does not imply a winning policy In Figures 2.2(a), 2.3(a), we plot the predictive loss curve in the training MDPs over different training epochs of Gridworld and snare problems. In particular, two-stage approach is trained to minimize such loss, but fails to win in Table 2.1. Indeed, low predictive loss on the training MDPs does not always imply a high off-policy evaluation on the training MDPs in Figure 2.2(b) due to the misalignment of predictive accuracy and decision quality, which is consistent with other studies in mismatch of predictive loss and evaluation metric [142,185,155,51].

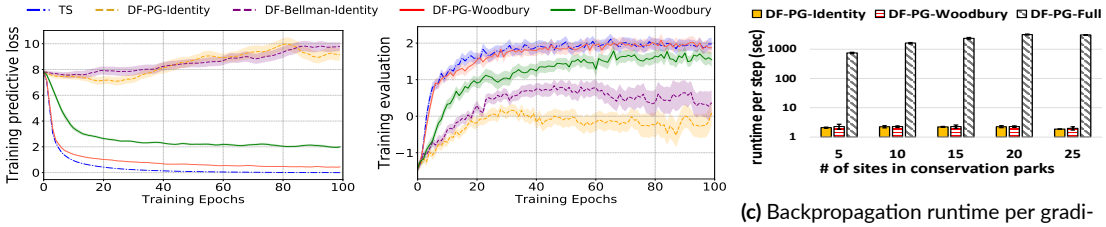
Comparison between different Hessian approximations In Table 2.1, we notice that more inaccurate Hessian approximation (identity) does not always lead to poorer performance. We hypothesize that this is due to the non-convex off-policy evaluation objective that we are optimizing, where higher variance might sometimes help escape local optimum more easily. The identity approximation is more unstable across different tasks and different trajectories given. In Table 2.1, the performance of Bellman-Identity and PG-Identity sometimes lead to wins over two stage and sometimes losses.

Comparison between policy gradient and Bellman-based decision-focused learning We observe that the Bellman-based decision-focused approach consistently outperforms the policy gradient-based approach when the trajectories are random, while the policy gradient-based decision-focused approach mostly achieves better performance when near-optimal trajectories are present. We hypothesize that this is due to the different objectives considered by different optimality



(a) Predictive loss in training MDPs **(b)** Performance in training MDPs **(c)** Backpropagation runtime per gradient step per instance

Figure 2.2: Learning curves of Gridworld problem with near-optimal trajectories. Two-stage minimizes the predictive loss in Figure 2.2(a), but this does not lead to good training performance in Figure 2.2(b). Figure 2.3(c) shows the backpropagation runtime per gradient step per instance of three Hessian approximations, which becomes intractable when trained for multiple instances and multiple epochs.



(a) Predictive loss in training MDPs **(b)** Performance in training MDPs **(c)** Backpropagation runtime per gradient step per instance

Figure 2.3: Learning curves of snare finding problems with random trajectories. Two-stage achieves both low predictive loss in Figure 2.3(a) and high training OPE in Figure 2.3(b), but the test performance is poor in Table 2.1. Figure 2.3(c) plots the backpropagation runtime per gradient step per instance.

conditions. The Bellman error aims to accurately cover *all* the value functions, which works better on random trajectories; the policy gradient aims to maximize the expected cumulative reward along the *optimal policy only*, which works better with near-optimal trajectories that have better coverage in the optimal regions.

Computation cost Lastly, Figures 2.2(c) and 2.3(c) show the backpropagation runtime of the policy-gradient based optimality condition per gradient step per training instance across different Hessian approximations and different problem sizes in the gridworld and snare finding problems. To train the model, we run 100 epochs for every MDP in the training set, which immediately makes the full Hessian computation intractable as it would take more than a day to complete.

Analytically, let n be the dimensionality of the policy space and $k \ll n$ be the number of sampled trajectories used to approximate the derivatives. As shown in Section 2.7, the computation cost of full Hessian $O(n^2 + n^\omega)$ is quadratic in n and strictly dominates all the others. In contrast, the costs of the identity matrix approximation $O(nk)$ and the Woodbury approximation $O(nk + k^\omega)$ are both linear in n. The Woodbury method offers an option to get a more accurate Hessian at low additional cost.

2.10 Conclusion

This paper considers learning a predictive model to address the missing parameters in sequential decision problems. We successfully extend decision-focused learning from optimization problems to MDP problems solved by deep reinforcement learning algorithms, where we apply sampling and low-rank approximation to Hessian matrix computation to address the associated computational challenges. All our results suggest that decision-focused learning can outperform two-stage approach by directly optimizing the final evaluation metric. The idea of considering sequential decision problems as differentiable layers also suggests a different way to solve online reinforcement learning problems, which we reserve as a future direction.

3

Decision-Focused Learning in Restless Multi-Armed Bandits

3.1 Introduction

Restless multi-armed bandits (RMABs)[334,300] are composed of a set of heterogeneous arms and a planner who can pull multiple arms under budget constraint at each time step to collect rewards.

Different from the classic stochastic multi-armed bandits[121,54], the state of each arm in an RMAB can change even when the arm is not pulled, where each arm follows a Markovian process to transition between different states with transition probabilities dependent on arms and the pulling decision. Rewards are associated with different arm states, where the planner's goal is to plan a sequential pulling policy to maximize the total reward received from all arms. RMABs are commonly used to model sequential scheduling problems where limited resources must be strategically assigned to different tasks sequentially to maximize performance. Examples include machine maintenance[122], cognitive radio sensing problem[31], and healthcare[211].

In this paper, we study offline RMAB problems with unknown transition dynamics but with given arm features. The goal is to learn a mapping from arm features to transition dynamics, which can be used to infer the dynamics of unseen RMAB problems to plan accordingly. Prior works[211,294] often learn the transition dynamics from the historical pulling data by *maximizing the predictive accuracy*. However, RMAB performance is evaluated *by its solution quality* derived from the predicted transition dynamics, which leads to a mismatch in the training objective and the evaluation objective. Previously, decision-focused learning[338] has been proposed to directly optimize the solution quality rather than predictive accuracy, by integrating the one-shot optimization problem[80,247] or sequential problems[321,111] as a differentiable layer in the training pipeline. Unfortunately, while decision-focused learning can successfully optimize the evaluation objective, it is computationally extremely expensive due to the presence of the optimization problems in the training process. Specifically, for RMAB problems, the computation cost of decision-focused learning arises from the complexity of the sequential problems formulated as Markov decision processes (MDPs), which limits the applicability to RMAB problems due to the PSPACE hardness of finding the optimal solution[244].

Our main contribution is a novel and scalable approach for decision-focused learning in RMAB problems using Whittle index policy, a commonly used approximate solution in RMABs. Our

three key contributions are (i) we establish the differentiability of Whittle index policy to support decision-focused learning to directly optimize the RMAB solution quality; (ii) we show that our approach of differentiating through Whittle index policy improves the scalability of decision-focused learning in RMAB; (iii) we apply our algorithm to an anonymized maternal and child health RMAB dataset previously collected by ARMMAN[20] to evaluate the performance of our algorithm in simulation.

We establish the differentiability of Whittle index by showing that Whittle index can be expressed as a solution to a full-rank linear system reduced from Bellman equations with transition dynamics as entries, which allows us to compute the derivative of Whittle index with respect to transition dynamics. On the other hand, to execute Whittle index policy, the standard selection process of choosing arms with top-k Whittle indices to pull is non-differentiable. We relax this non-differentiable process by using a differentiable soft top-k selection to establish differentiability. Our differentiable Whittle index policy enables decision-focused learning in RMAB problems to backpropagate from final policy performance to the predictive model. We significantly improve the scalability of decision-focused learning, where the computation cost of our algorithm $O(NM^{\omega+1})$ scales linearly in the number of arms N and polynomially in the number of states M with $\omega \approx 2.373$, while previous work scales exponentially $O(M^{\omega N})$. This significant reduction in computation cost is crucial for extending decision-focused learning to RMAB problems with large number of arms.

In our experiments, we apply decision-focused learning to RMAB problems to optimize importance sampling-based evaluation on synthetic datasets as well as an anonymized RMAB dataset about a maternal and child health program previously collected by[20] – these datasets are the basis of comparing different methods in simulation. We compare decision-focused learning with the two-stage method that trains to minimize the predictive loss. The two-stage method achieves the best predictive loss but significantly degraded solution quality. In contrast, decision-focused learning reaches a slightly worse predictive loss but with a much better importance sampling-based solution

quality evaluation and the improvement generalizes to the simulation-based evaluation that is built from the data. Lastly, the scalability improvement is the crux of applying decision-focused learning to real-world RMAB problems: our algorithm can run decision-focused learning on the maternal and child health dataset with hundreds of arms, whereas state of the art is a 100-fold slower even with 20 arms and grows exponentially worse.

3.2 Related Work

Restless multi-armed bandits with given transition dynamics This line of research primarily focuses on solving RMAB problems to get a sequential policy. The complexity of solving RMAB problems optimally is known to be PSPACE hard[244]. One approximate solution is proposed by Whittle[337], where they use Lagrangian relaxation to decompose arms and compute the associated Whittle indices to define a policy. Specifically, the indexability condition[6,328] guarantees this Whittle index policy to be asymptotically optimal[334]. In practice, Whittle index policy usually provides a near-optimal solution to RMAB problems.

Restless multi-armed bandits with missing transition dynamics When the transition dynamics are unknown in RMAB problems but an interactive environment is available, prior works[300,203,241,78] consider this as an online learning problem that aims to maximize the expected reward. However, these approaches become infeasible when interacting with the environment is expensive, e.g., healthcare problems[211]. In this work, we consider the offline RMAB problem, and each arm comes with an arm feature that is correlated to the transition dynamics and can be learned from the past data.

Decision-focused learning The predict-then-optimize framework[91] is composed of a predictive problem that makes predictions on the parameters of the later optimization problem, and

an optimization problem that uses the predicted parameters to come up with a solution, where the overall objective is the solution quality of the proposed solution. Standard two-stage learning method solves the predictive and optimization problems separately, leading to a mismatch of the predictive loss and the evaluation metric [142,185,155]. In contrast, decision-focused learning [338,207,89] learns the predictive model to directly optimize the solution quality by integrating the optimization problem as a differentiable layer [11,3] in the training pipeline. Our offline RMAB problem is a predict-then-optimize problem, where we first (offline) learn a mapping from arm features to transition dynamics from the historical data [211,294], and the RMAB problem is solved using the predicted transition dynamics accordingly. Prior work [211] is limited to using two-stage learning to solve the offline RMAB problems. While decision-focused learning in sequential problems were primarily studied in the context of MDPs [321,111] they come with an expensive computation cost that immediately becomes infeasible in large RMAB problems.

3.3 Restless Multi-armed Bandit

An instance of the restless multi-armed bandit (RMAB) problem is composed of a set of N arms, each is modeled as an independent Markov decision process (MDP). The i-th arm in a RMAB problem is defined by a tuple $(\mathcal{S}, \mathcal{A}, R_i, P_i)$. $\mathcal{S}$ and $\mathcal{A}$ are the identical state and action spaces across all arms. $R_i, P_i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ are the reward and transition functions associated to arm i. We consider finite state space with $|\mathcal{S}| = M$ fully observable states and action set $\mathcal{A} = \{0, 1\}$ corresponding to not pulling or pulling the arm, respectively. For each arm i, the reward is denoted by $R_i(s_i, a_i, s_i') = R(s_i)$, i.e., the reward $R(s_i)$ only depends on the current state s_i, where $R : \mathcal{S} \to \mathbb{R}$ is a vector of size M. Given the state s_i and action a_i, $P_i(s_i, a_i) = [P_i(s_i, a_i, s_i')]_{s_i' \in \mathcal{S}}$ defines the probability distribution of transitioning to all possible next states $s_i' \in \mathcal{S}$.

In a RMAB problem, at each time step $t \in [T]$, the learner observes $\boldsymbol{s}_t = [s_{t,i}]_{i \in [N]} \in \mathcal{S}^N$, the

states of all arms. The learner then chooses action $\boldsymbol{a}_t = [a_{t,i}]_{i\in[N]} \in \mathcal{A}^N$ denoting the pulling actions of all arms, which has to satisfy a budget constraint $\sum_{i\in[N]} a_{t,i} \leq K$, i.e., the learner can pull at most K arms at each time step. Once the action is chosen, arms receive action $\boldsymbol{a}_t$ and transitions under P with rewards $\boldsymbol{r}_t = [r_{t,i}]_{i\in[N]}$ accordingly. We denote a full trajectory by $\tau = (\boldsymbol{s}_1, \boldsymbol{a}_1, \boldsymbol{r}_1, \cdots, \boldsymbol{s}_T, \boldsymbol{a}_T, \boldsymbol{r}_T)$. The total reward is defined by the summation of the discounted reward across T time steps and N arms, i.e., $\sum_{t=1}^{T} \gamma^{t-1} \sum_{i\in[N]} r_{t,i}$, where $0 < \gamma \leq 1$ is the discount factor.

A policy is denoted by π, where $\pi(\boldsymbol{a} \mid \boldsymbol{s})$ is the probability of choosing action $\boldsymbol{a}$ given state $\boldsymbol{s}$. Additionally, we define $\pi(a_i = 1 \mid \boldsymbol{s})$ to be the marginal probability of pulling arm i given state $\boldsymbol{s}$, where $\pi(\boldsymbol{s}) = [\pi(a_i = 1 \mid \boldsymbol{s})]_{i\in[N]}$ is a vector of arm pulling probabilities. Specifically, we use π^* to denote the optimal policy that optimizes the cumulative reward, while π^{solver} to denote a near-optimal policy solver.

3.4 Problem Statement

This paper studies the RMAB problem where we do not know the transition probabilities $P = \{P_i\}_{i\in[N]}$ in advance. Instead, we are given a set of features $\boldsymbol{x} = \{x_i \in \mathcal{X}\}_{i\in[N]}$, each corresponding to one arm. The goal is to learn a mapping $m_w : \mathcal{X} \to \mathcal{P}$, parameterized by weights w, to make predictions on the transition probabilities $P = m_w(\boldsymbol{x}) := \{m_w(x_i)\}_{i\in[N]}$. The predicted transition probabilities are later used to solve the RMAB problem to derive a policy $\pi = \pi^{\text{solver}}(m_w(\boldsymbol{x}))$. The performance of the model m is evaluated by the performance of the proposed policy π.

3.4.1 Training and Testing Datasets

To learn the mapping m_w, we are given a set of RMAB instances as training examples $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}, \mathcal{T})\}$, where each instance is composed of a RMAB problem with feature $\boldsymbol{x}$ that is correlated to the unknown transition probabilities P, and a set of realized trajectories $\mathcal{T} = \{\tau^{(j)}\}_{j\in J}$ generated

from a given behavior policy π_{beh} that determined how to pull arms in the past. The testing set $\mathcal{D}_{\text{test}}$ is defined similarly but hidden at training time.

3.4.2 Evaluation Metrics

Predictive loss To measure the correctness of transition probabilities $P = \{P_i\}_{i\in[N]}$, we define the predictive loss as the average negative log-likelihood of seeing the given trajectories $\mathcal{T}$, i.e., $\mathcal{L}(P, \mathcal{T}) := -\log \Pr(\mathcal{T} \mid P) = -\mathop{\mathbb{E}}_{\tau\sim\mathcal{T}} \sum_{t\in[T]} \log P(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})$. Therefore, we can define the predictive loss of a model m_w on dataset $\mathcal{D}$ by:

$$\mathop{\mathbb{E}}_{(\boldsymbol{x},\mathcal{T})\sim\mathcal{D}} \mathcal{L}(m_w(\boldsymbol{x}), \mathcal{T}) \tag{3.1}$$

Policy evaluation On the other hand, given transition probabilities P, we can solve the RMAB problem to derive a policy $\pi^{\text{solver}}(P)$. We can use the historical trajectories $\mathcal{T}$ to evaluate how good the policy performs, denoted by $\text{Eval}(\pi^{\text{solver}}(P), \mathcal{T})$. Given dataset $\mathcal{D}$, we can evaluate the predictive model m_w on dataset $\mathcal{D}$ by:

$$\mathop{\mathbb{E}}_{(\boldsymbol{x},\mathcal{T})\sim\mathcal{D}} \text{Eval}(\pi^{\text{solver}}(m_w(\boldsymbol{x})), \mathcal{T}) \tag{3.2}$$

Two common types of policy evaluation are importance sampling-based off-policy policy evaluation and simulation-based evaluation, which will be discussed in Section 3.6.

3.4.3 Learning Methods

Two-stage learning To learn the predictive model m_w, we can minimize Equation 3.1 by computing gradient $\frac{d\mathcal{L}(m_w(\boldsymbol{x}),\mathcal{T})}{dw}$ to run gradient descent. However, this training objective (Equation 3.1) differs from the evaluation objective (Equation 3.2), which often leads to suboptimal per-

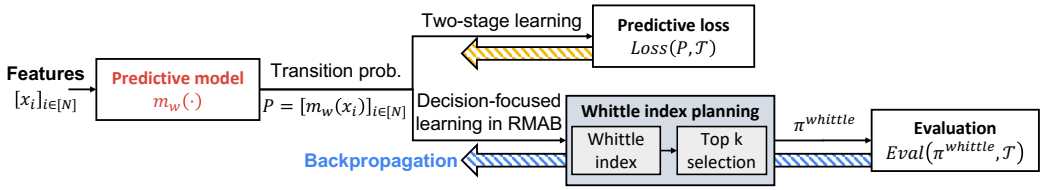


Figure 3.1: This flowchart visualizes different methods of learning the predictive model. Two-stage learning directly compares the predicted transition probabilities with the given data to define a predictive loss to run gradient descent. Decision-focused learning instead goes through a policy solver using Whittle index policy to estimate the final evaluation and run gradient ascent.

formance.

Decision-focused learning In contrast, we can directly run gradient ascent to maximize Equation 3.2 by computing the gradient $\frac{d\text{Eval}(\pi^{\text{solver}}(m_w(\boldsymbol{x})), \mathcal{T})}{dw}$. However, in order to compute the gradient, we need to differentiate through the policy solver π^{solver} and the corresponding optimal solution. Unfortunately, finding the optimal policy in RMABs is expensive and the policy is high-dimensional. Both of these challenges prevent us from computing the gradient to achieve decision-focused learning.

3.5 Decision-focused Learning in Restless Multi-armed Bandits

In this paper, instead of grappling with the optimal policy, we consider the Whittle index policy[337] – the dominant solution paradigm used to solve the RMAB problem. Whittle index policy is easier to compute and has been shown to perform well in practice. In this section we establish that it is also possible to backpropagate through the Whittle index policy. This differentiability of Whittle index policy allows us to run decision-focused learning to directly maximize the performance in the RMAB problem.

3.5.1 Whittle Index and Whittle Index Policy

Informally, the Whittle index of an arm captures the added value derived from pulling that arm. The key idea is to determine the Whittle indices of all arms and to pull the arms with the highest values of the index.

To evaluate the value of pulling an arm i, we consider the notion of 'passive subsidy', which is a hypothetical exogenous compensation β rewarded for not pulling the arm (i.e. for choosing action $a = 0$). Whittle index is defined as the smallest subsidy necessary to make pulling as rewarding as not pulling, assuming indexability[204]:

Definition 4 (Whittle index)**.** *Given state $u \in \mathcal{S}$, we define the Whittle index associated to state u by:*

$$W_i(u) := \inf_{\beta} \{ Q_i^{\beta}(u; a = 0) = Q_i^{\beta}(u; a = 1) \} \tag{3.3}$$

where the value functions are defined by the following Bellman equations, augmented with subsidy β for action $a = 0$.

$$V_i^{\beta}(s) = \max_a Q_i^{\beta}(s; a) \tag{3.4}$$

$$Q_i^{\beta}(s; a) = \beta \mathbf{1}_{a=0} + R(s) + \gamma \sum_{s'} \theta_i(s, a, s') V_i^{\beta}(s') \tag{3.5}$$

Given the Whittle indices of all arms and all states $W = [W_i(u)]_{i \in [N], u \in \mathcal{S}}$, the Whittle index policy is denoted by $\pi^{\text{whittle}} : \mathcal{S}^N \longrightarrow [0, 1]^N$, which takes the states of all arms as input to compute their Whittle indices and output the probabilities of pulling arms. This policy repeats for every time step to pull arms based on the index values.

3.5.2 Decision-focused Learning Using Whittle Index Policy

Instead of using the optimal policy π^* to run decision-focused learning with expensive computation cost, we use Whittle index policy π^{whittle} to determine how to pull arms as an approximate solution. In this case, in order to run decision-focused learning, we need to compute the derivative of the evaluation metric by chain rule:

$$\frac{d\text{Eval}(\pi^{\text{whittle}}, \mathcal{T})}{dw} = \frac{d\text{Eval}(\pi^{\text{whittle}}, \mathcal{T})}{d\pi^{\text{whittle}}} \frac{d\pi^{\text{whittle}}}{dW} \frac{dW}{dP} \frac{dP}{dw} \tag{3.6}$$

where W is the Whittle indices of all states under the predicted transition probabilities P. The policy π^{whittle} is the Whittle index policy induced by W. The flowchart is illustrated in Figure 3.1.

The term $\frac{d\text{Eval}(\pi^{\text{whittle}}, \mathcal{T})}{d\pi^{\text{whittle}}}$ can be computed via policy gradient theorem[296], and the term $\frac{dP}{dw}$ can be computed using auto-differentiation. However, there are still two challenges remaining: (i) how to differentiate through Whittle index policy to get $\frac{d\pi^{\text{whittle}}}{dW}$ (ii) how to differentiate through Whittle index computation to derive $\frac{dW}{dP}$.

3.5.3 Differentiability of Whittle Index Policy

A common choice of Whittle index policy is defined by:

Definition 5 (Strict Whittle index policy)**.**

$$\pi_W^{strict}(\boldsymbol{s}) = \mathbf{1}_{top\text{-}k([W_i(s_i)]_{i\in[N]})} \in \{0,1\}^N \tag{3.7}$$

which selects arms with the top-k Whittle indices to pull.

However, the strict top-k operation in the strict Whittle index policy is non-differentiable, which prevents us from computing a meaningful estimate of $\frac{d\pi^{\text{whittle}}}{dW}$ in Equation 3.6. We circumvent this

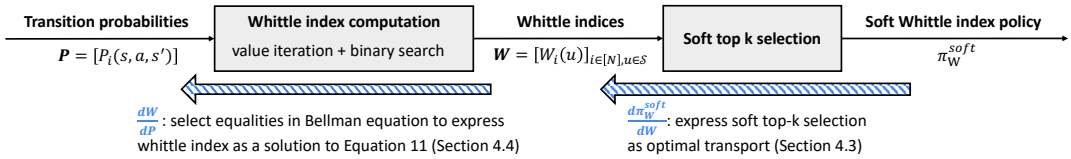


Figure 3.2: We establish the differentiability of Whittle index policy using a soft top-k selection to construct a soft Whittle index policy, and the differentiability of Whittle index by expressing Whittle index as a solution to a linear system in Equation 3.11.

issue by relaxing the top-k selection to a soft-top-k selection [346], which can be expressed as an optimal transport problem with regularization, making it differentiable. We apply soft-top-k to define a new differentiable soft Whittle index policy:

Definition 6 (Soft Whittle index policy)**.**

$$\pi_W^{soft}(\boldsymbol{s}) = \textit{soft-top-k}([W_j(s_i)]_{i \in [N]}) \in [0, 1]^N \tag{3.8}$$

Using the soft Whittle index policy, the policy becomes differentiable and we can compute $\frac{d\pi^{\text{whittle}}}{dW}$.

3.5.4 Differentiability of Whittle Index

The second challenge is the differentiability of Whittle index. Whittle indices are often computed using value iteration and binary search [258,210] or mixed integer linear program. However, these operations are not differentiable and we cannot compute the derivative $\frac{dW}{dP}$ in Equation 3.6 directly.

Main idea After computing the Whittle indices and the value functions of each arm i, the key idea is to construct linear equations that link the Whittle index with the transition matrix P_i. Specifically, we achieve this by resolving the max operator in Equation 3.4 of Definition 4 by determining the optimal actions a from the pre-computed value functions. Plugging back in Equation 3.5 and

manipulating as shown below yields linear equations in the Whittle index $W_i(u)$ and transition matrix P_i, which can be expressed as a full-rank linear system in P_i, with the Whittle index as a solution. This makes the Whittle index differentiable in P_i.

Selecting Bellman equation Let u and arm i be the target state and target arm to compute the Whittle index. Assume we have precomputed the Whittle index $\beta = W_i(u)$ for state u and the corresponding value functions $[V_i^\beta(s)]_{s\in\mathcal{S}}$ for all states under the same passive subsidy $\beta = W_i(u)$. Equation 3.5 can be combined with Equation 3.4 to get:

$$V_i^\beta(s) \geq \begin{cases} \beta + R(s) + \gamma \sum_{s'\in\mathcal{S}} \theta_i(s, a=0, s') V_i^\beta(s') \\ R(s) + \gamma \sum_{s'\in\mathcal{S}} \theta_i(s, a=1, s') V_i^\beta(s') \end{cases} \tag{3.9}$$

where $m = W_i(u)$.

For each $s \in S$, at least one of the equalities in Equation 3.9 holds because one of the actions must be optimal and match the state value function $V_i^\beta(s)$. We can identify which equality holds by simply plugging in values of precomputed value functions $[V_i^\beta(s)]_{s\in\mathcal{S}}$. Furthermore, for the target state u, both equalities must hold because by the definition of Whittle index, the passive subsidy $\beta = W_i(u)$ makes both actions equally optimal, i.e. in Equation 3.3, $V_i^\beta(u) = Q_i^\beta(u, a = 0) = Q_i^\beta(u, a = 1)$ for $\beta = W_i(u)$.

Thus Equation 3.9 can be written in matrix form:

$$\begin{bmatrix} \boldsymbol{V}_i^\beta \\ \boldsymbol{V}_i^\beta \end{bmatrix} \geq \begin{bmatrix} \mathbf{1}_M & \gamma \boldsymbol{P}_i(\mathcal{S}, a=0, \mathcal{S}) \\ \mathbf{0}_M & \gamma \boldsymbol{P}_i(\mathcal{S}, a=1, \mathcal{S}) \end{bmatrix} \begin{bmatrix} \beta \\ \boldsymbol{V}_i^\beta \end{bmatrix} + \begin{bmatrix} \boldsymbol{R}(S) \\ \boldsymbol{R}(S) \end{bmatrix} \tag{3.10}$$

where $\boldsymbol{V}_i^\beta := [V_i^\beta(s)]_{s\in\mathcal{S}}$, $\boldsymbol{R}(\mathcal{S}) = [R(s)]_{s\in\mathcal{S}}$, and $\boldsymbol{P}_i(\mathcal{S}, a, \mathcal{S}) := [P_i(s, a, s')]_{s,s'\in\mathcal{S}} \in \mathbb{R}^{M\times M}$.

By the aforementioned discussion, we know that there are at least $M + 1$ equalities in Equa-

tion 3.10 while there are also only $M+1$ variables ($m \in \mathbb{R}$ and $\boldsymbol{V}_i^\beta \in \mathbb{R}^M$). Therefore, we rearrange Equation 3.10 and pick only the rows where equalities hold to get:

$$A \begin{bmatrix} \mathbf{1}_M & \gamma \boldsymbol{P}_i(\mathcal{S}, a=0, \mathcal{S}) - I_M \\ \mathbf{0}_M & \gamma \boldsymbol{P}_i(\mathcal{S}, a=1, \mathcal{S}) - I_M \end{bmatrix} \begin{bmatrix} \beta \\ \boldsymbol{V}_i^\beta \end{bmatrix} = A \begin{bmatrix} -\boldsymbol{R}(S) \\ -\boldsymbol{R}(S) \end{bmatrix} \tag{3.11}$$

where we use a binary matrix $A \in \{0,1\}^{(M+1)\times 2M}$ with a single 1 per row to extract the equality. For example, we can set $A_{ij} = 1$ if the j-th row in Equation 3.10 corresponds to the equality in Equation 3.9 with the i-th state in the state space S for $i \in [M]$, and the last row $A_{(M+1),j} = 1$ to mark the additional equality matched by the Whittle index definition (see Appendix B.8 for more details). Matrix A picks $M+1$ equalities out from Equation 3.10 to form Equation 3.11.

Equation 3.11 is a full-rank linear system with $\beta = W_i(u)$ as a solution. This expresses $W_i(u)$ as an implicit function of $\boldsymbol{P}$, allowing for computation of $\frac{dW_i(u)}{d\boldsymbol{P}}$ via autodifferentiation, thus achieving differentiability of the Whittle index. We repeat this process for every arm $i \in [N]$ and every state u. Figure 3.2 summarizes the differentiable Whittle index policy and the algorithm is shown in Algorithm 2.

3.5.5 Computation Cost and Backpropagation

It is well studied that Whittle index policy can be computed more efficiently than solving the RMAB problem as a large MDP problem. Here, we show that the use of Whittle index policy also demonstrates a large speed up in terms of backpropagating the gradient in decision-focused learning.

In order to use Equation 3.11 to compute the gradient of Whittle indices, we need to invert the left-hand-side of Equation 3.11 with dimensionality $M+1$, which takes $O(M^\omega)$ where $\omega \approx 2.373$[9] is the best known matrix inversion constant. Therefore, the overall computation of all N arms and M states is $O(NM^{\omega+1})$ per gradient step.

In contrast, the standard decision-focused learning differentiates through the optimal policy using the full Bellman equation with $O(M^N)$ variables, where inverting the large Bellman equation requires $O(M^{\omega N})$ cost per gradient step. Thus, our algorithm significantly reduces the computation cost to a linear dependency on the number of arms N. This significantly improves the scalability of decision-focused learning.

3.5.6 Extension to Partially Observable RMAB

For partially observable RMAB problem, we focus on a subclass of RMAB problem known as collapsing bandits[210]. In collapsing bandits, belief states[219] are used to represent the posterior belief of the unobservable states. Specifically, for each arm i, we use $b_i \in \mathcal{B} = \Delta(\mathcal{S}) \subset [0,1]^M$ to denote the posterior belief of an arm, where each entry $b_i(s_i)$ denotes the probability that the true state is $s_i \in \mathcal{S}$. When arm i is pulled, the current true state $s_i \sim b_i$ is revealed and drawn from the posterior belief with expected reward $b_i^\top R$, where we can define the transition probability on the belief states. This process reduces partially observable states to fully observable belief states with in total MT states since the maximal horizon is T. Therefore, we can use the same technique to differentiate through Whittle indices of partially observable states.

3.6 Policy Evaluation Metrics

In this paper, we use two different variants of evaluation metric: importance sampling-based evaluation[296] and simulation-based (model-based) evaluation.

Importance sampling-based Evaluation We adopt Consistent Weighted Per-Decision Importance Sampling (CWPDIS)[304] as our importance sampling-based evaluation. Given target policy π and a trajectory $\tau = \{s_1, a_1, r_1, \cdots, s_T, a_T, r_T\}$ executed by the behavior policy π_{beh}, the

Algorithm 2: Decision-focused Learning in RMAB

Input: training set $\mathcal{D}_{\text{train}}$, learning rate r, model m_w
for *epoch* $= 1, 2, \cdots$ *and* $(x, \mathcal{T}) \in \mathcal{D}_{train}$ **do**
  Predict $P = m_w(x)$ and compute Whittle indices $W(P)$.
  Let $\pi^{\text{whittle}} = \pi_W^{\text{soft}}$ and compute $\text{Eval}(\pi^{\text{whittle}}, \mathcal{T})$.
  Update $w = w + r \frac{d\text{Eval}(\pi^{\text{whittle}}, \mathcal{T})}{d\pi^{\text{whittle}}} \frac{d\pi^{\text{whittle}}}{dW} \frac{dW}{dP} \frac{dP}{dw}$, where $\frac{dW}{dP}$ is computed from Equation 3.11.
Return: predictive model m_w

importance sampling weight is defined by $\rho_{ti} = \prod_{t'=1}^{t} \frac{\pi(a_{t',i}|s_{t'})}{\pi_{\text{beh}}(a_{t',i}|s_{t'})}$. We evaluate the policy π by:

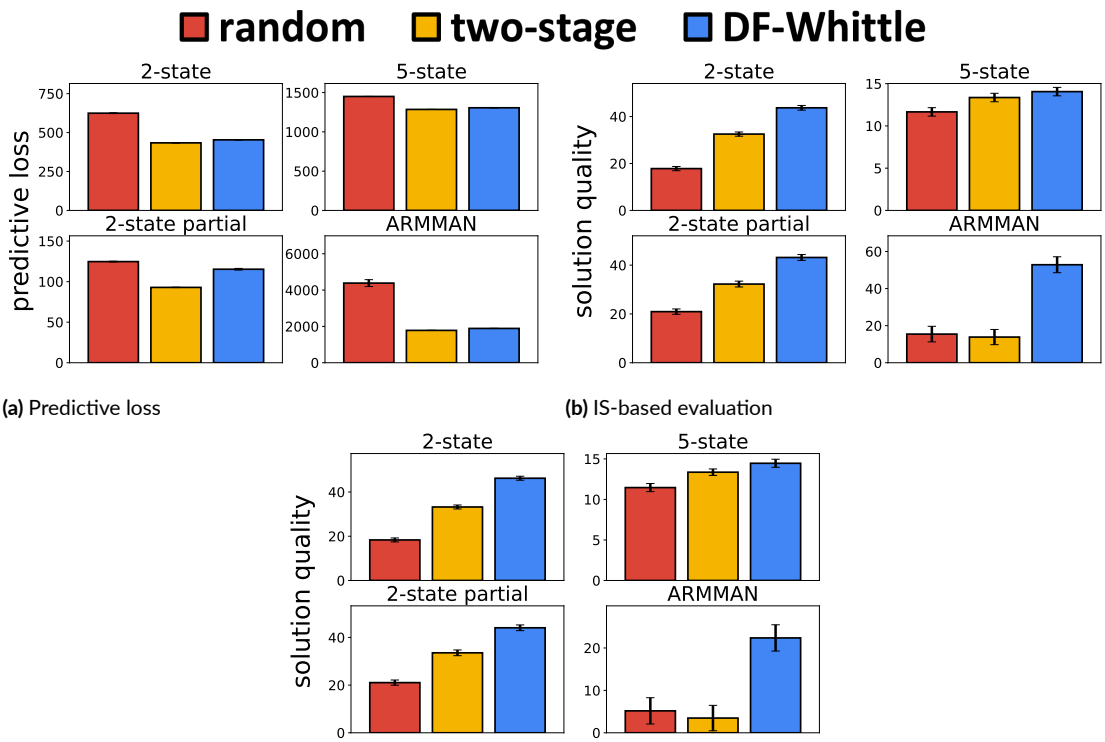
$$\text{Eval}_{\text{IS}}(\pi, \mathcal{T}) = \sum_{t \in [T], i \in [N]} \gamma^{t-1} \frac{\mathbb{E}_{\tau \sim \mathcal{T}} \left[r_{t,i} \rho_{ti}(\tau) \right]}{\mathbb{E}_{\tau \sim \mathcal{T}} \left[\rho_{ti}(\tau) \right]} \tag{3.12}$$

Importance sampling-based evaluations are often unbiased but with a larger variance due to the unstable importance sampling weights. CWPDIS normalizes the importance sampling weights to achieve a consistent estimate.

Simulation-based Evaluation An alternative way is to use the given trajectories to construct an empirical transition probability $\bar{P}$ to build a simulator and evaluate the target policy π. The variance of simulation-based evaluation is small, but it may require additional assumptions on the missing transition when the empirical transition $\bar{P}$ is not fully reconstructed.

3.7 Experiments

We compare two-stage learning (**TS**) with our decision-focused learning (**DF-Whittle**) that optimizes importance sampling-based evaluation directly. We consider three different evaluation metrics including predictive loss, importance sampling evaluation, and simulation-based evaluation to eval-



(a) Predictive loss

(b) IS-based evaluation

(c) Simulation-based evaluation

Figure 3.3: Comparison of predictive loss, importance sampling-based evaluation, and simulation-based evaluation on all synthetic domains and the real ARMMAN dataset. For the evaluation metrics, we plot the improvement against the no-action baseline that does not pull any arm. Although two-stage method achieves the smallest predictive loss, decision-focused learning consistently outperforms two-stage method in both *solution quality* evaluation metrics across all domains.

uate all learning methods. We perform experiments on three synthetic datasets including 2-state fully observable, 5-state fully observable, and 2-state partially observable RMAB problems. We also perform experiments on a real dataset on maternal and child health problem modelled as a 2-state fully observable RMAB problem with real features and historical trajectories. For each dataset, we use 70%, 10%, 20% of the RMAB problems as the training, validation, and testing sets, respectively. All experiments are averaged over 50 independent runs.

Synthetic datasets We consider RMAB problems composed of $N = 100$ arms, M states, budget $K = 20$, and time horizon $T = 10$ with a discount rate of $\gamma = 0.99$. The reward function is given by $R = [\frac{i-1}{M-1}]_{i \in [M]}$, while the transition probabilities are generated uniformly at random but with a constraint that pulling the arm ($a = 1$) is strictly better than not pulling the arm ($a = 0$) to ensure the benefit of pulling. To generate the arm features, we feed the transition probability of each arm to a randomly initialized neural network to generate fixed-length correlated features with size 16 per arm. The historical trajectories $\mathcal{T}$ with $|\mathcal{T}| = 10$ are produced by running a random behavior policy π_{beh}. The goal is to predict transition probabilities from the arm features and the training trajectories.

Real dataset The Maternal and Child Healthcare Mobile Health program operated by ARMMAN [20] aims to improve dissemination of health information to pregnant women and mothers with an aim to reduce maternal, neonatal and child mortality and morbidity. ARMMAN serves expectant/new mothers in disadvantaged communities with *median daily family income of $3.22 per day* which is seen to be below the world bank poverty line [34,42]. The program is composed of multiple enrolled beneficiaries and a planner who schedules service calls to improve the overall engagement of beneficiaries; engagement is measured in terms of total number of automated voice (health related) messages that the beneficiary engaged with. More precisely, this problem is modelled as a $M = 2$-state fully observable RMAB problem where each beneficiary's behavior is governed by an MDP with two states - Engaging and Non-Engaging state; engagement is determined by whether the beneficiary listens to an automated voice message (average length 115 seconds) for more than 30 seconds. The planner's task is to recommend a subset of beneficiaries every week to receive service calls from health workers to further improve their engagement behavior. We do not know the transition dynamics, but we are given beneficiaries' socio-demographic features to predict transition dynamics.

We use a subset of data from the large-scale anonymized quality improvement study performed

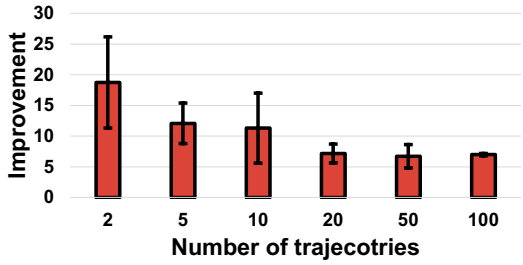


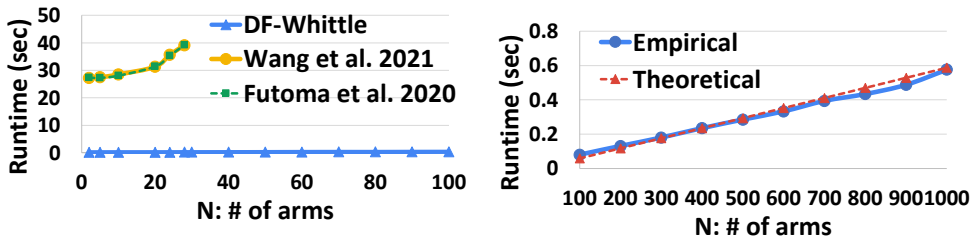
Figure 3.4: Performance improvement of decision-focused v.s. two-stage method with varying number of trajectories.

by ARMMAN for $T = 7$ weeks, obtained from Mate et al. [211], with beneficiary consent. In the study, a cohort of beneficiaries received Round-Robin policy, scheduling service calls in a fixed order, with a single trajectory $|\mathcal{T}| = 1$ per beneficiary that documents the calling decisions and the engagement behavior in the past. We randomly split the cohort into 8 training groups, 1 validation group, and 3 testing groups each with $N = 639$ beneficiaries and $K = 18$ budget formulated as an RMAB problem. The demographic features of beneficiaries are used to infer the missing transition dynamics.

Data usage All the datasets are anonymized. The experiments are secondary analysis using different evaluation metrics with approval from the ARMMAN ethics board. There is no actual deployment of the proposed algorithm at ARMMAN. For more details about the dataset, consent of data collection, please refer to Appendix B.2 and B.3.

3.8 Experimental Results

Performance improvement and justification of objective mismatch In Figure 3.3, we show the performance of random policy, two-stage, and decision-focused learning (DF-Whittle) on three evaluation metrics - predictive loss, importance sampling-based evaluation and simulation-based evaluation for all domains. For the evaluation metrics, we plot the improvement against the



(a) Comparing out algorithm to decision-focused baselines.

(b) Computation cost with varying number of arms N.

Figure 3.5: We compare the computation cost of our decision-focused learning with other baselines and the theoretical complexity $O(NM^{\omega+1})$ with varying number of arms N.

no-action baseline that does not pull any arms throughout the entire RMAB problem. We observe that two-stage learning consistently converges to a smaller predictive loss, while DF-Whittle outperforms two-stage on all solution quality evaluation metrics significantly (p-value < 0.05) by alleviating the objective mismatch issue. This result also provides evidence of aforementioned objective mismatch, where the advantage of two-stage in the predictive loss does not translate to solution quality.

Significance in maternal and child care domain In the ARMMAN data in Figure 3.3, we assume limited resources that we can only select 18 out of 638 beneficiaries to make service call per week. Both random and two-stage method lead to around 15 more (IS-based evaluation) listening to automated voice messages among all beneficiaries throughout the 7-week program by $18 \times 7 = 126$ service calls, when compared to not scheduling any service call; this low improvement also reflects the hardness of maximizing the effectiveness of service calls. In contrast, decision-focused learning achieves an increase of beneficiaries listening to 50 more voice messages overall; DF-whittle achieves a much higher increase by strategically assigning the limited service calls using the right objective in the learning method. The improvement is statistically significant (p-value $<$ 0.05).

In the testing set, we examine the difference between those selected for service call in two-stage and DF-Whittle. We observe that there are some interesting differences. For example, DF-Whittle chooses to do service calls to expectant mothers earlier in gestational age (22% vs 37%), and to a lower proportion of those who have already given birth (2.8% vs 13%) compared to two-stage. In terms of the income level, there is no statistic significance between two-stage and DFL (p-value = 0.20 see Appendix B.2). In particular, 94% of the mothers selected by both methods are below the poverty line[342].

Impact of Limited Data Figure 3.4 shows the improvement between decision-focused learning and two-stage method with varying number of trajectories given to evaluate the impact of limited data. We notice that a larger improvement between decision-focused and two-stage learning is observed when fewer trajectories are available. We hypothesize that less samples implies larger predictive error and more discrepancy between the loss metric and the evaluation metric.

Computation cost comparison Figure 3.5(a), compares the computation cost per gradient step of our Whittle index-based decision-focused learning and other baselines in decision-focused learning[321,111] by changing N (the number of arms) in $M = 2$-state RMAB problem. The other baselines fail to run with $N = 30$ arms and do not scale to larger problems like maternal and child care with more than 600 people enrolled, while our approach is 100x faster than the baselines as shown in Figure 3.5(a) and with a linear dependency on the number of arms N.

In Figure 3.5(b), we compare the empirical computation cost of our algorithm with the theoretical computation complexity $O(NM^{\omega+1})$ in N arms and M states RMAB problems. The empirical computation cost matches with the linear trend in N. Our computation cost significantly improves the computation cost $O(M^{\omega N})$ of previous work as discussed in Section 3.5.5.

3.9 Conclusion

This paper presents the first decision-focused learning in RMAB problems that is scalable for large real-world datasets. We establish the differentiability of Whittle index policy in RMAB by providing new method to differentiate through Whittle index and using soft-top-k to relax the arm selection process. Our algorithm significantly improves the performance and scalability of decision-focused learning, and is scalable to real-world RMAB problem sizes.

4

Decision-focused Learning in Maternal and Child Health*

4.1 Introduction

Non-profits often leverage the extensive cell phone coverage to feasibly reach underserved communities for information dissemination programs. In particular, NGOs working in the mobile health

States Covered in India	19
Partner NGOs	40
Partner Hospitals	97
Health Workers Trained	235K
Beneficiaries	27.2M

Scale of ARMMAN

Figure 4.1: Beneficiary receiving preventive health information

space can deliver timely and targeted health information via text or voice messages[250,166]. Unfortunately, such programs suffer from a dwindling engagement over time, with large number of beneficiaries dropping out from the program. NGOs can make use of health workers to personally reach out to beneficiaries through service calls, encourage their participation and address complaints. However, health workers' availability and time are scarce resources; only a limited number of beneficiaries can be given a service call every week. It is thus crucial to optimize which beneficiaries receive these personal service calls. We pose this as optimization problem of constrained sequential resource allocation solved using Restless Multi-Armed Bandits (RMAB). Each beneficiary is modelled as an arm following a Markov Decision Process and the action of whether to place a service call or not results in state change. The Whittle index heuristic[337] is the dominant approach for solving RMABs. However, for computing Whittle Indices, transition dynamics of each arm must be known. While many previous works make the assumption that transition dynamics parameters are already known, in the real world, these parameters must be inferred. When arm features are correlated with transition dynamics, historical data on arm pulls is leveraged to learn a mapping from arm features to transition dynamics[211,294]. The learnt mapping function is then used to predict the unknown parameters for new arms and solve the subsequent optimization problem.

*For completion of the decision-focused learning part, this chapter presents the field study result of the method proposed in Chapter 3. The work was collaborated with and primarily led by Shresth Verma.

This approach thus falls under the Predict-then-Optimize[90,87,89] framework, where an optimization problem is to be solved but the parameters defining the optimization problem are unknown. This is a *two-stage approach*: The first stage is to learn a predictive model which maps from some environment features to the parameters. Subsequently, in the second stage, the optimization problem formulated using the predicted parameters is solved. However, there is a key shortcoming in this two-stage framework. While the mapping function maximizes for the predictive accuracy of parameters, we are interested in the solution quality of the optimization problem parameterized by the predicted parameters. Decision-Focused Learning (DFL)[80,338,207,323] is proposed to address this mismatch between the training objective and the evaluation objective by embedding the optimization problem within the training pipeline. However, until now, Decision Focused Learning has only been studied through simulated experiments.

In this paper, we present the first work showcasing the real-world impact of DFL for RMABs through a large scale field study. For conducting the field study, we collaborate with ARMMAN, an NGO in India working in mobile health space for maternal and child health awareness (Figure 4.1). In prior works, a RMAB model using the previously mentioned two-stage learning approach has been used for optimizing live service call scheduling in the field[211]. We compare this two-stage approach with a DFL approach in optimizing service calls. Engagement is a key metric that captures beneficiaries' participation in the mobile program. Our results show that allocating health worker resources using a DFL policy reduces drop in engagement by 31% as compared to the no-service call baseline. On the other hand, the benefit from TS policy is not statistically significant. We also show that live service calls made by health care workers using DFL policy have higher effectiveness than TS policy resulting in better short-term as well as long-term outcomes in listenership behaviour.

Furthermore, we perform detailed post-hoc analysis of the real-world study and back the observations using simulated experiments to explain how DFL is making decisions and why those decisions result in a better performance. Our novel contributions are as follows:

- We show results from the first large-scale field study of Decision Focused Learning being applied to maternal and child health domain.
- We show that by optimizing for decision quality rather than predictive accuracy, DFL results in statistically significant improvement in final decision quality measured through engagement metric in the mobile health program.
- We provide an interpretation of how DFL strategically learns to distinguish between arms that benefit most from interventions, resulting in improved parameter predictions compared to the TS model.

Our positive results thus pave the way for future works applying Decision Focused Learning in real world agent-modelling tasks as well as optimization problems with unknown underlying problem parameters. We shall release the code for experiments upon acceptance.

4.2 Related Work

The optimization problem of constrained sequential resource allocation can be solved using Restless Multi-Armed Bandits (RMAB). RMABs have been used in real world applications such as anti-poaching patrol planning[258], healthcare interventions[211,210], and machine repair and maintenance[123]. The complexity of optimally solving RMAB problems is known to be PSPACE hard[244]. Whittle Index approach[337] is an approximate solution to RMAB problem which is aymptotically optimal under the indexability condition [334,6,328]. However, for computing the Whittle Index, transitions dynamics must be known. Under unknown system dynamics,[211,294] leverage the predict-then-optimize framework for learning a predictive model of transition dynamics from features using historical data.

The predict-then-optimization[91] framework (or two-stage learning) solves for an optimization problem with unknown parameters by learning a predictive model of parameters from environ-

ment features and subsequently solving the optimization problem. However, this two-stage process separates out the prediction and optimization problems, thereby causing a mismatch between the predictive loss that is minimized and the evaluation metric that is desired to be maximized[142,185,155]. Decision Focused Learning[338,207,89], solves this problem by embedding optimization problem as a differentiable layer in a deep learning pipeline. Most previous DFL[80,247,89,207] approaches solve one-shot optimization problems such as stochastic programming and security games in an end-to-end manner. Recently,[321,111] propose an extension of Decision Focused Learning for sequential decision making problems. Decision Focused Learning has been applied in directly optimizing game utilities in Network Security Games[320] and Stackelberg Security Games[248]. [323] further extend the Decision Focused learning methodology for Restless Multi Armed Bandit problems for generalized N-state MDP as well as a belief state MDP to optimze for decision quality. However, none of these works, either in the single shot setting or the sequential decision making settings, have ever been tested in the real world in the field; and hence were unable to thoroughly analyze comparative advantages of decision focused learning over baseline approaches with real world data.

4.3 Mobile Health Adherence

4.3.1 Mobile Health Program

ARMMAN is a non-governmental organization in India focused on reducing maternal and neonatal mortality among underpriviledged communities. The NGO operates a mobile health service that disseminates preventive health information to expectant or new mothers (beneficiaries) on a weekly basis via automated voice messages. A large fraction ($\sim$ 90%) of mothers in the program are below the World Bank international poverty line[342] and the program has so far served over a million mothers. However, despite the success of the program, beneficiaries' engagement with the voice calls dwindles over time with 22% of beneficiaries dropping out of the program within just 3 months

of enrolment. Live Service calls made by health workers can encourage beneficiaires' participation. However, the health workers' availability is limited and thus, only a fixed number of live service calls can be made every week. This constraint necessitates a smart scheduling strategy of which beneficiaries to reach out every week to best utilize health workers' efforts.

4.3.2 Restless Multi-Armed Bandits

We consider the Restless Multi-Armed Bandit model with N independent arms each characterized by a 2-action Markov Decision Process (MDP) Figure 4.2. Each MDP is defined using the tuple $\{\mathcal{S}, \mathcal{A}, R, \mathcal{P}\}$ where $\mathcal{S}$ refers to the state space, $\mathcal{A}$ is the action space, which in our case is discrete and binary, $\mathcal{A} \in \{0, 1\}$. R is the reward function such that $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$. $\mathcal{P}$ is the transition function, such that $\mathcal{P}(s, a, s')$, $(s, s') \in \mathcal{S}, a \in \mathcal{A}$ represents the probability of transitioning from state s to s' under action a. The policy function $\pi : \mathcal{S} \mapsto \mathcal{A}$ is defined as the mapping from states to action.

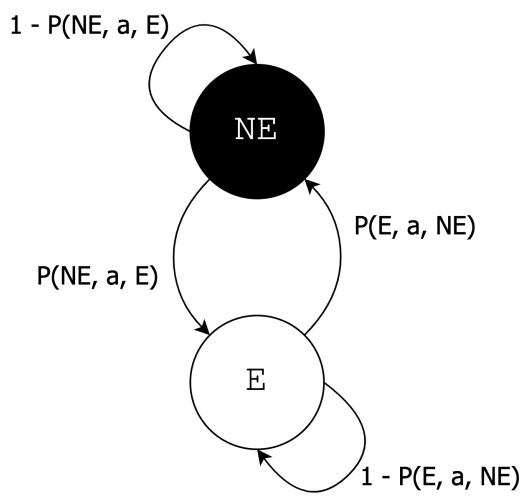


Figure 4.2: The beneficiary transitions from a current state s to a next state s' under action α, with probability $P(s, \alpha, s')$.

In our problem setup, we consider a 2-state 2-action MDP problem. Based on our discussions

with the NGO, states are defined using the engagement metric. If a beneficiary listens to at least 1 call for more than 30 seconds in a week, they are said to be in Engaging state ($s = 1$). Otherwise, the beneficiary is in Non-Engaging state ($s = 0$). The timestep of the MDP is chosen to be a period of 1 week. The actions correspond to whether to deliver (active) or not deliver (passive) live service call to a beneficiary. Additionally, the NGO can only deliver K live service calls in a week. The reward function at any given timestep is defined to be same as the current state $R(s, a) = s$. The planner's goal is then to maximize expected long term reward (engagement). Starting from a state s_0, this is defined using the value function V as :

$$V(s_0) = \mathbb{E}_{s_{t+1} \sim P} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t), s_{t+1} | \pi, s_0) \right] \tag{4.1}$$

where γ is the discount factor for rewards.

The *Whittle Index* for every arm is defined using the 'passive subsidy'. The passive subsidy is the additional reward accrued by an arm when the passive action is chosen. The whittle index is then defined as the passive subsidy such that expected future value is identical for both the passive and active actions. Formally, the whittle index WI_i for an arm i in state s can be defined as:

$$W_i(s) = \inf_m \{ V_i^m(s; a = 0) = V_i^m(s; a = 1) \} \tag{4.2}$$

where V_i^m is subsidized value function under passive subsidy m.

Intuitively, the Whittle index measures the value of pulling an arm conditioned on the observed state. Therefore, at every timestep, the *Whittle Index Policy* ranks all arms by their current state whittle index. The top-K arms with the highest whittle indices are chosen for active action to maximize the total pulling performance.

4.3.3 Missing Transition Probabilities in RMAB

Most works using RMABs make the assumption that MDP parameters are known beforehand. However, in practice, we may not have access to beneficiaries' transition probabilities to define the RMAB model. In our problem, the mobile health program receives new sets of beneficiaries without information about their transition behavior. This prevents us from applying techniques in RMAB to properly schedule service calls.

Learning challenge The solution we adopt here is to learn a mapping from the beneficiaries' demographic features and prior interaction with the program to forecast the transition probabilities. Similar to Predict-then-Optimize framework[90] we learn a predictive model and then determine the live service call schedule using the RMAB model.

Dataset We use the historical beneficiaries' listenership behaviour between January 2022 to May 2022 as the training dataset. Specifically, we have access to state trajectories of 19944 (N) beneficiaries over a period of 5 weeks (T), along with the action chosen for every beneficiary at every timestep. Note that passive actions make up majority of the historical data with only 3% of transitions happening under an active action. In addition to the trajectories, we have socio-demographic features for every beneficiary obtained at registration time. These features cover information such as age, gestational age, income, education, parity, gravidity, language of automated call, and registration channel.

4.4 Comparison of Learning Methods

In this section, we summarize the Two-Stage and the Decision-Focused learning approaches for obtaining the transition probability parameters of beneficiaries. Crucially, the TS approach maximizes

for the predictive accuracy while the DFL approaches maximizes the decision objective.

4.4.1 Two-stage Learning

In [211], TS model is shown to cut $\sim 28\%$ engagement drops as compared to a Round-Robin baseline. In our work, we consider outperforming the TS baseline to show applicability of DFL model. Thus we follow similar setup of the TS model as described in [211]. A mapping function f is learnt that predicts the Transition Probabilities given the socio-demographic features x_i for the i_{th} arm. Predicted Transition Probabilities for arm P_i can then be obtained as $P_i = f(x^i), i \in [N]$. Since our problem domain consists of two states and two actions, we have to predict four transition probabilities. We model the mapping function as a neural network f_w parameterized by the weights w. f_w is designed using two fully connected layers followed by four outputs and finally logistic function is applied to obtain probabilities. f_w is learnt by minimizing the negative log-liklihood of observed trajectories $\mathcal{T}$ under the predicted transition probabilities $f_w(x)$. The loss function $\mathcal{L}$ is thus given by

$$\mathcal{L}(f_w(x), \mathcal{T}) = \mathop{\mathbb{E}}_{i \in [N]} -\log(\mathcal{T}^i | f_w(x^i)) \tag{4.3}$$

The weights w of the neural network f_w are optimized by backpropogating the gradient $\frac{d\mathcal{L}(f_w(x),\mathcal{T})}{dw}$.

4.4.2 Decision-focused Learning

We replicate the Decision Focused learning pipeline from [323] where instead of optimizing for predictive accuracy, the final decision outcome is optimized. Off-Policy Policy Evaluation (OPE) is used to quantify the decision outcome. It measures the reward obtained from a learnt policy given the past trajectories from a different policy. The DFL architecture uses the same predictive model f_w as TS, described in the previous section. However, once Transition Probabilities are predicted as $P = f_w(x)$, we compute Whittle Indices using a differentiable function W. The whittle indices

$WI = W(P)$ parameterize a differentiable policy which we denote as π^{WI}. Finally, the differentiable evaluation objective is formulated using OPE of learnt policy under the observed trajectories $\mathcal{T}$ which is represented as $OPE(\pi^{WI}, \mathcal{T})$. The weights of the predictive model are learnt by maximizing the final objective and backpropogating through the complete pipeline. The gradient is thus given by $\frac{d\ \mathrm{OPE}(\pi^{WI},\mathcal{T})}{dw}$.

In Decision Focused Learning, we calculate this gradient by using the chain rule:

$$\frac{d\ \mathrm{OPE}(\pi^{WI}, \mathcal{T})}{dw} = \frac{d\ \mathrm{OPE}(\pi^{WI}, \mathcal{T})}{d\pi^{WI}} \frac{d\pi^{WI}}{dWI} \frac{dWI}{dP} \frac{dP}{dw} \tag{4.4}$$

We refer the reader to the appendix for more details on DFL pipeline.

4.5 Field Study

We collaborated with the NGO on the maternal and child health problem and conducted a service quality improvement field study to compare the performance of different learning approaches. All experiments reported in this paper are approved by an ethics review board at the NGO.

Hypothesis and research question: The main goal in this paper is to understand the performance of decision-focused learning in real-world problems. Decision-focused learning has shown better performance in many applications but only in simulation. There is no deployment or real-world evidences of whether decision-focused learning actually outperforms other learning methods in practice.

Control methods In earlier work[211], the two-stage approach was shown to outperform a benchmark of Round Robin Policy. The work also provides statistical significance results, illustrating the superiority of two-stage RMAB policy over non-AI baseline. Therefore outperforming

the two-stage approach is important to show the utility of decision-focused learning. In our field study, we compare the following live service call scheduling strategies: (1) Current Standard of care (CSOC), where no live service calls are delivered to the beneficiaries, (ii) Two-stage (TS) approach where beneficiaries are chosen for live service calls according to the Whittle Index Policy learnt using Two-Stage learning, and (iii) Decision-Focused Learning (DFL) approach where beneficiaries are chosen for live service calls according to the Whittle Index Policy learnt using Decision Focused learning. We use the performance of the CSOC group to anchor the performance of other AI-based methods. The performance of the CSOC group also measures the baseline engagement rate that the mobile health program receives without any intervention. Therefore, we focus on the improvement of AI-based methods against the CSOC method.

Eligibility criterion and randomization We consider the group of beneficiaries registered between the months of April 2022 to June 2022. Further, we filter out beneficiaries who have not listened to even a single automated voice call in the time period of 30 days before the study begins. This filtering is done to remove beneficiaries from the cohort who have long term connectivity issues such as phone number out of service or misentry of phone number at enrolment. Lastly, we randomly sample 9000 beneficiaries out of these eligible candidates to form our study cohort. We split these set of beneficiaries into three groups of 3000 beneficiares each - (i) CSOC group, (ii) TS group, and (iii) DFL group. We make sure that the distribution of socio-demographic features and start-state are the same across the three groups.

Experiment design Beneficiaries become eligible for live service calls 2 months post their enrolment into the program. Within the TS and DFL groups, we choose $K = 300$ beneficiaries for live service call every week based on NGO's constraints. These live service calls are sent out weekly for a period of 6 weeks. We continue to monitor the cohort for 4 more weeks even after the study

ends to measure the sustained effect of live service calls. It should be noted that, automated voice messages are sent to all groups throughout this period and only the delivery of live service calls by health workers changes across the three groups.

4.6 Experiment Results

In this section, we showcase the results from the field study. We also define multiple evaluation metrics and demonstrate how the different policies fare against each other.

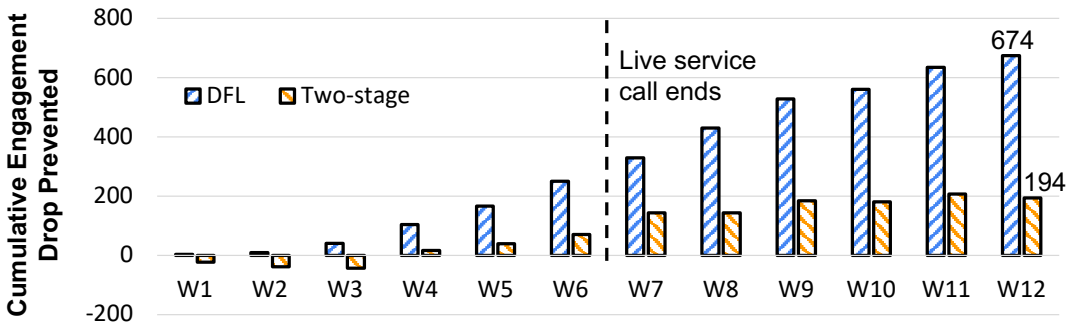


Figure 4.3: Weekly Cumulative Engagement Drop Prevented for the DFL and TS groups. Live service calls are only delivered for the first 6 weeks, after which, all three groups are only passively observed. The DFL group prevents more Cumulative Engagement Drops as compared to the TS group

4.6.1 Weekly and Cumulative Engagement

We first present the results from our study using the Engagement Metrics proposed by Mate et al. [211]. Engagement at time t for the ith beneficiary, represented by $E^i(t)$, is defined as 1 if the beneficiary listens to at least one automated call in a week for more than 30 seconds and 0 otherwise. Since the engagement of beneficiaries dwindles over time, we can measure the drop in engagement relative to

the engagement at start. The engagement drop and the cumulative engagement drop are defined as

$$E^i_{drop}(t) := E^i(0) - E^i(t);\ E^i_{cumu\_drop}(t) := \sum_{\zeta=0}^{\zeta=t} E^i_{drop}(\zeta)\,. \tag{4.5}$$

The cumulative engagement drop prevented over the CSOC group is simply the difference in cumulative engagement drop of the policy and the CSOC group. Figure 4.3 shows the cumulative engagement drops prevented over CSOC group for DFL and TS policies. We see that DFL prevented more drops across all weeks and by the end of study, DFL group has **560** more engagement drops prevented over the CSOC group as compared to TS group which only prevents **181** engagement drops. Given a total of 1765 cumulative engagement drops in the CSOC group, DFL group has 31% fewer cumulative engagement drops as compared to CSOC group while TS only results in 10% reduction in cumulative engagement drops.

4.6.2 Statistical Significance

We also establish statistical significance † of DFL's benefit using regression analysis [18]. We fit a linear regression model to predict the output variable $E^i_{cumu\_drop}$ by giving beneficiary features x_i as an input vector of length J along with and an indicator variable T_i denoting whether a beneficiary belongs to DFL ($T_i = 1$) or CSOC ($T_i = 0$) group. The regression model can thus be represented as

$$Y_i = k + \beta T_i + \sum_{j=1}^{J} \gamma_j x_{i,j} + \varepsilon_i \tag{4.6}$$

where β is the regression coefficient of the indicator variable T_i measuring the effect of treatment, γ_j is the regression coefficient of the j-th input feature, k is the constant term of regression and ε_i is the error. Y_i is the target variable that is fitted using the regression model and is same as $E^i_{cumu\_drop}$. The

† See Appendix for erratum

regression coefficient for T is found to be 0.19 with p-value of 0.024. On the other hand, similar comparison between TS ($T_i = 1$) vs CSOC ($T_i = 0$) yields a regression coefficient of 0.06 for T with p-value of 0.48. Thus, belonging to the DFL group resulted in significantly positive impact on cumulative engagement drops while for TS, no such statistical significance could be established.

Table 4.1: Statistical significance for service call impact tested using a linear regression model

	DFL vs CSOC	**TS vs CSOC**
% reduction in cumulative engagement drops	31%	10%
p-value	0.024	0.48
Coefficient β	0.19	0.06

4.6.3 Performance on Listenership Metrics

While the whittle index policy maximimizes the reward, which is defined using the engagement metric, we also measure if the policy improved other metrics characterizing listenership. Thus, we define metrics quantifying listenership behaviour of a beneficiary within a time window of 14 days before and after receiving a service call.

Definitions

1. Mean Duration: The mean duration of calls listened to within the time window.
2. No. of Engagements: The numbers of calls engaged with (30+ seconds listened) within the time window.
3. Engagements to Scheduled (E/S) Ratio: The ratio of numbers of calls engaged with to numbers of calls scheduled within the time window.

Results We calculate the change in these metrics between the time window before and after a live service call. Table 4.2 reports the mean change in these metrics across the three experimental

groups. We observe that across all the metrics, **DFL group has a significantly higher change in listenership behaviour through live service calls as compared to the TS group**. For instance, we can interpret the value of 17.054 in Mean Duration metric for DFL as active actions in DFL group resulting in beneficiaries listening to on average 17 seconds more of an automated call. This is in contrast to TS group, where live service calls only resulted in beneficiaries listening to 6 seconds more of an automated call. Note that the average duration of an automated message is 60 seconds. Thus a 17 seconds improvement in listenership corresponds to an average 28% increase in message content listened to among those treated with live service calls. Using t-test for comparison of means, we find that for each of the proposed metrics, mean change is statistically higher for DFL group as compared to TS group with p-value< 0.05.

Table 4.2: Performance of the DFL and TS policies across multiple listenership metrics. DFL policy shows a higher change in listenership behaviour from a service call as compared to the TS policy.

Policy	Change in Mean Duration	Change in No. of Engagements	Change in E/S
DFL	17.054	0.094	0.20
TS	6.764	0.009	0.07

Table 4.3: Multiple Error and Rank metrics evaluated for DFL and TS policies. While TS group shows a lower overall error in predicting transition probabilities, DFL group has lower predictive error in Top-K arms and a higher rank correlation with the optimal ranking.

	Rank Metrics		**Transition Probability Error Metrics**			
Policy	Precision @ K	Spearman's Correlation	MAE All	MAE Top-K	Mean NLL All	Mean NLL Top-K
DFL	0.41	**0.30**	0.31	**0.35**	0.79	**0.62**
TS	0.22	0.179	**0.25**	0.37	**0.42**	0.69

4.7 Understanding Decision-focused Learning

4.7.1 Learnings from Real World Experiment

The Decision Focused Learning method consists of an end-to-end pipeline starting from features to predicted Transition Probabilities to computed whittle index and finally the decision of whether a

beneficiary is in top-K list chosen for live service call. In this section, we interpret the DFL's strategy in contrast with the Two-Stage policy by performing post-hoc analysis across all these steps.

As a first step for this analysis, we compute the ground truth transition probabilities using the observed trajectories of beneficiaries during the time period of field study. Once Ground Truth Transition Probabilities are estimated, we subsequently compute the Ground Truth Whittle Index and Ground Truth top-K ranks.

Top-K Rank Lists We consider the ordered list of beneficiaries according to predicted whittle index in the Two-Stage and DFL experiment groups. Additionally, True Top-K rank list is also computed using the ground truth whittle index. To measure the agreement between the two lists, we use the following metrics:

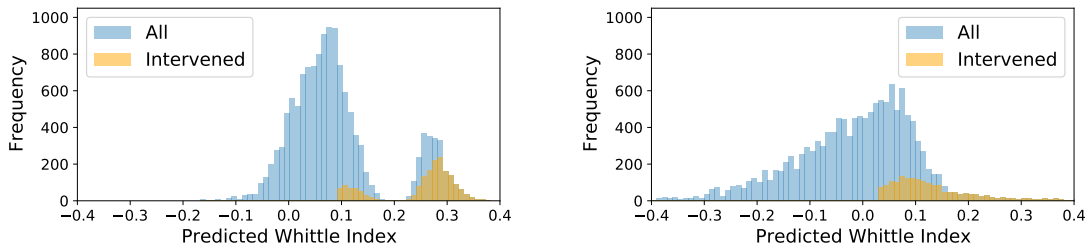
1. Precision @ K: This metric counts the proportion of relevant beneficiaries in the top-K positions of the policy rank list and is widely used in classification [356,255] and ranking problems [298]. The precision @ K in our problem is given by:

$$\text{Precision @ K} = \frac{|\text{Policy Top-K list} \cap \text{True Top-K list}|}{K}$$

2. Spearman's Rank Correlation: This metric calculates the rank correlation between the Predicted Whittle Index and Ground Truth Whittle Index of Policy's Top-K ranked beneficiaries.

In Table 4.3, we show the different rank metrics for the two comparison groups. In all the weeks, we find that **the DFL group has a higher agreement with the True Top-K ranks as compared to the Two-Stage experiment group.**

Whittle Indices For beneficiaries belonging to each of the experimental group, we have the corresponding computed Whittle Index from predicted Transition Probabilities. We call it the Predicted Whittle Index (note that these values are not directly predicted by the Neural Network models). Figure 4.4 shows the distribution of Predicted Whittle Index for DFL and TS experiment groups in Blue. We also mark the beneficiaries who are chosen for Active action within each experimental group in orange.



(a) Decision Focused Learning **(b)** Two Stage

Figure 4.4: Predicted Whittle index distribution and beneficiaries intervened for TS and DFL groups across all weeks. The DFL group has a bimodal distribution of predicted whittle index as compared to unimodal distribution in the TS group. Note that the right peak in DFL is not fully covered due to beneficiaries changing states over the course of study.

A striking observation is that **within the DFL group, the whittle indices have a bimodal distribution as opposed to a unimodal distribution for Two-Stage group**. This suggests that in DFL, the model is trying to learn a decision boundary between the beneficiaries to deliver active and passive action. This contrasts with the Two-Stage model where objective is solely to learn accurate transition probabilities.

Predicted Transition Probabilities Given the ground truth and predicted transition probabilities for both DFL and TS policies, we compute for the whole population (i) the Mean Negative Log Likelihood (NLL) of observed trajectories under predicted transition probabilities and (ii) the prediction error using Mean Absolute Error (MAE). In Table 4.3, we show that DFL has both higher MAE and higher Mean NLL as compared to TS. Thus DFL model is poorer in predicting

the transition probabilities. However, if we compute these metrics for just the true top-K beneficiaries (MAE Top-K and Mean NLL Top-K), we find that DFL has lower MAE as well as Mean NLL than TS. This suggests that the **DFL focuses on correctly predicting the transition probabilities for beneficiaries who will actually be picked, in contrast to the TS, which optimizes for predictive performance for the whole population**. It must be noted, that the predictive performance of TS is impacted due to limited historical data around active actions (limited service calls made by the NGO).

4.7.2 Short-term and Long-term Impact of Live Service Calls

In Figure 4.5, we plot mean reward accrued by beneficiaries in the next step after an active action for both Two-Stage and DFL group. This quantifies the short term impact of a live service call. In both the NE and E state, we observe that DFL leads to higher reward.

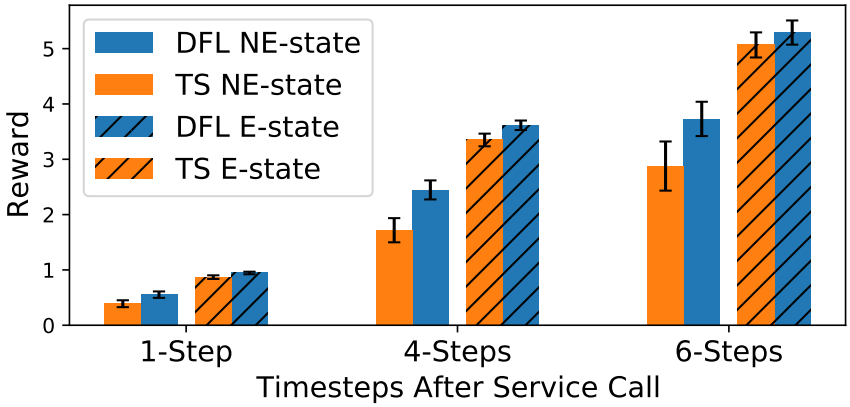


Figure 4.5: Mean reward accrued by beneficiaries in short term (1-step lookahead reward) and long term (4-steps and 6-steps lookahead rewards) after given an active action, DFL group has higher reward in both the short-term and the long term as compared to the TS group.

While short-term impact is only applicable for one timestep ahead, the Whittle Index policy optimizes for long-term rewards. In Figure 4.5, we also plot the reward obtained in 4 weeks and 6 weeks

following the live service call. We show this for both TS and DFL group. Again, we see that DFL's live service calls are more effective than TS policy even in the long term.

4.7.3 Who Benefits from DFL

In order to determine which beneficiaries benefited the most from the DFL policy, we first obtain the ratio of calls engaged with over total scheduled calls (E/S) for every beneficiary over the whole duration of study. Subsequently, we rank the beneficiaries based on the E/S ratio and compute average E/S ratio for different percentiles. We calculate these numbers for all three policies. In Figure 4.6, we plot the lift in E/S ratio over CSOC for different percentiles. While DFL shows a positive lift in listenership over CSOC across all percentiles, the maximum lift is achieved in the lowest percentiles. This shows that those with low listenership are the ones benefiting most from the DFL policy.

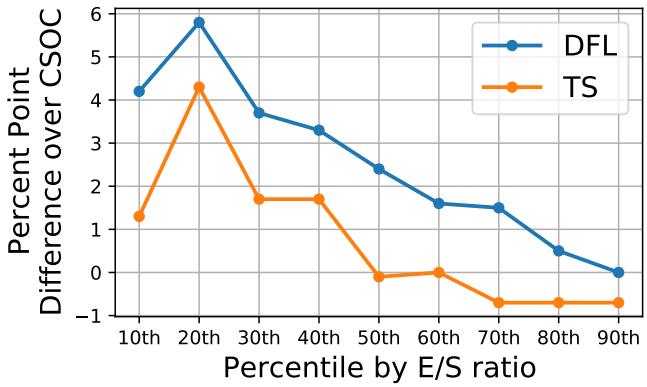


Figure 4.6: Lift in E/S ratio over CSOC for different percentiles. The highest lift in E/S ratio is in the lowest percentile suggesting that beneficiaries with poor listenership of automated voice messages benefited the most from live service calls.

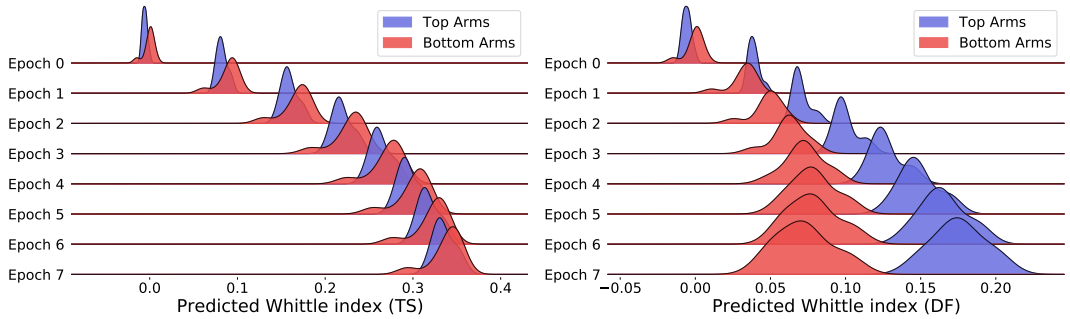
4.7.4 Learnings from Simulated Experiments

In this section, we conduct simulated experiments to improve our understanding of the DFL model and verify the observations made from the real world experiment. Specifically, we consider an RMAB system with 100 arms simulating beneficiaries enrolled in the NGO's program. The MDP parameters of each arm are randomly initialized. Additionally, we obtain a feature vector corresponding to every arm such that the features are correlated with the MDP parameters. Lastly, we simulated multiple trajectories for the whole system and store that as offline dataset for our experiments. All experimental results are reported by averaging over five seed values.

The Effect of Training Data Size While Decision Focused Learning optimizes for the decision objective, a TS model that perfectly predicts the optimization problem parameters should also achieve the optimal decision objective. However, in the real world, predictive models do make errors. These errors can be dependent on the quantity of training data that is available to the learning model.

We thus formulate the hypothesis that the gain from DFL model should be higher in limited data scenario. As size of training data grows, DFL and TS should converge to similar decision objective. To test this hypothesis, we run a simulated experiment with varying number of trajectories per arm. Figure 4.8(a) shows lift in Off-Policy Policy Evaluation from DFL over TS with increasing training data size. We observe that the highest lift is with smallest training data size and as we increase availability of training data, lift diminishes.

Shift in Whittle Index Distribution over Training Epochs As DFL learns to optimize the decision objective directly, we hypothesise that it should learn a model which effectively separates the top ranked and bottom ranked whittle index arms. On the other hand, since TS optimizes for predictive accuracy, it has no incentive to learn an optimal ranking of the arms by whittle



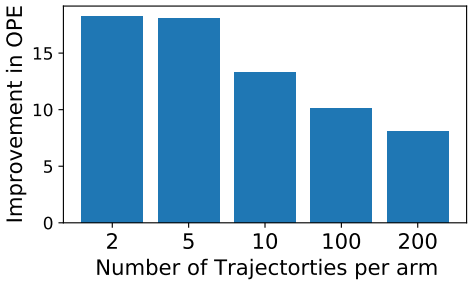
(a) Two-Stage Learning **(b)** Decision Focused Learning

Figure 4.7: Predicted whittle index distribution for optimal top and bottom arms, across the training epochs. DFL policy learns whittle indices such that the true top ranked and bottom ranked arms are segregated. TS policy fails to learn whittle indices following this strategy.

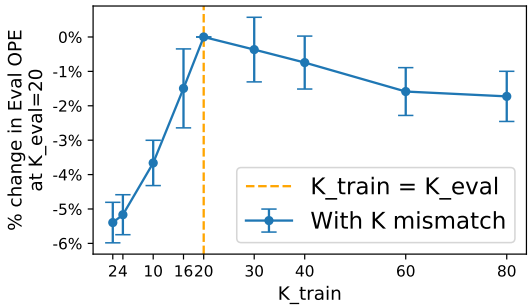
index. To verify this hypothesis, we plot the predicted whittle index distributions of true top-K and bottom-K arms. In Figure 4.7, we visualize how these distributions change over the training epochs, giving a glimpse into the learning process of the two models. We observe that both the TS and DFL model start with no prior information of the true top-K and bottom-K arms. However, over the training epochs, DFL learns whittle indices such that it separate the two groups. The Two-Stage model fails to learn such segregation in predicted whittle index distribution.

The Effect of Budget-K The budget constraint in the RMAB problem defines the number of arms chosen for active action every week. In a two-stage model, the learning step outputs the transition probabilities irrespective of the budget value K. However, in Decision Focused Learning, the mapping model which outputs the transition probabilities maximizes for the decision objective that relies on the value of K. To simulate the effect of mismatch in K, we train DFL model with changing K at train time (*K_train*), while keeping the K fixed at the time of evaluation (*K_eval*). Specifically, we note the OPE at evaluation time with $K = 20$ for different training scenarios with $K \in [2, 4, 10, 16, 20, 30, 40, 60, 80]$ as shown in Figure 4.8(b). We observe that the performance at evaluation time only drops slightly (by upto 6%) in both the cases of train time budget being

greater or lesser than the evaluation time budget. The sensitivity of the DFL's performance to the value of K_train supports the hypothesis that DFL learns a decision boundary optimized for the exact number of beneficiaries chosen for active action. Further, keeping $K\_eval = K\_train$ can help maximize the performance of DFL.



(a) Improvement in evaluation OPE of DFL over TS with changing number of trajectories per arm.



(b) Percentage drop in evaluation OPE with budget as 20 (K_eval) relative to maximum Eval OPE, across changing budget at train time (K_train).

Figure 4.8: Off-policy policy evaluation (OPE) of decision-focused learning and two-stage learning with varying number of trajectories and budget at train time.

4.8 Conclusion

Several applications at AAMAS first require learning a predictive model of agents' parameters and then optimizing based on result of such learning. This paper presents key results on importance of Decision Focused Learning to such applications. We conduct the first large-scale field study of Decision Focused learning through an RMAB problem in maternal and child health domain. We conclude that learning the MDP parameters of the RMAB problem through Decision Focused Learning results in higher participation of beneficiaries in the program (Figure 4.3). DFL's strategic selection of actions also results in more effective live service calls as demonstrated in Table 4.2. From the analysis showcased in previous sections, we attribute the success of DFL to the following: (i) The predicted whittle index distribution from DFL policy is bimodal in contrast to a unimodal

distribution in TS (see Figure 4.4) indicating that DFL model learns a decision boundary to highly rank beneficiaries that would benefit significantly more from receiving the service call than the rest of the population. (ii) DFL is more aligned with the optimal policy as shown by a higher rank correlation with the True Top-K Beneficiaries as compared to TS (Table 4.3). (iii) While TS results in a lower predictive error for the population as a whole, DFL by optimizing for decision quality results in improved transition probability prediction for the top-K beneficiaries (Table 4.3).

4.9 Ethics and data usage

Acknowledging the responsibility associated with real-world AI systems for undeserved communities, we have closely coordinated with interdisciplinary team of ARMMAN's field staff, social work researchers, public health researchers and ethical experts through all major steps of model iteration, development and experimentation. Particularly, prior to all experiments, approval was obtained from ethics review board at both ARMMAN and Google.

Consent and Data Usage

The consent for participating in the program is received from beneficiaries. All the data collected through the program is owned by the NGO and only the NGO is allowed to share data. This dataset will never be used by Google for any commercial purposes. We only use anonymized data and no personally identifiable information (PII) is made available to the AI models. The data exchange and use was thus regulated through clearly defined exchange protocols including anonymization, read-access only to researchers, restricted use of the data for research purposes only, and approval by ARMMAN's ethics review committee.

Universal Accessibility of Health Information

Our service call scheduling model focuses on improving quality of service calls and does not alter, for any beneficiary, the accessibility of health information. All participants will receive the same weekly health information by automated message regardless of whether they are scheduled to receive service calls or not. The service call program does not withhold any information from the participants nor conduct any experimentation on the health information. The health information is always available to all participants, and participants can always request service calls via a free missed call service.

5

Decision-focused Learning in Network Intervention

5.1 Introduction

Many real-world security problems present the challenge of how to allocate limited resources to large number of important targets, including infrastructure [112], transportation systems [240], urban

crime[357], and web security[308]. *Stackelberg security games* (SSGs) are frequently used to study the interaction between defender and attacker and optimally allocate the security resources accordingly. *Network security games (NSGs)*[332,101,272], a natural extension of SSGs, describe a strategic adversarial interaction between an attacker and a defender on a graph. The attacker's goal is to take a path from a starting location to a target without being caught by the defender. The defender declares (i.e., attacker surveils) a mixed strategy of how she will deploy her security resources to the edges of the network. NSGs are relevant in many real-world settings such as wildlife conservation[95,213], infrastructure protection[147], and nuclear material smuggling[243,225].

One key challenge in applying NSGs in the real world is learning an adversary's behavior from historical data. Past works[21,227,1] in security games have shown that constructing bounded rationality adversary models from data greatly improves performance of deployed models because attackers often behave quite differently from how rational models would suggest. A particular motivation for this paper is wildlife smuggling[98,267,361], a natural NSG domain where large amounts of historical attack data is available in the form of past seizures.

Almost all previous work on security games approaches the problem of adversary modeling by first building a full adversary model that aims to predict the adversary behavior as accurately as possible[2,75,95,236]. In early work, the judgments of human experts were used to estimate the adversary's preferences[297]. Later, in domains where historical attack data was available, machine learning was used to construct models instead (starting from Letchford et al.[190]). In NSGs, building an adversary model to maximize accuracy has several key limitations. First, the model is selected without any consideration of the impact of errors downstream. Prediction errors on paths that are frequently taken by the adversary have a large impact on defender utility, but are weighted the same as errors on paths that are rarely taken. Secondly, standard adversary models require human feature engineering to apply to NSGs due to a great variety of paths from the attacker's starting location to each potential target[105,350,125,126]. Once the adversary model is determined, the following defender utility maxi-

mization problem can be solved by any optimization techniques, including bilevel optimization [213], branch and cut [102], and double oracle [147].

Our approach represents a fundamental shift: we take an end-to-end, game-focused approach, focusing on learning a model that yields a high defender utility. More specifically, we take the downstream defender utility maximizatoin problem into account while learning the adversary model. To that end, we use a graph convolutional neural network architecture to learn the adversary's behavior, which allows us to overcome both of the issues of prior work. First, assuming we can differentiate through the defender's optimization problem, we can train the entire model end-to-end because the predictive model is differentiable, i.e., to take the optimization problem into account while training. Second, the graph convolutional network automatically extracts features from the graph, meaning that hand engineering is not necessary. Nevertheless, several challenges must be overcome to implement this approach: principally, poor scalability of naive end-to-end training and non-convexity of the game-focused objective.

A summary of our contributions is as follows: first, we construct a *graph convolution*-based adversary model for NSGs. This model is fully differentiable, does not require manual selection of path features, and transmits target value information over the network. Second, we develop a randomized block update scheme for differentiating through optimization problems, whose computation time is usually more than quadratic in terms of the number of variables due to the computation of Hessian matrix and matrix inversion. Such computational issue is especially influential for optimization problems with a huge number of variables, which is commonly seen in NSGs as every edge corresponds to one individual decision variable. In these cases, randomized block update can largely reduce the time complexity. We further provide an approximation guarantee relative to the complete derivatives, and we show empirically that our approach greatly improves scalability. We also show that through judicious use of the standard predictive loss as regularization, we can escape local minima in the end-to-end loss function.

5.2 Related Work

There is a rich literature on learning adversary behavior models in Stackelberg security games (SSGs) (starting from Letchford et al.[190]), but learning in NSGs has received much less attention. While SSGs generalize NSGs, the scalability concerns are quite different because reducing NSGs to SSGs may create exponentially many targets—one for each path to the target in the NSG. Thus, applying standard attacker bounded rationality models, such as quantal response (QR)[214,222] and subjective utility quantal response (SUQR)[236] is nontrivial. Yang et al.[350] and Ford et al.[105] reduced NSGs to SSGs by considering each individual path as an attacker pure strategy. Their approach scales poorly, creating exponentially many paths in many networks. It also relies on hand-crafting path features that capture adversary behavior well. Other authors have developed models that use Markovian dynamics to model the attacker. Gutfraind et al.[125] and Abbasi et al.[2] assume the attacker does not receive any information beyond the neighboring nodes—attackers do not make any decisions that are more long term than a single timestep. Gutfraind et al.[126] takes the opposite approach: attackers follow a path that minimizes some cost (such as the risk of being caught) with randomness in the individual decisions. This adds some global information, but requires the model designer to specify the choice of cost function in advance.

Past work in adversary modeling in SSGs has viewed the problem of constructing an adversary model and solving the defender's optimization as completely separate problems and does not consider the impact of errors in the defender model on the quality of the optimization outcome, with a few exceptions. Sinha et al.[285] and Haghtalab et al.[129] relate the predictive accuracy of the learned model to the defender's expected utility. In the case of Haghtalab et al., this view motivates the use of a non-standard loss function to achieve better utility. However, even these papers take a fundamentally two-stage approach: the model is trained independently of any information about the game itself, such as the defender's utilities. Perrault et al.[248] takes a game-focused approach to SSGs,

but the issues that arise in NSGs are different and require a greater focus on scalability.

A major challenge in our work is differentiating through the nonconvex defender optimization problem. Recent work has developed general approaches for differentiating convex problems[4]. Perrault et al.[248] present an approach for a limited class of nonconvex problems. Our setting is challenging in two ways. First, we have a decision variable for each edge in the network and these approaches scale poorly (more than quadratically) in the number of variables. Second, our setting is more severely nonconvex than that of Perrault et al.

5.3 Background

Stackelberg Security Games A *Stackelberg security game (SSG)*[354,297] is a two-player sequential game. The defender aims to protect a set of targets T with limited budget b which can only protect up to b targets. Each target $t \in T$ is associated with a defender penalty $U^d(t) \leq 0$ and an attacker reward $U^a(t) \geq 0$ when the target is successfully attacked. For simplicity, we assume there is no reward and penalty when the attacker is caught or fails to reach the target. Once the defender commits to her mixed strategy, the attacker can conduct surveillance to observe the defender's mixed strategy and choose one target to attack accordingly. We denote the defender's mixed strategy by $z \in \mathbb{R}^{|T|}$, where $0 \leq z_t \leq 1$ denotes the marginal probability that target t is protected. The budget constraint can be written as $1^\top z \leq b$. On the attacker side, we use $\theta(z, x)$ to represent the attacker's behavior, where $\theta_t(z, x)$ (or θ_t if there is no ambiguity) is the probability of attacking target t, and x is the available features revealed to both the defender and the attacker, e.g., the attacker payoff value $U^a(t)\ \forall t \in T$ can be considered as a feature. Notice that θ is a function of the defender strategy z and the feature x, which implies that the attacker can be reactive to the defender strategy and select

the target based on the underlying feature. We can write the defender's utility function as:

$$\mathrm{DefU}(z;\theta) = \sum_{t \in T} \theta_t(z,x) U^d(t)(1 - z_t). \tag{5.1}$$

This includes the case where the attacker is fully rational, where $\theta_t(z,x) = 1$ if $t = \arg\max_{t' \in T}(1 - z_{t'})U^a(t')$ else 0.

Bounded Rationality in SSGs *Quantal response (QR)*[214] models the attacker's behavior by setting the probability that each target is attacked to be proportional to the exponential of its payoff scaled by a constant. *Subjective utility quantal response (SUQR)*[236], which fits data better than QR in practice, sets the probability proportional to the exponential of a subjective utility or an attractiveness function of the attacker:

$$\theta_t(z,x) \propto \exp(-\omega z_t + \Phi(t,x)), \tag{5.2}$$

where $\omega > 0$ is a constant representing the attacker's risk aversion and $\Phi(t,x)$ denotes the subjective utility of target t given feature x.

Network Security Games *Network security games (NSGs)*[101,232] are SSGs played on a graph structure. Given an undirected (or directed) graph $G = (V,E)$, the defender allocates a limited number of checkpoints along edges in E, while the attacker tries to find a path from a source to a target without being caught. We divide the set of all vertices V into targets $T = \{t_1, t_2, ..., t_{|T|}\}$ and non-targets $S = \{s_1, s_2, ..., s_{|S|}\}$ (or potential sources). At each time, the attacker appears in one potential source $s \in S$ drawn from a given prior distribution $\pi \in \mathbb{R}^{|S|}$. From the defender's perspective, the defender strategy $z_e \, \forall e \in E$ is the marginal probability of covering edge e. Similarly, the defender has a limited number of resources b to protect the targets.

We use $\alpha = \{v_1, v_2, ..., v_{|\alpha|}\}$ to denote a path which starts from a source $v_1 \in S$ and ends with a target $v_{|\alpha|} \in T$. We use $\mathcal{A}$ to denote the set of all possible paths from any source to any target, which could be exponentially many or infinitely many when the graph contains any cycle. Similar to SSGs, let $U^d(t)$ be the defender's payoff when the target t is attacked successfully and U^d_{caught} be the defender's payoff when the attacker is caught. Let $U^d = \{U^d(t_1), ..., U^d(t_{|T|}), U^d_{\text{caught}}\} \in \mathbb{R}^{|T|+1}$ denote the defender's payoff vector. In addition, we assume each node $v \in \mathcal{V}$ has a node feature vector $x_v \in \mathbb{R}^D$ consisting of characteristics of node v, e.g., the attacker payoff of the current node $U^a(v)$ if $v \in T$. We use $x \in \mathbb{R}^{|V| \times D}$ to denote all the node features in graph G.

Bounded Rationality in NSGs In this paper, we assume the attacker to be boundedly rational, where the attacker's behavior is characterized by a function $\theta(z, x)$, where $\theta_\alpha(z, x)$ represents the probability of choosing path α under coverage z and feature x. Given the coverage z, we can compute the defender expected utility:

$$\text{DefU}(z; \theta) = \sum_{\alpha \in \mathcal{A}} \theta_\alpha(z, x) U^d(\alpha) \prod_{e \in \alpha} (1 - z_e), \tag{5.3}$$

where $U^d(\alpha) = U^d(t)$ is the defender utility when the attacker successfully passes through α to attack its target t.

The difference between Equation 5.1 and 5.3 is that there are multiple layers of protection along the path α. Therefore the probability of successfully attacking a target is the product of all the success probabilities of crossing each edge e in the path. The defender's optimization problem is generally hard. For example, if the function $\theta(z, x)$ is given by full rationality restricted to only polynomial many paths $\mathcal{A}$, the defender optimization problem is NP-hard [147]. Furthermore, the set of all possible paths $\mathcal{A}$ could be exponentially large or infinitely many when there is any cycle.

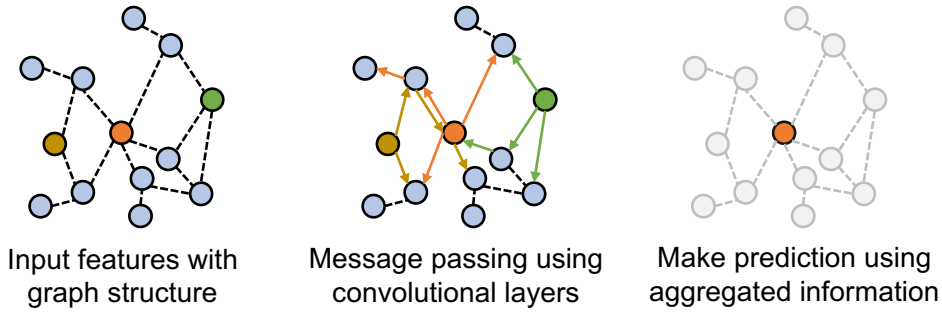


Figure 5.1: The convolutional layers of GCNs can propagate and aggregate information in a non-linear fashion. In NSGs, such message passing ability corresponds to the attacker's ability of conducting surveillance to neighbor nodes.

Graph Convolutional Networks There has been much recent attention paid to *graph convolutional networks (GCNs)*[223,172,130]. Given a graph, the convolutional layers in GCNs can transmit information through message passing, which allows information to propagate to distant nodes and be aggregated in a non-linear fashion. GCNs are much more expressive than hand-crafted features. In this paper, we apply GCNs, parameterized by w, to map each node $v \in V$ and the entire node features x with graph structure to a scalar $\Phi(v, x; w)$, which represents the extent that the attacker is "pulled" toward that node. The message passing in GCNs is similar to the information gathering conducted by the adversary, where a rough understanding of faraway targets is available to the adversary.

5.4 Adversary Model

Our attacker model is Markovian—the probability of using a path α can be decomposed into the product of transition probabilities:

$$\theta_\alpha(z, x) = \prod_{e \in \alpha} \theta_e(z, x). \tag{5.4}$$

Motivated by the SUQR model, we propose a **local SUQR** model, which assumes the probability that the attacker moves from u to v using edge $e = (u, v)$ is proportional to $exp(-\omega z_{u\to v} - \eta \mathbf{y}_v + \Phi(v, x; w))\ \forall v \in N_{\text{out}}(u)$. $\Phi(v, x; w)$ represents the subjective utility or attractiveness of node v parameterized by w, which can be learned by GCN. The variable $\mathbf{y}_v$, with a weight $\eta \geq 0$, represents the downstream future risk or coverage perceived by the attacker at node v. In other words, the attacker tends to move toward the target with higher attractiveness $\Phi(v, x; w)$, but avoids using the edge $e = (u, v) \in E$ with higher coverage $z_{u\to v}$ and avoids moving towards nodes v with higher future risk $\mathbf{y}_v$.

Given a defender coverage strategy, there are many heuristic ways to obtain a measure of future risk. For example, we can follow the above Markovian behavior without the effect of the future risk, where the probability of being caught can be analytically computed efficiently. Another heuristic is the shortest distance to any target, as suggested by Gutfraind et al. [126]. The only restriction put on the choice of the future risk is differentiability.

We can compute the transition probability from u to any $v \in N_{\text{out}}(u)$ as:

$$\theta_{u\to v}(z, x; w) = \frac{exp(-\omega z_{u\to v} - \eta \mathbf{y}_v + \Phi(v, x; w))}{\sum_{v' \in N_{\text{out}}(u)} exp(-\omega z_{u\to v'} - \eta \mathbf{y}_{v'} + \Phi(v', x; w)}. \tag{5.5}$$

Unlike previous boundedly rational models [350,105], we do not need to enumerate all the feasible paths, which could be exponentially large. Unlike the nonreactive Markovian model [125], our model is reactive to the defender's strategy. Unlike Gutfraind et al. [126], we are not limited to noisily following a shortest path.

In local SUQR, the path structure is automatically encoded in the reactive Markovian behavior. Since the edge coverage effect is involved in the transition probability, the probability of taking a path is also exponentially proportional to the total coverage along the path, which is also included in other bounded rational models [350,105]. The flexibility and the generalizability of the attractiveness

function allow us to apply any graph learning algorithms to extract the adversary behavior. Compared to previous hyperparameters tuning models, our model is more expressive and can adapt to a broader range of adversary behavior.

5.5 Problem Statement

For each instance, a directed graph $G = (V, E)$ with node features x is presented to both the defender and the attacker. The attacker has a hidden rationality function θ^*, which is a function of node features x and the defender coverage z. The defender first chooses a coverage $\{z_e\}_{e \in E}$ under the budget constraint $1^\top z \leq b$. The attacker observes z and then behaves based on his own rationality function θ^*. We assume that the defender has access to historical play between the defender and the attacker, which can be used to form an estimate of the adversary behavior. The goal of the defender is to maximize the received expected reward.

5.6 Two-stage Learning for Network Security Games

The main comparison of the remainder of the paper is between our GCN-based adversary model implemented as two-stage vs. our game-focused methods. Thus, we briefly describe the two-stage approach that we consider.

Predictive Model A two-stage approach fits the GCN-based attractiveness function $\Phi(v, x)$ for all $v \in V$ to minimize the difference between predicted behavior θ given by Equation 5.5 and the corresponding true attacker behavior θ^*. Given the attacker behavior θ^* and a prediction θ, we can define the loss by either matrix norm or the KL-divergence of the path distribution inferred by two behaviors under previous coverage z and features x. These losses are generally infeasible to compute since there are infinite many possible paths. In practice, however, we often have paths sampled

from the true behavior θ^* we can use to approximately compute the KL-divergence between two behaviors. Given the choice of loss function $\mathcal{L}$, we can train a model θ by minimizing the average loss:

$$\mathbb{E}_{(z,x,\theta^*)\in D}\mathcal{L}(\theta^*, \theta; z, x) \tag{5.6}$$

Prescriptive Model Given a graph G, node features x, and predicted attacker behavior q, the defender's goal is to choose an optimal coverage z^* satisfied the budget constraint to maximize her own objective value.

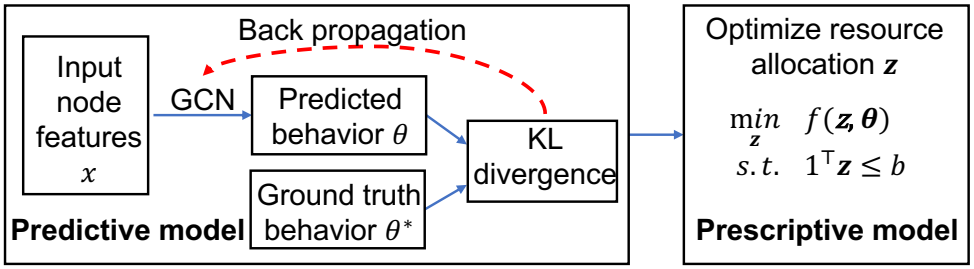
When the defender strategy z is chosen, the attacker follows his own Markovian behavior $\theta(z, x)$. But due to the allocated coverage, the attacker will be caught with probability z_e when he passes through edge e. This can be cast as an absorbing Markov chain, where the probability of crossing an edge e is $\theta_e(z, x)(1 - z_e)$, and the rest of the probability the attacker will be caught and turned into a dummy caught state v_{caught}. We also assume that once the attacker reaches either any terminal or caught state v_{caught}, the attacker cannot go back to any other states, i.e., these are absorbing states. Therefore, given a coverage z, we can model the attacker's behavior as an absorbing Markov chain. We can analytically compute the corresponding defender utility. To align with the standard minimization formulation, we denote the *negative* defender utility by $f(z, \theta)$. For ease of notation, we omit the presence of node features. The optimization problem is given by:

$$\begin{aligned} \min_z \quad & f(z, \theta) \\ \text{s.t.} \quad & 1^\top z \leq b, \qquad 0 \leq z_e \leq 1 \quad \forall e \in \mathcal{E} \end{aligned} \tag{5.7}$$

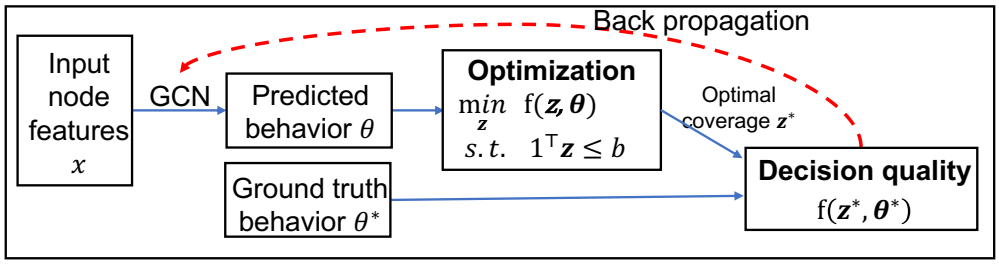
Unfortunately, the function f is neither convex nor submodular when the attacker is reactive. The standard approach is to apply constrained black-box optimization solvers to solve the problem, e.g.,

Sequential Least SQuares Programming (SLSQP) [177,44].

5.7 Naive Game-Focused Learning for Network Security Games



(a) Two-stage method



(b) game-focused method

Figure 5.2: Two-stage method trains the behavior model by minimizing the predictive loss, while the game-focused method trains the behavior model by optimizing the final decision quality.

In general, a good predictive model does not necessarily imply a high defender utility in the second stage. Sometimes a slightly inaccurate prediction might lead to a better final decision. This happens frequently especially when the predictive model cannot perfectly represent the ground truth. For example, in our case, the model relies on the Markovian assumption and SUQR assumption in Equation 5.5, which might not be able to fully recover the underlying attacker behavior.

Game-focused learning, instead, can directly optimize the final solution quality by back-propagating from the final solution quality all-the-way back to the predictive model. Game-focused learning has been proven to be able to outperform a standard two-stage learning approach [248], finding a

shortcut to better final solution quality. However, the major issue of back-propagation is the non-differentiable optimization layer in the prescriptive state. Amos et al.[11] provides a method to differentiate through the optimization layer when the optimization program is convex; Perrault et al.[248] instead used quadratic function as a surrogate to deal with the case when the optimization program is non-convex.

More specifically, the idea of tackling non-convex function in Perrault et al.[248] is to approximate the non-convex function by a quadratic function around a local minimum z^{opt} using Taylor expansion, which can be written as:

$$f(z, \theta) \approx f(z^{\text{opt}}, \theta) + (\Delta z)^T \frac{\partial f}{\partial z} + \frac{1}{2}(\Delta z)^T \frac{\partial^2 f}{\partial z^2}(\Delta z) \tag{5.8}$$

where $\Delta z = z - z^{\text{opt}}$. They use this approximate quadratic program (QP) as a surrogate of the non-convex optimization problem, where the optimal solution z^* of QP matches the local optimum z_{opt} computed before. This allows us to differentiate through a QP and compute the gradient of optimal solution z^* with respect to the linear coefficient $p = \frac{\partial f}{\partial z}|_{z=z^{\text{opt}}}$.

$$\frac{df(z^*, \theta^*)}{dw} = \frac{df(z^*, \theta^*)}{dz^*} \frac{dz^*}{dp} \frac{dp}{dw} \tag{5.9}$$

where $p = \frac{\partial f}{\partial z}|_{z=z^{\text{opt}}}$ is a function of θ with $\frac{dp}{dw} = \frac{dp}{d\theta}\frac{d\theta}{d\Phi}\frac{d\Phi}{dw}$ can be decomposed and computed. Equation 5.9 gives us the gradient of the final solution quality with respect to the model parameter w, which allows us to directly run stochastic gradient descent end-to-end. We apply this approach to our domain. The algorithm is sketched in Algorithm 3 and Figure 5.2(b).

Issues of Game-focused Learning Although game-focused learning ideally can achieve better final performance compared to two-stage learning, in this section, we point out two main issues that arise when this game-focused learning is applied to NSGs: scalability and non-convexity.

Algorithm 3: Naive Game-focused Learning[248]

Input: Training data D, initialized $\mathrm{GCN}(\cdot, \cdot; w) : V \times x \to \mathbb{R}$
while *until converge* **do**
  for $(G, \theta^*, x) \in D$ **do**
    Compute prediction θ in Eq. 5.5 by $\Phi = \mathrm{GCN}(V, \xi; w)$
    Find optimum z^{opt} of Optimization 5.7
    $Q = \frac{\partial^2 f(z,\theta)}{\partial z^2}|_{z=z^{\mathrm{opt}}}, p = \frac{\partial f(z,\theta)}{\partial z}|_{z=z^{\mathrm{opt}}} - Qz^*$
    Re-solve QP: $z^* = \underset{z \text{ feasible}}{\operatorname{argmin}} \frac{1}{2} z^\top Q z + z^\top p$
    Update w by gradient $\frac{df(z^*,\theta^*)}{dz^*} \frac{dz^*}{dp} \frac{dp}{dw}$
Return: trained model $\mathrm{GCN}(\cdot, \cdot; w)$

• *Scalability:* In the forward and backward paths of solving QP (Equation 5.8), we need to solve and be able to back-propagate through the QP, which involves the computation of matrix inverse. Taking matrix inverse grows between quadratic and cubically as the size of the decision variable z grows. Moreover, in order to compute the Taylor expansion 5.8, we need to compute the Hessian $\frac{\partial^2 f}{\partial z^2}$ explicitly, which is usually the major bottleneck of the computation cost when the target function f is complex.

• *Non-convexity:* In the non-convex setting, the objective function $f(z, \theta)$ can be non-convex in both z and θ. The gradient-based approaches rely on updating model parameters w and thus θ to improve the solution quality. However, since the f is non-convex in θ, it could create non-convex searching space for gradient-based approaches, which could easily get stuck in local optimum or saddle points. Two-stage methods escape this problem because their loss function $\mathcal{L}(\theta, \theta^*)$ in Equation 5.6 is convex, which gradient-based approaches can more easily handle.

Algorithm 4: Block Game-focused Learning

Input: Training data D, initialized $\text{GCN}(\cdot, \cdot; w) : V \times x \to \mathbb{R}$, block size k
while *until converge* **do**
  for $(G, \theta^*, x) \in D$ **do**
    Compute prediction θ in Eq. 5.5 by $\Phi = \text{GCN}(V, \xi; w)$
    Find optimal solution z^{opt} of Optimization 5.7
    Sample $C \subset \{1, 2, ..., |E|\}$ with $|C| = k$
    $Q_{CC} = \frac{\partial^2 f(z,\theta)}{\partial z_C^2}|_{z=z^{\text{opt}}}, p_C = \frac{\partial f(z,\theta)}{\partial z_C}|_{z=z^{\text{opt}}} - Q_{CC} z_C^*$
    Re-solve quadratic program: $z_C^* = \underset{z_C \text{ feasible}}{\text{argmin}} \frac{1}{2} z_C^\top Q_{CC} z_C + z_C^\top p_C$
    Update w by gradient $\frac{df(z^*,\theta^*)}{dz_C^*} \frac{dz_C^*}{dp_C} \frac{dp_C}{dw}$
Return: trained model $\text{GCN}(\cdot, \cdot; w)$

5.8 Improving Naive Game-focused Learning

In this section, we provide a scalable randomized block update approach to resolve the scalability issue, which also suggests a block game-focused algorithm as a scalable version of game-focused learning approach. To resolve the non-convexity issue, we apply the intermediate loss as a regularization, which helps game-focused methods escape local minimums. We further provide theoretical guarantees to link the randomized block update to the naive game-focused learning approach.

5.8.1 Block Game-focused Learning

Instead of using the entire Taylor expansion (Equation 5.8) to approximate the objective function locally, we can use a partial Taylor expansion with respect to a subset of variables to approximate it:

$$f(z, \theta) \approx f(z^*, \theta) + (\Delta z_C)^T \frac{\partial f}{\partial z_C} + \frac{1}{2} (\Delta z_C)^T \frac{\partial^2 f}{\partial z_C^2} (\Delta z_C), \tag{5.10}$$

where $C \subset \{1, 2, \ldots, |E|\}$ is a subset of indices and z_C is the corresponding truncation over indices C of the entire variables z. Equation 5.10 is equivalent to freezing the variables outside of C and applying Taylor expansion to the rest of them. In this formulation, we only need to compute the Hessian with respect to z_C. When the size of C is significantly smaller than the original variable size $|E|$, it can save the computational time of Hessian quadratically. Furthermore, while back-propagating through the KKT conditions, the QP formulation of Equation 5.10 results in a smaller size of quadratic term, which can reduce the computation of matrix inverse. The block-wise chain rule can be written as:

$$\frac{df(z^*, \theta^*)}{dw} \approx \frac{df(z^*, \theta^*)}{dz_C^*} \frac{dz_C^*}{dp_C} \frac{dp_C}{dw} \tag{5.11}$$

where $p = \frac{\partial f}{\partial z_C}|_{z=z^{\text{opt}}}$, $\frac{dp_C}{dw} = \frac{dp_C}{d\theta} \frac{d\theta}{d\Phi} \frac{d\Phi}{dw}$. When the block size is smaller, the approximation can be more inaccurate. But we will show in the later section that the block gradient is an approximation to the entire gradient.

All the above reasons suggest a randomized block update algorithm, which is described in Algorithm 4*. The algorithm randomly samples a block of variables to compute Hessian and back-propagate accordingly. In comparison, Algorithm 3 requires to compute the entire Hessian matrix $Q = \frac{\partial^2 f(z,\theta)}{\partial z^2}|_{z=z^{\text{opt}}}$, which is usually very expensive. Instead, Algorithm 4 only requires the computation of a block Hessian $Q_{CC} = \frac{\partial^2 f(z,\theta)}{\partial z_C^2}|_{z=z^{\text{opt}}}$, which can save at least quadratic amount of Hessian computation depending on the block size. It can also reduce the running time of the following quadratic program due to reducing the number of variables.

*The implementation of Algorithm 3 and Algorithm 4 can be found in https://github.com/guaguakai/scalable-game-focused-learning

5.8.2 Block Selection

In Algorithm 4, the idea of block game-focused learning is to restrict the focus to a subset of variables and to update accordingly. The choice of the sampled block could affect the convergence rate. Here we propose three block selection approaches: i) *random* approach selects block uniformly at random; ii) *coverage*-based approach randomly selects indices with probability proportional to z^*, which guarantees that there is space for the variables in the block to reallocate coverage; iii) *derivative*-based approach selects indices with probability proportional to the magnitude of the derivatives $\frac{df(z^*,\theta)}{dz_i^*}$, which is the weight put on the chain rule.

5.8.3 Regularization

Another issue associated with the naive game-focused learning method is the non-convex objective function, where gradient-based approaches can encounter issues of local optimums and saddle points. Instead, the two-stage approach optimizes the intermediate loss, which is generally convex in the prediction space. Therefore, we propose to add a weighted two-stage loss as a regularization to smoothify the final objective value. As the training epochs increase, the weight put on the two-stage loss drops exponentially with a decay rate 0.95, pulling the learning back to game-focused methods. This regularization technique helps resolve the non-convexity issue of naive game-focused method, which can achieve better performance afterward.

5.8.4 Approximation Guarantees

In this section, we will show that both Algorithm 3 and 4 have 0 gradient when the prediction perfectly matches to the ground truth, showing that both algorithms are stable at the global optimum. Later on, we will show that Algorithm 4 is an approximate version of Algorithm 3. This shows that our block game-focused approach can not only achieve scalability due to the reduction in Hessian

and QP computation, but it is also aligned with the standard naive game-focused approach with theoretical guarantees.

Theorem 3. *When the intermediate prediction matches the ground truth, i.e.,* $\theta(\cdot, \cdot; w^*) = \theta^*$*, we have* $\frac{df(z^*, \theta^*)}{dw}|_{w=w^*} = 0$ *for both Algorithm 3 and Algorithm 4 with any block C.*

This theorem implies that if the predictive model is rich enough and able to reach the ground truth, then the gradient computed in both algorithms is equal to 0 at the ground truth. So if we can avoid getting stuck by local optimum, then both algorithms will be able to learn the ground truth. This is also true for the two-stage learning when the loss is defined as any convex norms.

Theorem 4. *The quadratic programs in Algorithm 3 and Algorithm 4 share the same primal solutions on the block C. They also share the same dual solution on the non-degenerate constraints containing at least one variable in the block.*

When restricting to variables inside the block, there are some degenerate constraints containing only variables outside of the block, which are always satisfied in the block QP. Thus, there is no restriction put on the dual variable corresponding to these degenerative constraints, which we have no control on them. But in this theorem, we prove that the dual solution to the other valid constraints will match to the dual solution given by the QP in Algorithm 3.

Theorem 5. *Given the primal solution* z^* *and the dual solution* λ^* *of the quadratic program in Algorithm 3 with linear constraints* G, h, A, b*, the Hessian* $Q = \frac{\partial^2 f}{\partial z}$*, linear coefficient* $p = \frac{\partial f}{\partial z}$*, and the sampled indices* $C \subset \{1, 2, ..., |E|\}$*, the gradient* $\frac{dz_C^*}{dp_C} \in \mathbb{R}^{|C| \times |C|}$ *computed in Algorithm 4 is an approximation to the block component of the gradient* $\frac{dz^*}{dp} \in \mathbb{R}^{|E| \times |E|}$ *computed in Algorithm 3. More specifically,*

$$\left\| \left(\frac{dz^*}{dp} \right)_{CC} - \frac{dz_C^*}{dp_C} \right\| \leq \frac{\Delta + \Delta_C}{\mu_{min}(Q)} \max(\lambda^*, 1) \left\| K_{CC}^{-1} \right\| \left\| \left(\frac{dz^*}{dp} \right)_{CC} \right\| \tag{5.12}$$

where $\Delta = \left\| G^\top G + A^\top A \right\|$, $\Delta_C = \left\| Q_{C\bar{C}}^\top Q_{\bar{C}C} \right\|$, *and* $\mu_{min}(Q)$ *is the smallest eigenvalue of positive definite matrix* Q. K_{CC} *is the KTT matrix given by the quadratic program in Algorithm 4.*

The Δ in the numerator is a constant that only depends on the constraint matrices. The other term Δ_C depends on the choice of block C, which measures the magnitude of the off-diagonal elements of the Hessian matrix Q. This is usually a small term when the Hessian Q is diagonally dominant. Another interesting finding is that this bound depends on the convexity of the Hessian Q. When the Hessian is more convex, then the smallest eigenvalue of Q is also larger, giving a stronger bound in Theorem 5. The last term K_{CC}^{-1} measures the stability of the KKT matrix K_{CC}. We can get a good bound if the KKT matrix K_{CC} is far from singular. Greif et al.[124] provides various bounds on the eigenvalues of the KKT matrix. However, in general, poor constraints can still lead to a KKT matrix close to singular. It also indicates that a good choice of C can imply a more stable KKT matrix, leading to a better estimate in Theorem 5.

Theorem 5 also implies an alternative explanation to Algorithm 4, where the gradient in Algorithm 4 is an approximation to the partial gradient with indices C in Algorithm 3:

$$\frac{df(z^*, \theta^*)}{dz_C^*} \left(\frac{dz^*}{dp} \right)_{CC} \frac{dp_C}{dw} \approx \frac{df(z^*, \theta^*)}{dz_C^*} \frac{dz_C^*}{dp_C} \frac{dp_C}{dw} \tag{5.13}$$

which implies that Algorithm 4 can be thought as an approximate block-wise gradient descent of Algorithm 3, which relates to the literature of block coordinate gradient descent[305,246].

5.9 Experiments

In this section, we compare *two-stage (TS), naive game-focused (naive-GF) mentioned in Section 5.7, block game-focused (block-GF), and regularized block game-focused (reg-block-GF)* methods on synthetic data to show that our block game-focused and regularized block game-focused methods can achieve better performance especially in larger instances. These two methods are also able to scale up

to large instances, where the naive game-focused method cannot. Lastly, we study the convergence and scalability of the block game-focused and regularized block game-focused methods with different block sizes and block sampling methods. This allows us to choose the right block size to balance between solution quality and scalability.

5.9.1 Synthetic Data Generation

Graph and features:

we first randomly generate a graph G with various node sizes, 5 random sources with uniform initial distribution π, and 5 random targets with defender penalties $U^d(t)$ for all $t \in T$ drawn from $[-10, -5]$ uniformly at random. We focus on stochastic block model [137] and geometric graphs [333], which can respectively model community structures and physical road networks†. For each node in the graph, we draw an attractiveness value, depending on the shortest distance to the targets plus a uniform noise, as the attacker's unbiased preference. We also randomly generate the past coverage z subject to budget constraints. To generate the node features x, we feed the private attractiveness values to a randomly initialized GCN, where the GCN will output a fixed size vector per node as our node features x. A different level of Gaussian noise was added to the features to model the noise in the real-world scenario.

Attacker behavior:

we choose $\omega = 4$ as the risk aversion parameter suggested by Perrault et al. [248] and Abbasi et al. [1], and set $\eta = 0$ to ignore the future risk factor for the sake of simplicity. For each instance with given attractiveness and the defender coverage, we simulate 100 attacks by initializing the attacker at one

†For stochastic block model, we separate nodes into communities with 10 nodes in each community, then connect nodes within the same community with probability 0.4 and nodes not in the same community with probability 0.1. For geometric graph, we randomly places nodes in a unit square and connects nodes with distance smaller than 0.2.

of the sources and following the localized SUQR behavior described in Section 5.4 until the attacker reaches to one of the targets. These sampled paths Λ are used to reconstruct a Markovian behavior: $\theta^*_{u\to v}(z,x) := \frac{|\{e=(u,v),e\in\alpha,\alpha\in\Lambda\}|}{|\{e=(u,w),e\in\alpha,\alpha\in\Lambda,w\in N(u)\}|}$ [301], which is then used as our ground truth to evaluate the solution quality[‡]. Each instance is composed of the graph G, past coverage z, node features x, the attacker behavior θ^*, and the sampled paths Λ (only used in two-stage method).

Training, validating, and testing:

we generate 50 instances (G, θ^*, z^*, x) as our entire dataset, which are randomly separated into training, validating, testing set with size 35, 5, 10. The model is trained on the training set for 100 epochs, where the best model is chosen from the 100 epochs with the highest score in the validation set. In the following experiments, to achieve statistical significance, for every method and different setup, we ran 50 independent trials and recorded the average results on the testing set.

5.9.2 Solution Quality

In this section, we compare the solution quality of all methods on stochastic block models and geometric graphs. We generate a set of random graphs with features as described in Section 5.9.1, where Gaussian noise with std. of 0.2 is added to the features to model noisy real-world data. We set $b = 2$. As our goal is efficient approaches for adversary models in large-scale NSGs, the focus of this paper is then on experimenting with many different settings (graph sizes and types), techniques (different variations of game-focused learning), noise, and other variables in building an adversary model. In addition, since we care more about how much defender utility that various learning approaches can

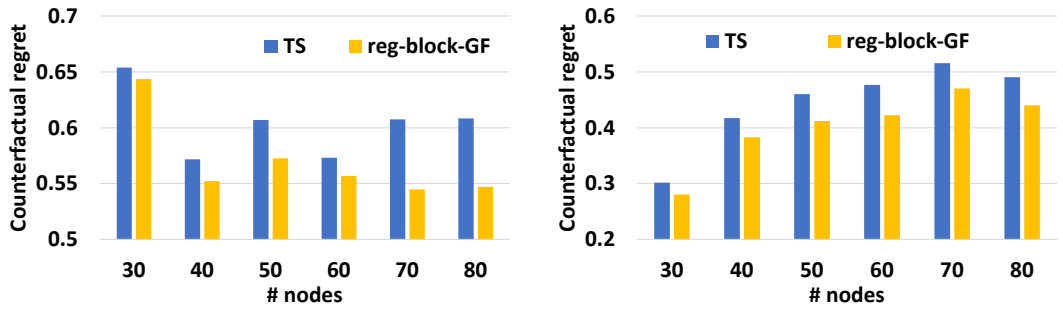
[‡]The reason of using sampled paths instead of the actual generated attractiveness values as our ground truth is to align with the real-world data, where it is almost impossible to have access to the underlying attacker preference or Markovian behavior; instead, we generally have access to the paths or edges where illegal activities have been found, which can be used as sampled paths or edges and used to reconstruct the Markovian behavior as we did here.

improve, we focus on the *counterfactual regret*, which is defined as the gap between the defender utility of our solution and the true optimum when the ground truth is given in advance. Smaller regret implies that the solution is closer to the actual optimum.

In Figure 5.3(a) and 5.3(b), we can see that our regularized block game-focused method outperforms two-stage method (note that all of the improvements in the average regret reported by the reg-block-GF method over the two-stage method are statistically significant with $p < 0.05$). When the instance gets larger, the difference between two approaches also gets larger, showcasing the limit of the standard two-stage behavior learning approach. In Figure 5.4(a) and 5.4(b), we compare the solution quality of different game-focused methods. Due to the computational issue, the naive game-focused method can only scale up to graphs with 40 nodes. The block game-focused method can scale up to larger instances but it sacrifices some solution quality compared to the naive game-focused approach. Finally, the regularized block game-focused method can achieve both scalability and solution quality by using the block update and regularization term.

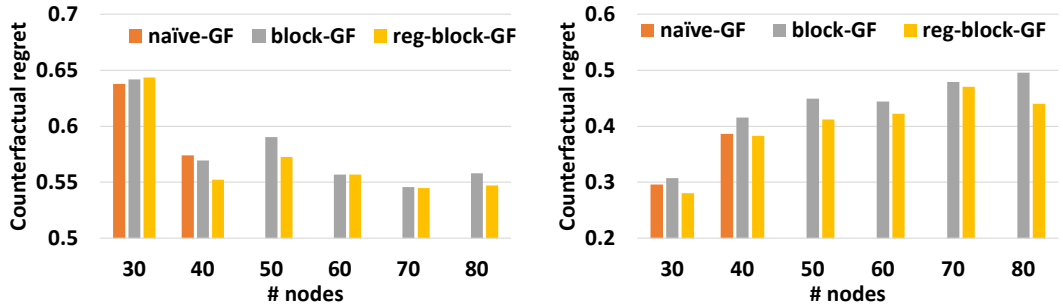
5.9.3 The Impact of Noise

Figure 5.5(a) and 5.5(b) compare the performances under different level of noise, where a noise with std. of r is added to the normalized features. We can see that the more noise implies larger regret and poorer performance. But we can also notice that the gap between regularized block game-focused method and the two-stage method gets larger when more noise is introduced. This is probably due to the mismatch between the low intermediate loss and the good final solution quality when the feature is noisy. This also explains why regularized block game-focused method can outperform two-stage in Figure 5.3 when the features are noisy.



(a) Stochastic block model **(b)** Geometric graphs

Figure 5.3: Solution quality comparison between two-stage and regularized block game-focused method. The difference in solution quality gets larger when the graph size increases.

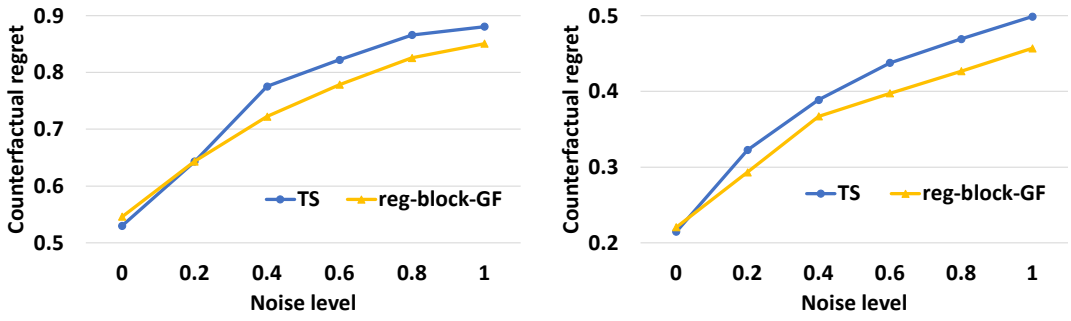


(a) Stochastic block model **(b)** Geometric graphs

Figure 5.4: Solution quality comparison between game-focused methods. Randomized block update can improve scalability while the regularization can improve the solution quality.

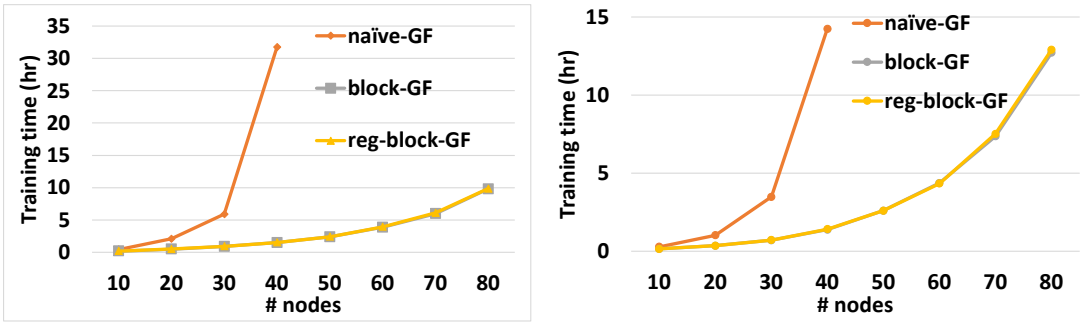
5.9.4 Scalability

Figure 5.6(a) and 5.6(b) show the scalability of all game-focused methods. We limit the training time to be up to 48 hours. Any programs last more than that were cut and the corresponding results were recorded. Naive game-focused method can only handle graphs with up to 40 nodes and it scales extremely poorly. Our proposed methods, block game-focused and regularized block game-focused with a block size #nodes/2, can scale to much larger instances.



(a) Stochastic block model **(b)** Geometric graphs

Figure 5.5: The figures show the effect of noise to all the methods, where regularized block game-focused method is more resilient to noise in the features.



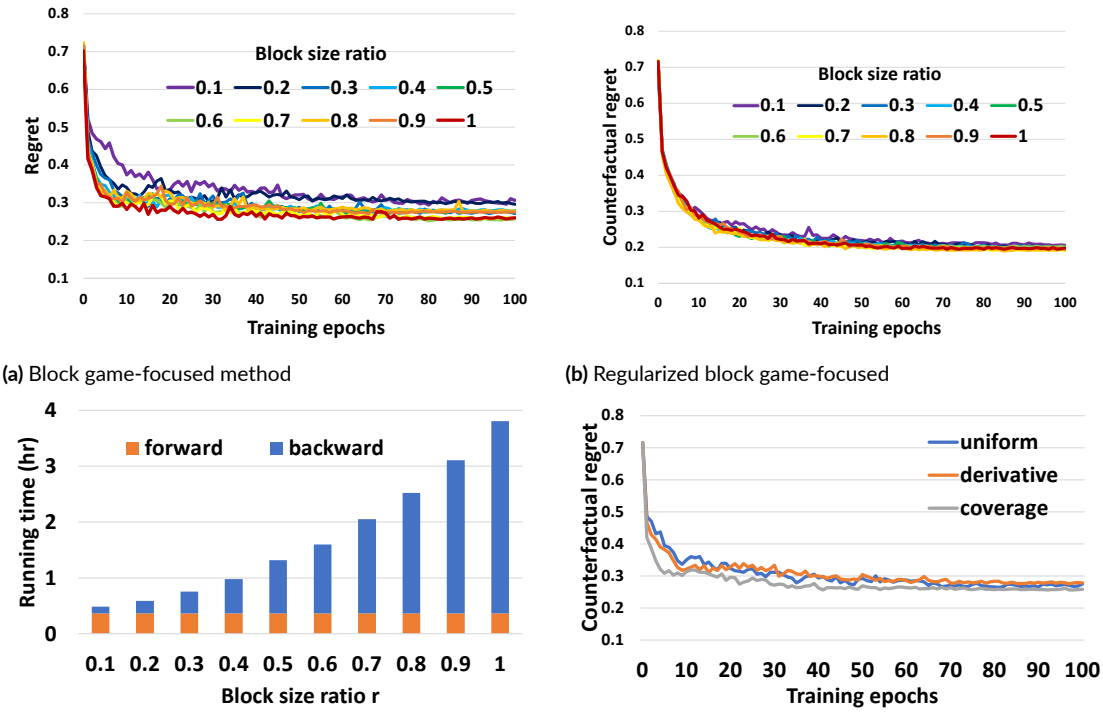
(a) Stochastic block model **(b)** Geometric graphs

Figure 5.6: Naive game-focused method can only scale up to 40 nodes. Instead, block game-focused and regularized block game-focused can solve larger instances with 80 nodes.

5.9.5 Block Size Selection

To study the effect of block size, we select various block sizes proportional to the total number of variables and run the block game-focused learning and regularized block game-focused methods to compare the convergence. In Figure 5.7(a), we can see that for the block-game-focused method, the convergence and the final performance are better when the block size is larger. Figure 5.7(b) shows the convergence of regularized block game-focused method with different block sizes. In this case, a larger block size still helps, but the difference is relatively tiny.

Figure 5.7(c) shows the running time of the forward (lines 4-5) and backward path (lines 6-9



(a) Block game-focused method

(b) Regularized block game-focused

(c) Running time

(d) Block selection

Figure 5.7: Figure (a) and (b) show the convergence rate of different block sizes. Figure (c) shows the running time of backward path for different block sizes, which grows significantly more than linear. Figure (d) shows the effect of different block sampling methods. All methods converge with slightly different speed, where coverage-based sampling is the best and it is also what we use in other experiments.

in Algorithm 4) for the block game-focused method with various block sizes, where forward path solves prescriptive stage with black-box optimization and the backward path requires computing the Hessian and solving the quadratic program to back-propagate. In practice, we would like to select a block size such that the running time of forward and backward paths are of the same order to balance between the convergence and scalability, which explains the reason that we eventually choose block size $=$ #nodes/2 for all other experiments. Lastly, Figure 5.7(d) compares different block selections mentioned in Section 5.8.2, where convergence speed differs but mostly lead to the same point. Coverage-based selection converges the most quickly, and thus we use it throughout the other experiments.

5.10 Conclusions

In this paper, we introduce a fundamentally different behavior learning approach, game-focused learning, to network security games, placing the downstream defender utility maximization problem into the loop of behavior learning. We propose a novel local SUQR model as our adversary model, where GCNs can be applied to automatically handle the information propagation in the graph. We further identify two existing issues of game-focused learning method: scalability and non-convexity, which are addressed by our block game-focused and by regularizing respectively. Block game-focused method can largely reduce the computational cost while maintaining the focus on the final solution quality as naive game-focused learning does. We also provide theoretical guarantees on the block game-focused method. In the experimental section, we run extensive experiments to verify the reduction on the training time and show an improvement in terms of solution quality. The block game-focused method reduces the training time, but sacrifices a little solution quality, while regularized block game-focused can achieve both speed and performance.

6

Automatically Learning Surrogates for Decision-focused Learning

6.1 Introduction

Uncertainty is a common feature of many real-world decision-making problems because critical data may not be available when a decision must be made. Here is a set of representative examples:

recommender systems with missing user-item ratings[144], portfolio optimization where future performance is uncertain[208], and strategic decision-making in the face of an adversary with uncertain objectives[164]. Often, the decision-maker has access to features that provide information about the values of interest. In these settings, a *predict-then-optimize*[90] approach naturally arises, where we learn a model that maps from the features to a value for each parameter and optimize using this point estimate[360]. In principle, any predictive modeling approach and any optimization approach can be applied, but using a generic loss function to train the model may result in poor decision performance. For example, a typical ratings prediction approach in recommendation system may equally weight errors across different items, but in the recommendation task, misclassifying a trendy item can result in more revenue loss than misclassifying an ordinary item. We may instead want to train our model using a "task-based" or "decision-focused" loss, approximating the decision quality induced by the predictive model, which can be done by embedding the optimization problem as a layer in the training pipeline. This end-to-end approach improves performance on a variety of tasks [42,338,80].

Unfortunately, this end-to-end approach suffers from poor scalability because the optimization problem must be solved and differentiated through on every training iteration. Furthermore, the output of the optimization layer may not be smooth, sometimes leading to instabilities in training and consequently poor solution quality. We address these shortcomings that arise in the end-to-end approach due to the presence of a complex optimization layer by replacing it with a simpler surrogate problem. The surrogate problem is learned from the data by automatically finding a reparameterization of the feasible space in terms of meta-variables, each of which is a linear combination of the original decision variables. The new surrogate problem is generally cheaper to solve due to the smaller number of meta-variables, but it can be lossy—the optimal solution to the surrogate problem may not match the optimal solution to the original. Since we can differentiate through the surrogate layer, we can optimize the choice of surrogate together with predictive model training to

minimize this loss. The dimensionality reduction offered by a compact surrogate simultaneously reduces training times, helps avoid overfitting, and sometimes smooths away bad local minima in the training landscape.

In short, we make several contributions. First, we propose a linear reparameterization scheme for general optimization layers. Second, we provide theoretical analysis of this framework along several dimensions: (i) we show that desirable properties of the optimization problem (convexity, submodularity) are retained under reparameterization; (ii) we precisely characterize the tractability of the end-to-end loss function induced by the reparameterized layer, showing that it satisfies a form of coordinate-wise quasiconvexity; and (iii) we provide sample complexity bounds for learning a model which minimizes this loss. Finally, we demonstrate empirically on a set of three diverse domains that our approach offers significant advantages in both training time and decision quality compared previous approaches to embedding optimization in learning.

6.2 Related work

Surrogate models [106,260,186] are a classic technique in optimization, particularly for black-box problems. Previous work has explored linear reparameterizations to map between low and high fidelity models of a physical system [36,265,14] (e.g., for aerospace design problems). However, both the motivation and underlying techniques differ crucially from our work: previous work has focused on designing surrogates by hand in a domain-specific sense, while we leverage differentiation through the optimization problem to automatically produce a surrogate that maximizes overall decision quality.

Our work is closest to the recent literature on differentiable optimization. Amos et al. [11] and Agrawal et al. [3] introduced differentiable quadratic programming and convex programming layers, respectively, by differentiating through the KKT conditions of the optimization problem. Donti et

al.[80] and Wilder et al.[338] apply this technique to achieve end-to-end learning in convex and discrete combinatorial programming, respectively. Perrault et al.[249] applied the technique to game theory with a non-convex problem, where a sampling approach was proposed by Wang et al.[320] to improve the scalability of the backward pass. All the above methods share scalability and non-smoothness issues: each training iteration requires solving the entire optimization problem and differentiating through the resulting KKT conditions, which requires $O(n^3)$ time in the number of decision variables and may create a non-smooth objective. Our surrogate approach aims to rectify both of these issues.

6.3 Problem Statement

We consider an optimization problem of the form: $\min_{z \text{ feasible}} f(z, \theta_{\text{true}})$. The objective function depends on a parameter $\theta_{\text{true}} \in \Theta$. If θ_{true} were known, we assume that we could solve the optimization problem using standard methods. We consider the case that parameter θ_{true} is unknown and must be inferred from the given available features x. We assume that x and θ_{true} are correlated and drawn from a joint distribution $\mathcal{D}$, and our data consists of samples from $\mathcal{D}$. Our task is to select the optimal decision $z^*(x)$, function of the available feature, to optimize the expected objective value:

$$\min_{z^* \text{ feasible}} E_{(x,\theta_{\text{true}})\sim\mathcal{D}}[f(z^*(x), \theta_{\text{true}})] \tag{6.1}$$

In this paper, we focus on a *predict-then-optimize*[90,87] framework, which proceeds by learning a model $m_w(\cdot)$, mapping from the features x to the missing parameter θ_{true}. When feature x is given, we first infer $\theta = m_w(x)$ and then solve the resulting optimization problem to get the optimal solution z^*:

$$\min_z \quad f(z, \theta), \quad \text{s.t.} \quad h(z) \leq 0, \quad Az = b \tag{6.2}$$

This reduces the decision-making problem with unknown parameters to a predictive modeling problem: how to learn a model $m_w(\cdot)$ that leads to the best performance.

A standard approach to solve the predict-then-optimize problem is *two-stage* learning, which trains the predictive model without knowledge of the decision-making task (Figure 6.1). The predictive model minimizes the mismatch between the predicted parameters and the ground truth: $E_{(x,\theta_{\text{true}})\in\mathcal{D}}\, \ell(m_w(x), \theta_{\text{true}})$, with any loss metric ℓ. Such a two-stage approach is efficient in terms of training the model, but it may lead to poor performance when a standard loss function is used. Performance can be improved if the loss function is carefully chosen to suit the task[88], but doing so is challenging for an arbitrary optimization problem.

Gradient-based end-to-end learning approaches in domains with optimization layers involved, e.g., decision-focused learning[338,80], directly minimize Equation 6.1 as the training objective, which requires back-propagating through the optimization layer in Equation 6.2. This end-to-end approach is able to achieve better solution quality compared to two-stage learning, in principle. However, because the decision-focused approach has to repeatedly solve the optimization program and back-propagate through it, scalability becomes a serious issue. Additionally, the complex optimization layer can also jeopardize the smoothness of objective value, which is detrimental for training parameters of a neural network-based predictive model with gradient-based methods.

6.4 Surrogate Learning

The main idea of the surrogate approach is to replace Equation 6.2 with a carefully selected surrogate problem. To simplify Equation 6.2, we can linearly reparameterize $z = Py$, where $y \in \mathbb{R}^m$ with $m \ll n$ and $P \in \mathbb{R}^{n\times m}$,

$$\min_{y} \quad g_P(y, \theta) := f(Py, \theta) \quad \text{s.t.} \quad h(Py) \leq 0, \quad APy = b \tag{6.3}$$

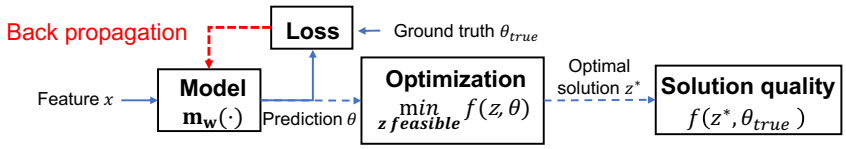


Figure 6.1: Two-stage learning back-propagates from the loss to the model, ignoring the latter effect of the optimization layer.

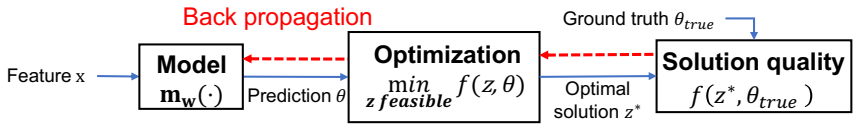


Figure 6.2: End-to-end decision-focused learning back-propagates from the solution quality through the optimization layer to the model we aim to learn.

Since this reparameterization preserves all the equality and inequality constraints in Equation 6.2, we can easily transform a feasible low-dimensional solution y^* back to a feasible high-dimensional solution with $z^* = Py^*$. The low-dimensional surrogate is generally easier to solve, but lossy, because we restrict the feasible region to a hyperplane spanned by P. If we were to use a random reparameterization, the solution we recover from the surrogate problem could be far from the actual optimum in the original optimization problem, which could significantly degrade the solution quality.

This is why we need to learn the surrogate and its reparameterization matrix. Because we can differentiate through the surrogate optimization layer, we can estimate the impact of the reparameterization matrix on the final solution quality. This allows us to run gradient descent to learn the reparameterization matrix P. The process is shown in Figure 6.3. Notice that the surrogate problem also takes the prediction θ of the predictive model as input. This implies that we can jointly learn the predictive model and the reparameterization matrix by solely solving the cheaper surrogate problem.

Differentiable optimization In order to differentiate through the optimization layer as shown in Figure 6.2, we can compute the derivative of the solution quality, evaluated on the optimal solution z^* and true parameter θ_{true}, with respect to the model's weights w by applying the chain

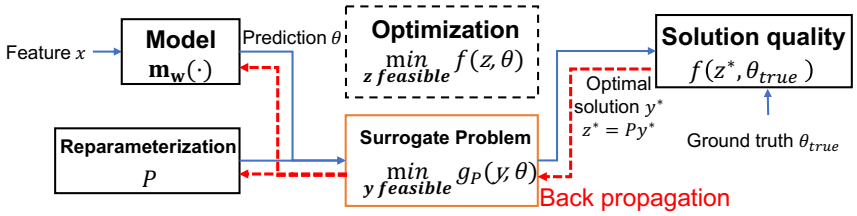


Figure 6.3: Surrogate decision-focused learning reparameterizes Equation 6.2 by $z = Py$ to get a surrogate model in Equation 6.3. Then, forward and backward passes go through the surrogate model with a lower dimensional input y to compute the optimal solution and train the model.

rule:

$$\frac{df(z^*, \theta_{\text{true}})}{dw} = \frac{df(z^*, \theta_{\text{true}})}{dz^*} \frac{dz^*}{d\theta} \frac{d\theta}{dw}$$

where $\frac{dz^*}{d\theta}$ can be obtained by differentiating through KKT conditions of the optimization problem.

Similarly, in Figure 6.3, we can apply the same technique to obtain the derivatives with respect to the weights w and reparameterization matrix P:

$$\frac{df(z^*, \theta_{\text{true}})}{dw} = \frac{df(z^*, \theta_{\text{true}})}{dz^*} \frac{dz^*}{dy^*} \frac{dy^*}{d\theta} \frac{d\theta}{dw}, \quad \frac{df(z^*, \theta_{\text{true}})}{dP} = \frac{df(z^*, \theta_{\text{true}})}{dz^*} \frac{dz^*}{dy^*} \frac{dy^*}{dP}$$

where y^* is the optimal solution of the surrogate problem, $z^* = Py^*$, and $\frac{dy^*}{dw}, \frac{dy^*}{dP}$ can be computed by differentiating through the KKT conditions of the surrogate optimization problem.

6.5 Analysis of Linear Reparameterization

The following sections address three major theoretical aspects: (i) complexity of solving the surrogate problem, (ii) learning the reparameterization, and (iii) learning the predictive model.

6.5.1 Convexity and DR-Submodularity of the Reparameterized Problem

In this section, we assume the predictive model and the linear reparameterization are fixed. We prove below that convexity and continuous diminishing-return (DR) submodularity[46] of the original function f is preserved after applying the reparameterization. This implies that the new surrogate problem can be efficiently solved by gradient descent or by Frank-Wolfe[47,145,108] with an approximation guarantee.

Proposition 1. *If f is convex, then $g_P(y, \theta) := f(Py, \theta)$ is convex.*

Proposition 2. *If f is DR-submodular and $P \geq 0$, then $g_P(y, \theta) := f(Py, \theta)$ is DR-submodular.*

6.5.2 Convexity of Reparameterization Learning

In this section, we assume the predictive model m is fixed. We want to analyze the convergence of learning the surrogate and its linear reparameterization P. Let us denote the optimal value of the optimization problem in the form of Equation 6.3 to be $\mathrm{OPT}(\theta, P) := \min_{y \text{ feasible}} g_P(y, \theta) \in \mathbb{R}$. It would be ideal if $\mathrm{OPT}(\theta, P)$ is convex in P so that gradient descent would be guaranteed to recover the optimal reparameterization. Unfortunately, this is not true in general, despite the fact that we use a linear reparameterization: $\mathrm{OPT}(\theta, P)$ is not even globally quasiconvex in P.

Proposition 3. *$OPT(\theta, P) := \min_{y \text{ feasible}} g_P(y, \theta)$ is not globally quasiconvex in P.*

Fortunately, we can guarantee the partial quasiconvexity of $\mathrm{OPT}(\theta, P)$ in the following theorem:

Theorem 6. *If $f(\cdot, \theta)$ is quasiconvex, then $OPT(\theta, P) := \min_{y \text{ feasible}} g_P(y, \theta)$ is quasiconvex in P_i, the i-th column of matrix P, for any $1 \leq i \leq m$, where $P = [P_1, P_2, \ldots, P_m]$.*

This indicates that the problem of optimizing each meta-variable given the values of the others is tractable, providing at least some reason to think that the training landscape for the reparameterization is amenable to gradient descent. This theoretical motivation is complemented by our experiments, which show successful training with standard first-order methods.

6.5.3 Sample Complexity of Learning Predictive Model in Surrogate Problem

In this section, we fix the linear reparameterization and analyze the sample complexity of learning the predictive model to achieve small decision-focused loss in the objective value. We analyze a special case where our objective function f is a linear function and the feasible region S is compact, convex, and polyhedron. Given the hypothesis class of our model $m \in \mathcal{H}$, we can use results from Balghiti et al.[87] to bound the Rademacher complexity and the generalization bound of the solution quality obtained from the surrogate problem. For any hypothesis class with a finite Natarajan dimension, the surrogate problem preserves the linearity of the objective function. Thus learning in the linear surrogate problem also *preserves the convergence of the generalization bound, and thus the convergence of the solution quality*. In the case of a linear hypothesis class $H = H_{\text{lin}}$, we can derive a closed-form bound. The Rademacher complexity depends on the dimensionality of the surrogate problem and the diameter of the feasible region, which can be shrunk by using a low-dimensional surrogate:

Theorem 7. *Let $\mathcal{H}_{lin}$ be the hypothesis class of all linear function mappings from $x \in \mathcal{X} \subset \mathbb{R}^p$ to $\theta \in \Theta \in \mathbb{R}^n$, and let $P \in \mathbb{R}^{n \times m}$ be a linear reparameterization used to construct the surrogate. The expected Rademacher complexity over t i.i.d. random samples drawn from $\mathcal{D}$ can be bounded by:*

$$Rad^t(\mathcal{H}_{lin}) \leq 2mC\sqrt{\frac{2p\log(2mt\,\|P^+\|\rho_2(S))}{t}} + O(\frac{1}{t}) \tag{6.4}$$

where $C := sup_\theta(max_z f(z,\theta) - min_z f(z,\theta))$ *is the gap between the optimal solution quality and the worst solution quality,* $\rho_2(S)$ *is the diameter of the set S, and* P^+ *is the pseudoinverse.*

Equation 6.4 gives a bound on the Rademacher complexity, an upper bound on the generalization error with t samples given. Although a lower dimensional surrogate leads to less representational power (i.e., lower decision quality), it also leads to better generalizability. This implies that we have to choose an appropriate reparameterization size to balance representational power and generalizability.

6.6 Experiments

We conduct experiments on three different domains where decision-focused learning has been applied: (i) adversarial behavior learning in network security games with a non-convex objective[320], (ii) movie recommendation with a submodular objective[338], and (iii) portfolio optimization problem with a convex quadratic objective[97]. Throughout all the experiments, we compare the performance and the scalability of the surrogate learning (**surrogate**), two-stage (**TS**), and decision-focused (**DF**) learning approaches. Performance is measured in terms of regret, which is defined as the difference between the achieved solution quality and the solution quality if the unobserved parameters θ^* were observed directly—smaller is better. To compare scalability, we show the training time per epoch and inference time. The inference time corresponds to the time required to compute a decision for all instances in the testing set after training is finished. A short inference time may have intrinsic value, e.g., allowing the application to be run in edge computing settings. All methods are trained using gradient descent with optimizer Adam[171] with learning rate 0.01 and repeated over 30 independent runs to get the average. Each model is trained for at most 100 epochs with early stopping[256] criteria when 3 consecutive non-improving epochs occur on the validation set. The reparameteriza-

tion size is set to be 10% of the problem size throughout all three examples*.

6.6.1 Adversarial Behavior Learning and Interdiction Games

Given a network structure $G = (V, E)$, a NSG (network security game) [332,101,272] models the interaction between the defender, who places checkpoints on a limited number of edges in the graph, and an attacker who attempts to travel from a source to any of a set of target nodes in order to maximize the expected reward. The NSG is an extension of Stackelberg security games [284,163], meaning that the defender commits to a mixed strategy first, after which the attacker chooses the path (having observed the defender's mixed strategy but not the sampled pure strategy). In practice, the attacker is not perfectly rational. Instead, the defender can attempt to predict the attacker's boundedly rational choice of path by using the known features of the nodes en route (e.g., accessibility or safety of hops) together with previous examples of paths chosen by the attacker.

Once the parameters θ of the attacker behavioral model are given, finding the optimal defender's strategy reduces to an optimization problem $\max f(z, \theta)$ where z_e is the probability of covering edge $e \in E$ and f gives the defender's expected utility for playing mixed strategy z when the attacker's response is determined by θ. The defender must also satisfy the budget constraint $\sum_{e \in E} z_e \leq k$ where $k = 3$ is the total defender resources. We use a GCN (graph convolutional network) [223,172,130] to represent the predictive model of the attacker. We assume the attacker follows reactive Markovian behavior [320,126], meaning that the attacker follows a random walk through the graph, where the probability of transitioning across a given edge (u, v) is a function of the defender's strategy z and an unknown parameter θ_v representing the "attractiveness" of node v. The walk stops when the attacker either is intercepted by crossing an edge covered by the defender or reaches a target. The defender's utility is $-u(t)$ if the attacker reaches target t and 0 otherwise, and f takes an expectation

*The implementation of this chapter can be found in the following link: https://github.com/guaguakai/surrogate-optimization-learning

over both the random placement of the defender's checkpoints (determined by z) and the attacker's random walk (determined by z and θ). Our goal is to learn a GCN which takes node features as input and outputs the attractiveness over nodes θ.

Experimental setup: We generate random geometric graphs of varying sizes with radius 0.2 in a unit square. We select 5 nodes uniformly at random as targets with payoffs $u(t) \sim \text{Unif}(5, 10)$ and 5 nodes as sources where the attacker chooses uniformly at random from. The ground truth attractiveness value θ_v of node $v \in V$ is proportional to the proximity to the closest target plus a random perturbation sampled as $\text{Unif}(-1, 1)$ which models idiosyncrasies in the attacker's preferences. The node features x are generated as $x = \text{GCN}(\theta) + 0.2\mathcal{N}(0, 1)$, where GCN is a randomly initialized GCN with four convolutional layers and three fully connected layers. This generates random features with correlations between x_v (the features of node v) and both θ_v and the features of nearby nodes. Such correlation is expected for real networks where neighboring locations are likely to be similar. The defender's predictive model (distinct from the generative model) uses two convolutional and two fully connected layers, modeling a scenario where the true generative process is more complex than the learned model. We generate 35 random (x, θ) pairs for the training set, 5 for validation, and 10 for testing. Since decision-focused (DF) learning fails to scale up to larger instances, we additionally compare to a block-variable sampling approach specialized to NSG [320] (**block**), which can speed up the backward pass by back-propagating through randomly sampled variables.

6.6.2 Movie Recommendation and Broadcasting Problem

In this domain, a broadcasting company chooses k movies out of a set of n available to acquire and show to their customers C. k reflects a budget constraint. Each user watches their favorite T movies, with a linear valuation for the movies they watch. This is a variant of the classic facility location problem; similar domains have been used to benchmark submodular optimization algorithms [193,81].

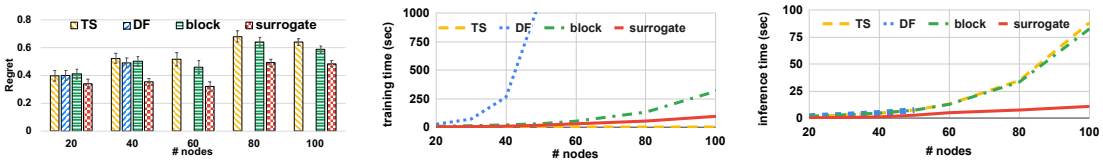
In our case, the additional complication is that the user's preferences are unknown. Instead, the company uses user's past behavior to predict $\theta_{ij} \in [0, 1]$, the preference score of user j for movie i.

The company would like to maximize the overall satisfaction of users without exceeding the budget constraint $k = 10$. The variable $z = \{z_i\}_{i \in \{1,2,\ldots,n\}}$ represents the decision of whether to acquire movie i or not. Once the preferences θ_{ij} are given, the company wants to maximize the objective function:

$$f(z) := \sum_{j \in C} \text{user } j\text{'s satisfaction} = \sum_{j \in C} \max_{\substack{s_j \in \{0,1\}^n \\ \text{s.t.} \sum_i s_{ij} = T}} \sum_{i \in \{1,2,\ldots,n\}} z_i s_{ij} \theta_{ij} \tag{6.5}$$

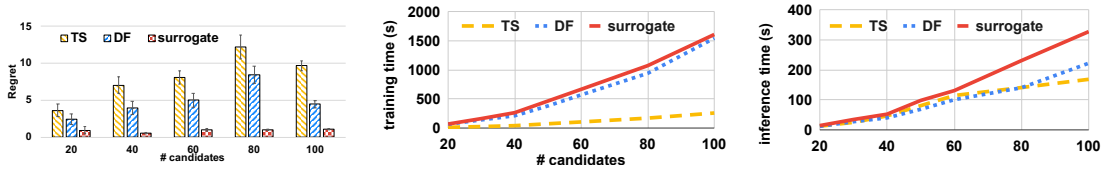
where s_j denotes the user j's selection over movies.

Experimental setup: We use neural collaborative filtering[135] to learn the user preferences. Commonly used in recommendation systems, the idea is to learn an embedding for each movie and user. The ratings are computed by feeding the concatenated user's and movie's embeddings to a neural network with fully connected layers. We use MovieLens[131] as our dataset. The MovieLens dataset includes 25M ratings over 62,000 movies by 162,000 users. We first randomly select n movies as our broadcasting candidates. We additionally select 200 movies and use the users' ratings on the movies as the users' features. Then we split the users into disjoint groups of size 100 and each group serves as an instance of broadcasting task, where we want to choose $k = 10$ from the n candidate movies to recommend to the group members. Each user chooses $T = 3$ movies. 70% of the user groups are used for training, 10% for validation, and 20% for testing.



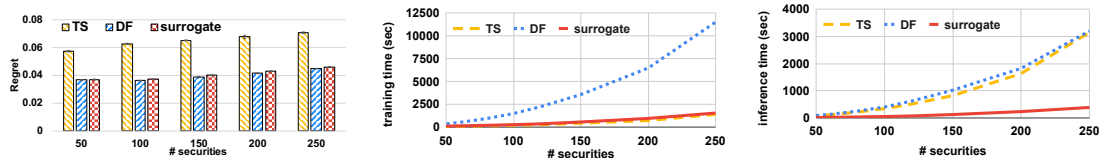
(a) Performance in regret **(b)** Training time per epoch **(c)** Inference time

Figure 6.4: Experimental results in network security games with a non-convex optimization problem.



(a) Performance in regret **(b)** Training time per epoch **(c)** Inference time

Figure 6.5: Experimental results in movie recommendation with a submodular objective. Surrogate achieves much better performance by smoothing the training landscape.



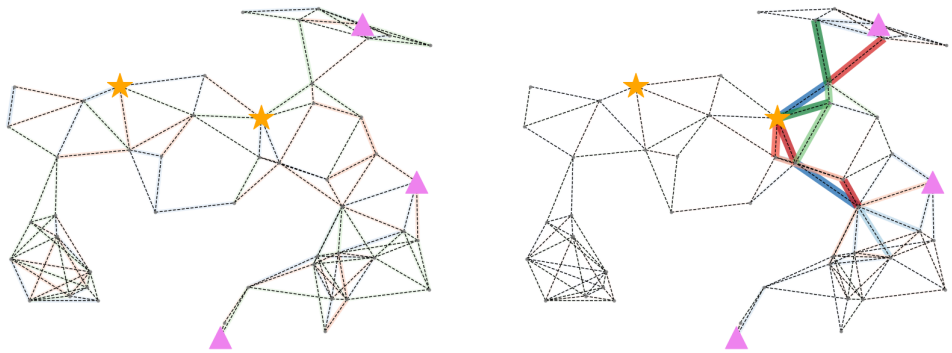
(a) Performance in regret **(b)** Training time per epoch **(c)** Inference time

Figure 6.6: Experimental results in portfolio optimization with a convex optimization problem. Surrogate performs comparably, but achieves a 7-fold speedup in training and inference.

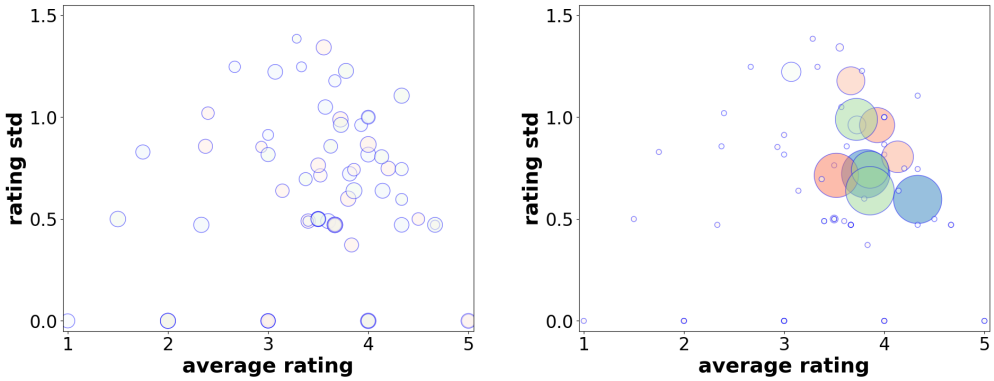
6.6.3 Stock Market Portfolio Optimization

Portfolio optimization can be treated as an optimization problem with missing parameters[254], where the return and the covariance between stocks in the next time step are not known in advance. We learn a model that takes features for each security and outputs the predicted future return. We adopt the classic Markowitz[208,217] problem setup, where investors are risk-averse and wish to maximize a weighted sum of the immediate net profit and the risk penalty. The investor chooses a vector $z \geq 0$ with $\sum z_i = 1$, where z_i represents the fraction of money invested in security i. The investor aims to maximize the penalized immediate return $f(z) := p^\top z - \lambda z^\top Q z$, where p is the immediate net return of all securities and $Q \in \mathbb{R}^{n \times n}$ is a positive semidefinite matrix representing the covariance between the returns of different securities. A high covariance implies two securities are highly correlated and thus it is more risky to invest in both. We set the risk aversion constant to be $\lambda = 2$.

Experimental setup: We use historical daily price and volume data from 2004 to 2017 downloaded from the Quandl WIKI dataset[259]. We evaluate on the SP500, a collection of the 505 largest companies representing the American market. Our goal is to generate daily portfolios of stocks from a given set of candidates. Ground truth returns are computed from the time series of prices, while the ground truth covariance of two securities at a given time step is set to be the cosine similarity of their returns in the next 10 time steps. We take the previous prices and rolling averages at a given time step as features to predict the returns for the next time step. We learn the immediate return p via a neural network with two fully connected layers with 100 nodes each. To predict the covariance matrix Q, we learn an 32-dimensional embedding for each security, and the predicted covariance between two securities is the cosine similarity of their embeddings. We chronologically split the dataset into training, validation, and test sets with 70%, 10%, and 20% of the data respectively.



(a) A NSG instance with 50 nodes, 2 targets (orange stars), and 3 sources (purple triangles).



(b) 100 candidate movies shown as circles with their average ratings and standard deviations as two axes.

Figure 6.7: These plots visualize how the surrogate captures the underlying problem structure. Both domains use a reparameterization with 3 meta-variables, each shown in red, blue, and green. The color indicates the most significant meta-variable governing the edge or circle, while the color intensity and size represent the weights put on it. The left figure in both domains shows the initial reparameterization, while the right figure shows the reparameterization after 20 epochs of training.

6.7 Discussion of Experimental Results

Performance: Figures 6.4(a), 6.5(a), and 6.6(a) compare the regret of our surrogate approach to the other approaches. In the non-convex (Figure 6.4(a)) and submodular (Figure 6.5(a)) settings, we see a larger improvement in solution quality relative to decision-focused learning. This is due to the huge number of local minima and plateaus in these two settings where two-stage and decision-focused approaches can get stuck. For example, when an incorrect prediction is given in the movie recommendation domain, some recommended movies could have no users watching them, resulting in a sparse gradient due to non-smoothness induced by the max in the objective function. The surrogate approach can instead spread the sparse gradient by binding variables with meta-variables, alleviating gradient sparsity. We see relatively less performance improvement (compared to decision-focused) when the optimization problem is strongly convex and hence smoother (Figure 6.6(a)), though the surrogate approach still achieves similar performance to the decision-focused approach.

Scalability: When the objective function is non-convex (Figure 6.4(b), 6.4(c)), our surrogate approach yields substantially faster training than standard decision-focused learning approaches (DF and block). The boost is due to the dimensionality reduction of the surrogate optimization problem, which can lead to speedups in solving the surrogate problem and back-propagating through the KKT conditions. While the two-stage approach avoids solving the optimization problem in the training phase (trading off solution quality), at test time, it still has to solve the expensive optimization problem, resulting a similarly expensive inference runtime in Figure 6.4(c).

When the objective function is submodular (Figure 6.5(b), 6.5(c)), the blackbox optimization solver[313] we use in all experiments converges very quickly for the decision-focused method, resulting in training times comparable to our surrogate approach. However, Figure 6.5(a) shows that the decision-focused approach converges to a solution with very poor quality, indicating that rapid con-

vergence may be a symptom of the uninformative local minima that the decision-focused method becomes trapped in.

Lastly, when the optimization problem is a quadratic program (Figure 6.6(b), 6.6(c)), solving the optimization problem can take cubic time, resulting in around a cubic speedup from the dimensionality reduction offered by our surrogate. Consequently, we see 7-fold faster training and inference times.

Visualization: We visualize the reparameterization for the NSG and movie recommendation domains in Figure 6.7. The initial reparameterization is shown in Figure 6.7(a) and 6.7(b). Initially, the weights put on the meta-variables are randomly chosen and no significant problem structure—no edge or circle colors—can be seen. After 20 epochs of training, in Figure 6.7(a), the surrogate starts putting emphasis on some important cuts between the sources and the targets, and in Figure 6.7(b), the surrogate is focused on distinguishing between different top-rated movies with some variance in opinions to specialize the recommendation task. Interestingly, in Figure 6.7(b), the surrogate puts less weight on movies with high average rating but low standard deviation because these movies are very likely undersampled and we do not have enough people watching them in our training data. Overall, adaptively adjusting the surrogate model allows us to extract the underlying structure of the optimization problem using few meta-variables. These visualizations also help us understand how focuses are shifted between different decision variables.

6.8 Conclusion

In this paper, we focus on the shortcomings of scalability and solution quality that arise in end-to-end decision-focused learning due to the introduction of the differentiable optimization layer. We address these two shortcomings by learning a compact surrogate, with a learnable linear reparameterization matrix, to substitute for the expensive optimization layer. This surrogate can be jointly

trained with the predictive model by back-propagating through the surrogate layer. Theoretically, we analyze the complexity of the induced surrogate problem and the complexity of learning the surrogate and the predictive model. Empirically, we show this surrogate learning approach leads to improvement in scalability and solution quality in three domains: a non-convex adversarial modeling problem, a submodular recommendation problem, and a convex portfolio optimization problem.

Part II

Optimization in Online Learning

7

Improving GP-UCB Algorithm by Harnessing Decomposed Feedback

7.1 Introduction

Many challenging sequential decision making problems involve interventions in complex physical or social systems, where the system dynamics must be learned over time. For instance, a challenge

commonly faced by policymakers is to control disease outbreaks[316], but the true process by which disease spreads in the population is not known in advance. We study such problems from the perspective of online learning, where a decision maker aims to optimize an unknown expensive objective function[53]. At each step, the decision maker commits to an action and receives the objective value for that action. For instance, a policymaker may implement a disease control policy[273,226] for a given time period and observe the number of subsequent infections. This information allows the decision maker to update their knowledge of the unknown function. The goal is to obtain low cumulative regret, which measures the difference in objective value between the actions that were taken and the true (unknown) optimum.

This problem has been well-studied in optimization and machine learning. When a parametric form is not available for the objective (as is often the case with complex systems that are difficult to model analytically), a common approach uses a Gaussian process (GP) as a nonparametric prior over smooth functions. This Bayesian approach allows the decision maker to form a posterior distribution over the unknown function's values. Consequently, the GP-UCB algorithm, which iteratively selects the point with the highest upper confidence bound according to the posterior, achieves a no-regret guarantee[292].

While GP-UCB and similar techniques[73,331] have seen a great deal of interest in the purely black-box setting, many physical or social systems naturally admit an intermediate level of feedback. This is because the system is composed of multiple interacting components, each of which can be measured individually. For instance, disease spread in a population is a product of the interactions between individuals in different demographic groups or locations[341], and policymakers often have access to estimates of the prevalence of infected individuals within each subgroup[79,339]. The true objective (total infections) is the sum of infections across the subgroups. Similarly, climate systems involve the interactions of many different variables (heat, wind, humidity, etc.) which can be sensed individually then combined in a nonlinear fashion to produce outputs of interest (e.g., an individ-

ual's risk of heat stroke)[293]. Prior work has studied the benefits of using additive models[161]. However, they only examine the special case where the target function decomposes into a sum of lower-dimensional functions. Motivated by applications such as flu prevention, we consider the more general setting where the subcomponents are full-dimensional and may be composed nonlinearly to produce the target. This general perspective is necessary to capture common policy settings which may involve intermediate observables from simulation or domain knowledge.

However, to our knowledge, no prior work studies the challenge of integrating such decomposed feedback in online decision making. Our first contribution is to remedy this gap by proposing a *decomposed GP-UCB* algorithm (D-GPUCB). D-GPUCB uses a separate GP to model each individual measurable quantity and then combines the estimates to produce a posterior over the final objective. Our second contribution is a theoretical no-regret guarantee for D-GPUCB, ensuring that its decisions are asymptotically optimal. Third, we prove that the posterior variance at each step must be less than the posterior variance of directly using a GP to model the final objective. This formally demonstrates that more detailed modeling reduces predictive uncertainty. Finally, we conduct experiments in two domains using real-world data: flu prevention and heat sensing. In each case, D-GPUCB achieves substantially lower cumulative regret than previous approaches.

7.2 Preliminaries

7.2.1 Noisy Black-box Optimization

Given an unknown black-box function $f : \mathcal{X} \to \mathbb{R}$ where $\mathcal{X} \subset \mathbb{R}^n$, a learner is able to select an input $\boldsymbol{x} \in \mathcal{X}$ and access the function to see the outcome $f(\boldsymbol{x})$ – this encompasses one evaluation. Gaussian process regression[264] is a non-parametric method to learn the target function using Bayesian methods[156,288]. It assumes that the target function is an outcome of a Gaussian process with given kernel $k(\boldsymbol{x}, \boldsymbol{x}')$ (covariance function). Gaussian process regression is commonly used and

only requires an assumption on the function smoothness. Moreover, Gaussian process regression can handle observation error. It allows the observation at point $\boldsymbol{x}_t$ to be noisy: $y_t = f(\boldsymbol{x}_t) + \varepsilon_t$, where $\varepsilon_t \sim N(0, \sigma^2 I)$.

7.2.2 Decomposition

In this paper, we consider a modification to the Gaussian process regression process. Suppose we have some prior knowledge of the unknown reward function $f(\boldsymbol{x})$ such that we can write the unknown function as a combination of known and unknown subfunctions:

Definition 7 (Linear Decomposition)**.**

$$f(\boldsymbol{x}) = \sum_{j=1}^{J} g_j(\boldsymbol{x}) f_j(\boldsymbol{x}) \tag{7.1}$$

where $f_j, g_j : \mathbb{R}^n \to \mathbb{R}$.

Here $g_j(\boldsymbol{x})$ are known, deterministic functions, but $f_j(\boldsymbol{x})$ are unknown functions that generate noisy observations. For example, in the flu prevention case, the total infected population can be written as the summation of the infected population at each age[79]. Given treatment policy $\boldsymbol{x}$, we can use $f_j(\boldsymbol{x})$ to represent the unknown infected population at age group j with its known, deterministic weighted function $g_j(\boldsymbol{x}) = 1$. Therefore, the total infected population $f(\boldsymbol{x})$ can be simply expressed as $\sum_{j=1}^{J} f_j(\boldsymbol{x})$.

Interestingly, any deterministic linear composition of outcomes of Gaussian processes is still an outcome of Gaussian process. That means if all of the f_j are generated from Gaussian processes, then the entire function f can also be written as an outcome of another Gaussian process.

Next, we generalize this definition to the non-linear case, which we call a general decomposition:

Definition 8 (General Decomposition)**.**

$$f(\boldsymbol{x}) = g(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_J(\boldsymbol{x})) \tag{7.2}$$

The function g can be any deterministic function (e.g. polynomial, neural network). Unfortunately, a non-linear composition of Gaussian processes may not be a Gaussian process, so we cannot guarantee function f to be an outcome of a Gaussian process. We will cover the result of linear decomposition first and then generalize it to the cases with general decomposition.

7.2.3 Gaussian Process Regression

Although Gaussian process regression does not require rigid parametric assumptions, a certain degree of smoothness is still needed to ensure its guarantee of no-regret. We can model f as a sample from a GP: a collection of random variables, one for each $\boldsymbol{x} \in \mathcal{X}$. A GP$(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$ is specified by its mean function $\mu(x) = E[f(\boldsymbol{x})]$ and covariance function $k(\boldsymbol{x}, \boldsymbol{x}') = E[(f(\boldsymbol{x}) - \mu(\boldsymbol{x}))(f(\boldsymbol{x}') - \mu(\boldsymbol{x}'))]$. For GPs not conditioned on any prior, we assume that $\mu(\boldsymbol{x}) \equiv 0$. We further assume bounded variance $k(\boldsymbol{x}, \boldsymbol{x}) \leq 1$. This covariance function encodes the smoothness condition of the target function f drawn from the GP.

For a noisy sample $\boldsymbol{y}_T = [y_1, ..., y_T]^\top$ at points $A_T = \{\boldsymbol{x}_t\}_{t \in [T]}, y_t = f(\boldsymbol{x}_t) + \varepsilon_t \, \forall t \in [T]$ with $\varepsilon_t \sim N(0, \sigma^2(\boldsymbol{x}_t))$ Gaussian noise with variance $\sigma^2(\boldsymbol{x}_t)$, the posterior over f is still a Gaussian process with posterior mean $\mu_T(\boldsymbol{x})$, covariance $k_T(\boldsymbol{x}, \boldsymbol{x}')$ and variance $\sigma_T^2(\boldsymbol{x})$:

$$\mu_T(\boldsymbol{x}) = \boldsymbol{k}_T(\boldsymbol{x})^\top \boldsymbol{K}_T^{-1} \boldsymbol{k}_T(\boldsymbol{x}'), \tag{7.3}$$

$$k_T(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{k}_T(\boldsymbol{x})^\top \boldsymbol{K}_T^{-1} \boldsymbol{k}(\boldsymbol{x}'), \tag{7.4}$$

$$\sigma_T^2(\boldsymbol{x}) = k_T(\boldsymbol{x}, \boldsymbol{x}') \tag{7.5}$$

where $\boldsymbol{k}_T(\boldsymbol{x}) = [k(\boldsymbol{x}_1, \boldsymbol{x}), \ldots, k(\boldsymbol{x}_T, \boldsymbol{x})]^\top$, and $\boldsymbol{K}_T$ is the positive definite kernel matrix $[k(\boldsymbol{x}, \boldsymbol{x}')]_{\boldsymbol{x},\boldsymbol{x}' \in A_T} + \mathrm{diag}([\sigma^2(\boldsymbol{x}_t)]_{t \in [T]})$.

Algorithm 5: GP Regression

1 **Input:** kernel $k(\boldsymbol{x}, \boldsymbol{x}')$, noise function $\sigma(\boldsymbol{x})$, and previous samples $\{(\boldsymbol{x}_t, y_t)\}_{t \in [T]}$
2 **Return:** $k_T(\boldsymbol{x}, \boldsymbol{x}'), \mu_T(\boldsymbol{x}), \sigma_T^2(\boldsymbol{x})$

7.2.4 Bandit Problem with Decomposed Feedback

Considering the output value of the target function as the learner's reward (penalty), the goal is to learn the unknown underlying function f while optimizing the cumulative reward. This is usually known as an online learning or multi-arm bandit problem[24]. In this paper, given the knowledge of deterministic decomposition function g (Definition 7 or Definition 8), in each round t, the learner chooses an input $\boldsymbol{x}_t \in \mathcal{X}$ and observes the value of each unknown decomposed function f_j perturbed by a noise: $y_{j,t} = f_j(\boldsymbol{x}_t) + \varepsilon_{j,t}, \varepsilon_{j,t} \sim N(0, \sigma_j^2)\, \forall j \in [J]$. At the same time, the learner receives the composed reward from this input $\boldsymbol{x}_t$, which is $y_t = g(y_{1,t}, y_{2,t}, \ldots, y_{J,t}) = f(\boldsymbol{x}_t) + \varepsilon_t$ where ε_t is an aggregated noise. The goal is to maximize the sum of noise-free rewards $\sum_{t=1}^{T} f(\boldsymbol{x}_t)$, which is equivalent to minimizing the cumulative regret $R_T = \sum_{t=1}^{T} r_t = \sum_{t=1}^{T} f(\boldsymbol{x}^*) - f(\boldsymbol{x}_t)$, where $\boldsymbol{x}^* = \arg\max_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$ and individual regret $r_t = f(\boldsymbol{x}^*) - f(\boldsymbol{x}_t)$.

This decomposed feedback is related to the semi-bandit setting, where a decision is chosen from a combinatorial set and feedback is received about individual elements of the decision[233,233]. Our work is similar in that we consider an intermediate feedback model which gives the decision maker access to decomposed feedback about the underlying function. However, in our setting a single point is chosen from a continuous set, rather than multiple items from a discrete one. Additional feedback is received about components of the objective function, not the items chosen. Hence, the technical challenges are quite different.

7.3 Problem Statement and Background

Using the flu prevention as an example, a policymaker will implement a yearly disease control policy and observe the number of subsequent infections. A policy is an input $\boldsymbol{x}_t \in \mathbb{R}^n$, where each entry $x_{t,i}$ denotes the extent to vaccinate the infected people in age group i. For example, if the government spends more effort $x_{t,i}$ in group i, then the people in this group will be more likely to get a flu shot.

Given the decomposition assumption and samples (previous policies) at points $\boldsymbol{x}_t \; \forall t \in [T]$, including all the function values $f(\boldsymbol{x}_t)$ (total infected population) and decomposed function values $f_j(\boldsymbol{x}_t)$ (infected population in group j), the learner attempts to learn the function f while simultaneously minimizing regret. Therefore, we have two main challenges: (i) how best to approximate the reward function using the decomposed feedback and decomposition (non-parametric approximation), and (ii) how to use this estimation to most effectively reduce the average regret (bandit problem).

7.3.1 Regression: Non-parametric Approximation

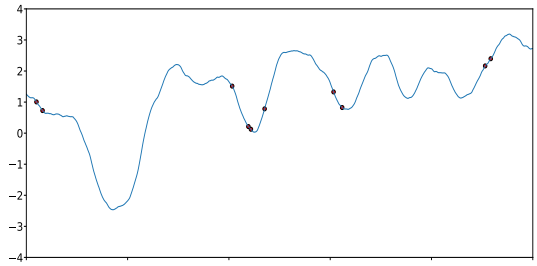
Our first aim is to fully utilize the decomposed problem structure to get a better approximation of $f(x)$. The goal is to learn the underlying disease pattern faster by using the decomposed problem structure. Given the linear decomposition assumption that $f(\boldsymbol{x}) = \sum_{j=1}^{J} g_j(\boldsymbol{x}) f_j(\boldsymbol{x})$ and noisy samples at points $\{\boldsymbol{x}_t\}_{t \in [T]}$, the learner can observe the outcome of each decomposed function $f_j(\boldsymbol{x}_t)$ at each sample point $\boldsymbol{x}_t \; \forall t \in [T]$. Our goal is to provide a Bayesian update to the unknown function which fully utilizes the learner's knowledge of the decomposition.

7.3.2 Bandit Problem: Minimizing Regret

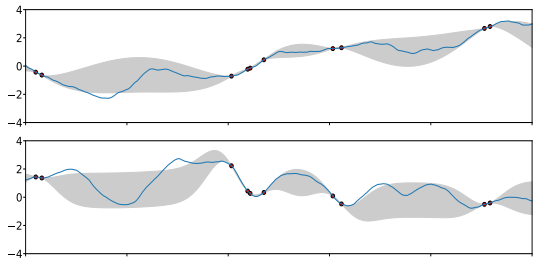
In the flu example, each annual flu-awareness campaign is constrained by a budget, and we assume policymaker does not know the underlying disease spread pattern. At the beginning of each year, the policymaker chooses a new campaign policy based on the previous years' results and observes the outcome of this new policy. The goal is to minimize the cumulative regret (all additional infections in prior years) while learning the underlying unknown function (disease pattern).

We will show how a decomposed GP regression, with a GP-UCB algorithm, can be used to address these challenges.

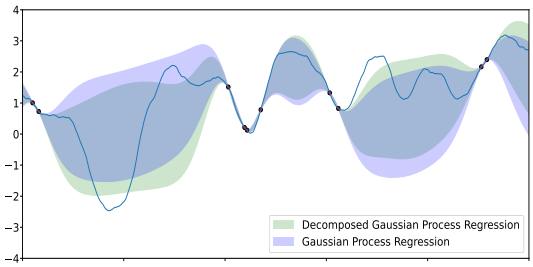
7.4 Decomposed Gaussian Process Regression



(a) Target function $f = f_1 + f_2$ and its sampled values as the observations.



(b) Decomposed functions f_1, f_2 and their posteriors of GP regression.



(c) The posteriors of decomposed GP regression and GP regression

Figure 7.1: Illustration of decomposed GP regression (Algorithm 6) and comparison with GP regression (Algorithm 5). Decomposed GP regression shows a smaller average variance (0.878 v.s. 0.943) and a closer estimation to f.

First, we propose a decomposed GP regression (Algorithm 6). The idea behind decomposed GP regression is as follows: given the linear decomposition assumption (Definition 7), run Gaussian process regression for each $f_j(\boldsymbol{x})$ individually, and get the aggregated approximation by $f(\boldsymbol{x}) = \sum_{j=1}^{J} g_j(\boldsymbol{x}) f_j(\boldsymbol{x})$ (illustrated in Figure 7.1).

Assuming we have T previous samples with input $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T$ and the noisy outcome of each individual function $\boldsymbol{y}_{j,t} = f_j(\boldsymbol{x}_t) + \varepsilon_{j,t}\ \forall j \in [J], t \in [T]$, where $\varepsilon_{j,t} \sim N(0, \sigma_j^2)$, the outcome of the target function $f(x)$ can be computed as $y_t = \sum_{j=1}^{J} g_j(\boldsymbol{x}_t) y_{j,t}$. Further assume the function $f_j(\boldsymbol{x})$ is an outcome of $GP(0, k_j)\ \forall j$. Therefore the entire function f is also an outcome of $GP(0, k)$ where $k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{j=1}^{J} g_j(\boldsymbol{x}) k_j(\boldsymbol{x}, \boldsymbol{x}') g_j(\boldsymbol{x}')$.

We are going to compare two ways to approximate the function $f(\boldsymbol{x})$ using existing samples. (i) Directly use Algorithm 5 with the composed kernel $k(\boldsymbol{x}, \boldsymbol{x}')$ and noisy samples $\{(\boldsymbol{x}_t, y_t)\}_{t \in [T]}$ – the typical GP regression process. (ii) For each $j \in [J]$, first run Algorithm 5 with kernel $k_j(\boldsymbol{x}, \boldsymbol{x}')$ and noisy samples $\{(\boldsymbol{x}_t, y_{j,t})\}_{t \in [T]}$. Then compose the outcomes with the deterministic weighted function $g_j(\boldsymbol{x})$ to get $f(\boldsymbol{x})$. This is shown in Algorithm 6.

Algorithm 6: Decomposed GP Regression

Input: kernel functions $k_j(\boldsymbol{x}, \boldsymbol{x}')$ to each $f_j(\boldsymbol{x})$ and previous samples $(\boldsymbol{x}_t, y_{j,t})\ \forall j \in [J], t \in [T]$

for $j = 1, 2 \ldots, J$ **do**

  Let $\mu_{j,T}(\boldsymbol{x}), k_{j,T}(\boldsymbol{x}, \boldsymbol{x}'), \sigma_{j,T}^2(\boldsymbol{x})$ be the output of GP regression with $k_j(\boldsymbol{x}, \boldsymbol{x}')$ and $(\boldsymbol{x}_t, y_{j,t})$.

Return: $k_T(\boldsymbol{x}, \boldsymbol{x}') = \sum_{j=1}^{J} g_j^2(\boldsymbol{x}) k_{j,T}(\boldsymbol{x}, \boldsymbol{x}') g_j^2(\boldsymbol{x}'), \mu_T(\boldsymbol{x}) = \sum_{j=1}^{J} g_j(\boldsymbol{x}) \mu_{j,T}(\boldsymbol{x})$, $\sigma_T^2(\boldsymbol{x}) = k_T(\boldsymbol{x}, \boldsymbol{x})$

In order to analytically compare Gaussian process regression (Algorithm 5) and decomposed Gaussian process regression (Algorithm 6), we are going to compute the variance (uncertainty) returned by both algorithms. We will show that the latter variance is smaller than the former. Proofs

are in the Appendix for brevity.

Proposition 4. *The variance returned by Algorithm 5 is*

$$\sigma^2_{T,entire}(\boldsymbol{x}) = k(\boldsymbol{x},\boldsymbol{x}) - \sum_{i,j} \boldsymbol{z}_i^\top (\sum_l \boldsymbol{D}_l \boldsymbol{K}_{l,T} \boldsymbol{D}_l)^{-1} \boldsymbol{z}_j \tag{7.6}$$

where $\boldsymbol{D}_j = diag([g_j(\boldsymbol{x}_1), ..., g_j(\boldsymbol{x}_T)])$ *and* $\boldsymbol{z}_i = \boldsymbol{D}_i \boldsymbol{k}_{j,T}(\boldsymbol{x}) g_j(\boldsymbol{x}) \in \mathbb{R}^T$.

Proposition 5. *The variance returned by Algorithm 6 is*

$$\sigma^2_{T,decomp}(\boldsymbol{x}) = k(\boldsymbol{x},\boldsymbol{x}) - \sum_l \boldsymbol{z}_l^\top (\boldsymbol{D}_l \boldsymbol{K}_{l,T} \boldsymbol{D}_l)^{-1} \boldsymbol{z}_l \tag{7.7}$$

In order that our approach has lower variance, we first recall the matrix-fractional function and its convex property.

Lemma 1. *Matrix-fractional function* $h(\boldsymbol{X},\boldsymbol{y}) = \boldsymbol{y}^\top \boldsymbol{X}^{-1} \boldsymbol{y}$ *is defined and also convex on* $domf = \{(\boldsymbol{X},\boldsymbol{y}) \in \mathbf{S}_+^T \times \mathbb{R}^T\}$.

Now we are ready to compare the variance provided by Proposition 4 and Proposition 5.

Theorem 8. *The variance provided by decomposed Gaussian process regression (Algorithm 6) is less than or equal to the variance provided by Gaussian process regression (Algorithm 5), which implies the uncertainty by using decomposed Gaussian process regression is smaller.*

Proof sketch. In order to compare the variance given by Proposition 4 and Proposition 5, we calculate the difference of Equation 7.6 and Equation 7.7. Their difference can be rearranged as a Jensen inequality with the form of Matrix-fractional function (Lemma 1), which turns out to be convex. By Jensen inequality, their difference is non-negative, which implies the variance given by decomposed GP regression is no greater than the variance given by GP regression. □

Theorem 8 implies that decomposed GP regression provides a posterior with smaller variance, which could be considered the uncertainty of the approximation. In fact, the posterior belief after the GP regression is still a Gaussian process, which implies the underlying target function is characterized by a joint Gaussian distribution, where a smaller variance directly implies a more concentrated Gaussian distribution, leading to less uncertainty and smaller root-mean-squared error. Intuitively, this is due to Algorithm 6 adopts the decomposition knowledge but Algorithm 5 does not. This contribution for handling decomposition in the GP regression context is very general and can be applied to many problems. We will show some applications of this idea in the following sections, focusing first on how a linear and generalized decompositions can be used to augment the GP-UCB algorithm for multi-armed bandit problems.

7.5 Decomposed GP-UCB Algorithm

The goal of a traditional bandit problem is to optimize the objective function $f(\boldsymbol{x})$ by minimizing the regret. However, in our bandit problem with decomposed feedback, the learner is able to access samples of individual functions $f_j(\boldsymbol{x})$. We first consider a linear decomposition $f(\boldsymbol{x}) = \sum_{j=1}^{J} g_j(\boldsymbol{x}) f_j(\boldsymbol{x})$.

Srinivas et al. proposed the GP-UCB algorithm for classic bandit problems and proved that it is a no-regret algorithm that can efficiently achieve the global optimal objective value. A natural question arises: can we apply our decomposed GP regression (Algorithm 6) and also achieve the no-regret property? This leads to our second contribution: the decomposed GP-UCB algorithm, which uses decomposed GP regression when decomposed feedback is accessible. This algorithm can incorporate the decomposed feedback (the outcomes of decomposed function f_j), achieve a better approximation at each iteration while maintaining the no-regret property, and converge to a globally optimal value.

Theorem 9. *Let $\delta \in (0,1)$ and $\beta_t = 2\log(|\mathcal{X}|t^2\pi^2/6\delta)$. Running decomposed GP-UCB (Algorithm*

Algorithm 7: Decomposed GP-UCB

Input: Input space $\mathcal{X}$; GP priors $\mu_{j,0}, \sigma_{j,0}, k_j \; \forall j \in [J]$
for $t = 1,2,\ldots$ **do**
  Compute all mean $\mu_{j,t-1}$ and variance $\sigma^2_{j,t-1} \forall j$
  $\mu_{t-1}(\boldsymbol{x}) = \sum_{j=1}^{J} g_j(\boldsymbol{x}) \mu_{j,t-1}(\boldsymbol{x})$
  $\sigma^2_{t-1}(\boldsymbol{x}) = \sum_{j=1}^{J} g_j^2(\boldsymbol{x}) \sigma^2_{j,t-1}$
  Choose $\boldsymbol{x}_t = \arg\max_{\boldsymbol{x} \in \mathcal{X}} \mu_{t-1}(\boldsymbol{x}) + \sqrt{\beta_t} \sigma_{t-1}(\boldsymbol{x})$
  Sample $y_{j,t} = f_j(\boldsymbol{x}_t) \; \forall j \in [J]$
  Perform Bayesian update to obtain $\mu_{j,t}, \sigma_{j,t} \; \forall j \in [J]$

7) for a composed sample $f(\boldsymbol{x}) = \sum_{j=1} g_j(\boldsymbol{x}) f_j(\boldsymbol{x})$ *with bounded variance* $k_j(\boldsymbol{x}, \boldsymbol{x}) \leq 1$ *and each* $f_j \sim GP(0, k_j(\boldsymbol{x}, \boldsymbol{x}'))$*, we obtain a regret bound of* $\mathcal{O}(\sqrt{T \log |\mathcal{X}| \sum_{j=1}^{J} B_j^2 \gamma_{j,T}})$ *with high probability, where* $B_j = \max_{\boldsymbol{x} \in \mathcal{X}} |g_j(\boldsymbol{x})|$*. Precisely,*

$$Pr\Big\{R_T \leq \sqrt{C_1 T \beta_T \sum_{j=1}^{J} B_j^2 \gamma_{j,T}} \; \forall T \geq 1\Big\} \geq 1 - \delta \tag{7.8}$$

where $C_1 = 8/\log(1 + \sigma^{-2})$ *with noise variance* σ^2*.*

We present Algorithm 7, which replaces the Gaussian process regression in GP-UCB with our decomposed Gaussian process regression (Algorithm 6). According to Theorem 8, our algorithm takes advantage of decomposed feedback and provides a more accurate and less uncertain approximation at each iteration. We also provide a regret bound in Theorem 9, which guarantees no-regret property to Algorithm 7.

According to the linear decomposition and the additive and multiplicative properties of kernels, the entire underlying function is still an outcome of GP with a composed kernel $k(\boldsymbol{x}, \boldsymbol{x}') =$

$\sum_{j=1}^{J} g_j(\boldsymbol{x}) k_j(\boldsymbol{x}, \boldsymbol{x}') g_j(\boldsymbol{x}')$, which implies that GP-UCB algorithm can achieve a similar regret bound by normalizing the kernel $k(\boldsymbol{x}, \boldsymbol{x}') \leq \sum_{j=1}^{J} B_j^2 = B^2$. The regret bound of GP-UCB can be given by:

$$Pr\{R_T \leq \sqrt{C_1 T \beta_T B^2 \gamma_{\text{entire},T}} \forall T \geq 1\} \geq 1 - \delta \tag{7.9}$$

where $\gamma_{\text{enitre},T}$ is the upper bound on the information gain $I(y_T; f_T)$ of the composed kernel $k(\boldsymbol{x}, \boldsymbol{x}')$.

But due to Theorem 8, D-GPUCB can achieve a lower variance and more accurate approximation at each iteration, leading to a smaller regret in the bandit setting, which will be shown to empirically perform better in the experiments.

7.5.1 No-Regret Property and Benefits of D-GPUCB

Previously, in order to guarantee a sublinear regret bound to GP-UCB, we require an analytical, sublinear bound $\gamma_{\text{entire},T}$ on the information gain.[292] provided several elegant upper bounds on the information gain of various kernels. However, in practice, it is hard to give an upper bound to a composed kernel $k(\boldsymbol{x}, \boldsymbol{x}')$ and apply the regret bound (Inequality 7.9) provided by GP-UCB in the decomposed context.

Instead, D-GPUCB and the following generalized D-GPUCB provide a clearer expression to the regret bound, where their bounds (Theorem 9, 10) only relate to upper bounds $\gamma_{j,T}$ of the information gain of each kernel $k_j(\boldsymbol{x}, \boldsymbol{x}')$. This resolves the problem of computing an upper bound of a composed kernel. We can use the various sublinear upper bounds of different kernels, which have been widely studied in prior literature[292].

7.5.2 Generalized Decomposed GP-UCB Algorithm

We now consider the general decomposition (Definition 8): $f(\boldsymbol{x}) = g(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_J(\boldsymbol{x}))$. To achieve the no-regret property, we further require the function g to have bounded partial derivatives $|\nabla_j g(\boldsymbol{x})| \leq B_j \, \forall j \in [J]$. This corresponds to the linear decomposition case, where $|\nabla_j g| = |g_j(\boldsymbol{x})| \leq B_j$.

Since, a non-linear composition of Gaussian processes is no longer a Gaussian process, the standard GP-UCB algorithm does not have any guarantees for this setting. However, we show that our approach, which leverages the special structure of the problem, still enjoys a no-regret guarantee:

Algorithm 8: Generalized Decomposed GP-UCB

Input: Input space $\mathcal{X}$; GP priors $\mu_{j,0}, \sigma_{j,0}, k_j \, \forall j \in [J]$
for $t = 1, 2, \ldots$ **do**
  Compute the aggregated mean and variance bound:
  $\mu_{t-1}(\boldsymbol{x}) = g(\mu_{1,t-1}(\boldsymbol{x}), \ldots, \mu_{J,t-1}(\boldsymbol{x}))$
  $\sigma^2_{t-1}(\boldsymbol{x}) = J \sum_{j=1}^{J} B_j^2 \sigma^2_{j,t-1}(\boldsymbol{x})$
  Choose $\boldsymbol{x}_t = \arg\max_{\boldsymbol{x} \in \mathcal{X}} \mu_{t-1}(\boldsymbol{x}) + \sqrt{\beta_t} \sigma_{t-1}(\boldsymbol{x})$
  Sample $y_{j,t} = f_j(\boldsymbol{x}_t) \, \forall j \in [J]$
  Perform Bayesian update to obtain $\mu_{j,t}, \sigma_{j,t} \, \forall j \in [J]$

Theorem 10. *By running generalized decomposed GP-UCB with $\beta_t = 2\log(|\mathcal{X}| J t^2 \pi^2 / 6\delta)$ for a composed sample $f(\boldsymbol{x}) = g(f_1(\boldsymbol{x}), \ldots, f_J(\boldsymbol{x}))$ of GPs with bounded variance $k_j(\boldsymbol{x}, \boldsymbol{x}) \leq 1$ and each $f_j \sim GP(0, k_j(\boldsymbol{x}, \boldsymbol{x}'))$. we obtain a regret bound of $\mathcal{O}(\sqrt{T \log |\mathcal{X}| \sum_{j=1}^{J} B_j^2 \gamma_{j,T}})$ with high probability, where $B_j = \max_{\boldsymbol{x} \in \mathcal{X}} |\nabla_j g(\boldsymbol{x})|$. Precisely,*

$$Pr\Big\{R_T \leq \sqrt{C_1 T \beta_T \sum_{j=1}^{J} B_j^2 \gamma_{j,T}} \, \forall T \geq 1\Big\} \geq 1 - \delta \tag{7.10}$$

where $C_1 = 8/\log(1+\sigma^{-2})$ *with noise variance* σ^2.

The intuition is that so long as each individual function is drawn from a Gaussian process, we can still perform Gaussian process regression on each function individually to get an estimate of each decomposed component. Based on these estimates, we compute the corresponding estimate to the final objective value by combining the decomposed components with the function g. Since the gradient of function g is bounded, we can propagate the uncertainty of each individual approximation to the final objective function, which allows us to get a bound on the total uncertainty. Consequently, we can prove a high-probability bound between our algorithm's posterior distribution and the target function, which enables us to bound the cumulative regret by a similar technique as Theorem 9.

The major difference for general decomposition is that the usual GP-UCB algorithm no longer works here. The underlying unknown function may not be an outcome of Gaussian process. Therefore the GP-UCB algorithm does not have any guarantees for either convergence or the no-regret property. In contrast, D-GPUCB algorithm still works in this general case if the learner is able to attain the decomposed feedback.

Our result greatly enlarges the feasible functional space where GP-UCB can be applied. We have shown that the generalized D-GPUCB preserves the no-regret property even when the underlying function is a composition of Gaussian processes. Given the knowledge of decomposition and decomposed feedback, based on Theorem 10, the functional space that generalized D-GPUCB algorithm can guarantee no-regret is closed under arbitrary bounded-gradient function composition. This leads to a very general functional space, showcasing the contribution of our algorithm.

7.5.3 Continuous Sample Space

All the above theorems are for discrete sample spaces $\mathcal{X}$. However, most real-world scenarios have a continuous space. Srinivas et al. used the discretization technique to reduce the compact and convex continuous sample space to a discrete case by using a larger exploration constant:

$$\beta_t = 2\log(2t^2\pi^2/(3\delta)) + 2d\log(t^2 dbr\sqrt{\log(4da/\delta)})$$

while assuming $\Pr\{\sup_{\boldsymbol{x}\in\mathcal{X}} |\partial f/\partial \boldsymbol{x}_i| > L\} \leq ae^{-(L/b)^2}$. (In the general decomposition case, $\beta_t = 2\log(2Jt^2\pi^2/(3\delta)) + 2d\log(t^2 dbr\sqrt{\log(4da/\delta)})$). All of our proofs directly follow using the same technique. Therefore the no-regret property and regret bound also hold in continuous sample spaces.

7.6 Experiments

In this section, we run several experiments to compare decomposed Gaussian process regression (Algorithm 6), D-GPUCB (Algorithm 7), and generalized D-GPUCB (Algorithm 8). We also test on both discrete sample space and continuous sample space. All of our examples show a promising convergence rate and also improvement against the GP-UCB algorithm, again demonstrating that more detailed modeling reduces the predictive uncertainty.

7.6.1 Decomposed Gaussian Process Regression

For the decomposed Gaussian process regression, we compare the average standard deviation (uncertainty) provided by GP regression (Algorithm 5) and decomposed GP regression (Algorithm 6) over varying number of samples and number of decomposed functions. We use the following three common types of stationary kernel[264]:

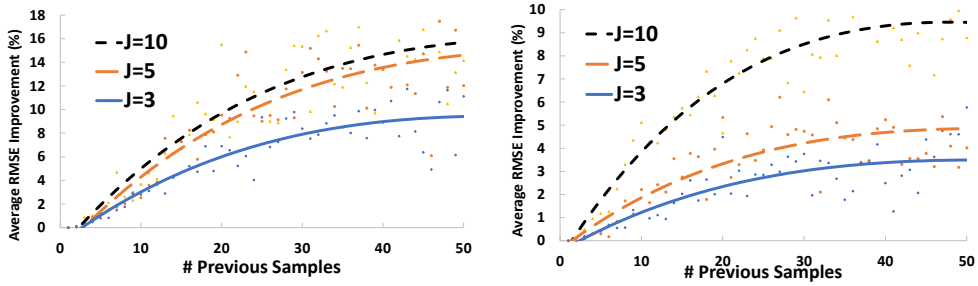
- The Square Exponential kernel is $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-(2l^2)^{-1} \|\boldsymbol{x} - \boldsymbol{x}'\|^2)$, l is a length-scale hyper parameter.

- The Matérn kernel is given by $k(\boldsymbol{x}, \boldsymbol{x}') = (2^{1-\nu}/\Gamma(\nu)) r^{\nu} B_{\nu}(r)$, $r = (\sqrt{2\nu}/l) \|\boldsymbol{x} - \boldsymbol{x}'\|$, where ν controls the smoothness of sample functions and B_{ν} is a modified Bessel function.

- The Rational Quadratic kernel is $k(\boldsymbol{x}, \boldsymbol{x}') = (1 + \|\boldsymbol{x} - \boldsymbol{x}'\|^2 / (2\alpha l^2))^{-\alpha}$. It can be seen as a scale mixture of square exponential kernels with different length-scales.

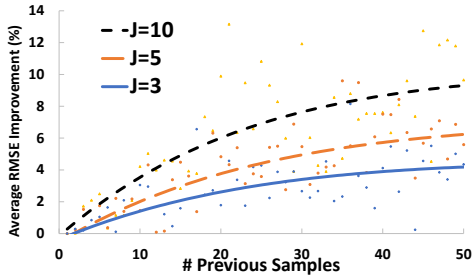
For each kernel category, we first draw J kernels with random hyper-parameters. We then generate a random sample function f_j from each corresponding kernel k_j as the target function, combined with the simplest linear decomposition (Definition 7) with $g_j(\boldsymbol{x}) \equiv 1 \forall j$. For each setting and each $T \leq 50$, we randomly draw T samples as the previous samples and perform both GP regression and decomposed GP regression. We record the average improvement in terms of root-mean-squared error (RMSE) against the underlying target function over 100 independent runs for each setting. We also run experiments on flu domain with square exponential kernel based on real data and SIR model[79], which is illustrated in Figure 7.2(d).

Empirically, our method reduces the RMSE in the model's predictions by 10-15% compared to standard GP regression (without decomposed feedback). This trend holds across kernels, and includes both synthetic data and the flu domain (which uses a real dataset). Such an improvement in predictive accuracy is significant in many real-world domains. For instance, CDC-reported 95% confidence intervals for vaccination-averted flu illnesses for 2015 range from 3.5M-7M and averted medical visits from 1.7M-3.5M. Reducing average error by 10% corresponds to estimates which are tighter by hundreds of thousands of patients, a significant amount in policy terms. These results confirm our theoretical analysis in showing that incorporating decomposed feedback results in more accurate estimation of the unknown function.

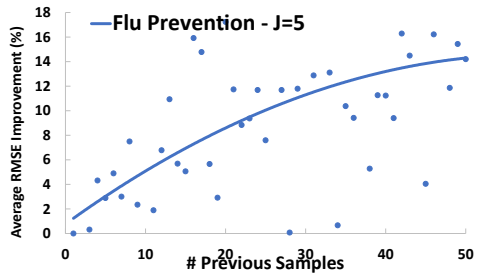


(a) Square exponential kernel

(b) Matern kernel



(c) Rational quadratic kernel



(d) Flu domain

Figure 7.2: Average improvement (with trend line) using decomposed GP regression and GP regression, in RMSE

7.6.2 Comparison between GP-UCB and D-GPUCB

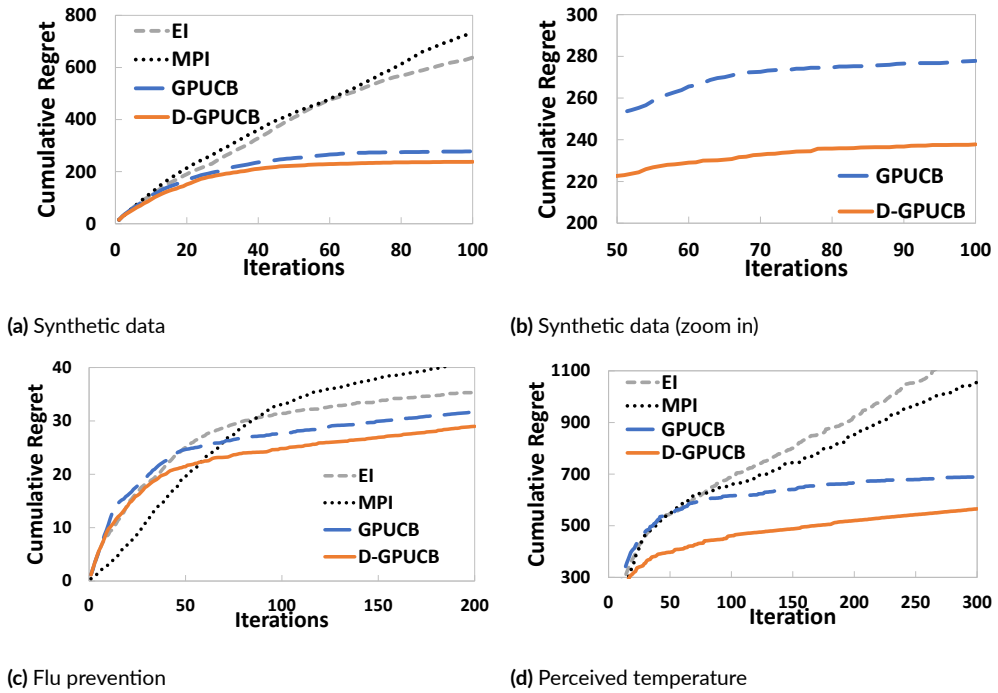
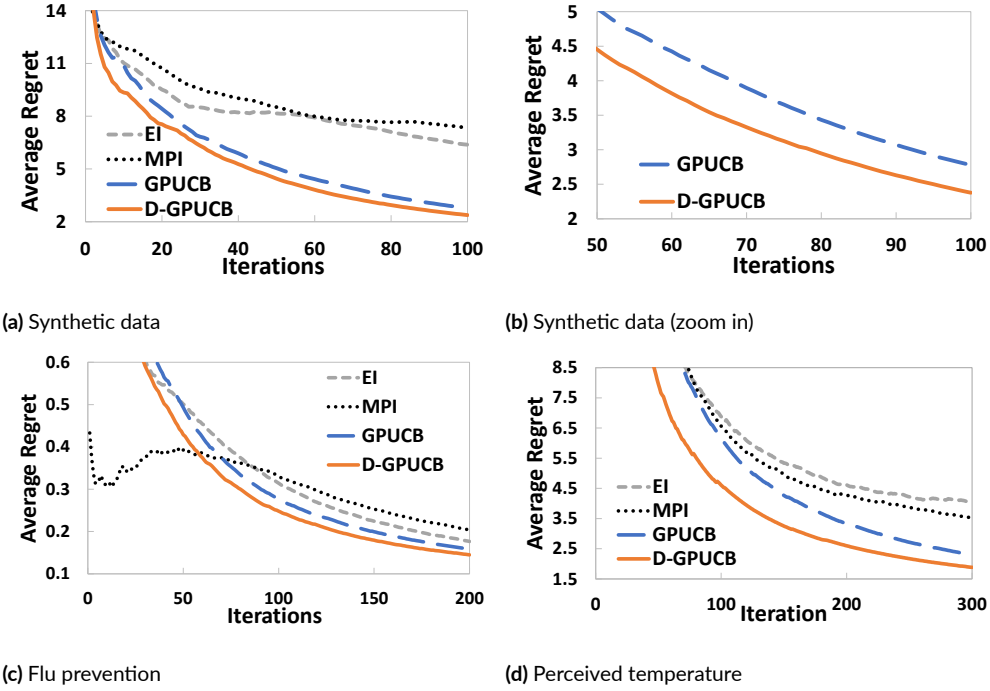


Figure 7.3: Comparison of cumulative regret: D-GPUCB, GP-UCB, and various heuristics on synthetic (a, b) and real data (c, d)

We now move the online setting, to test whether greater predictive accuracy results in improved decision making. We compare our D-GPUCB algorithm and generalized D-GPUCB with GP-UCB, as well as common heuristics such as Expected Improvement (EI)[218] and Most Probable Improvement (MPI)[184]. For all the experiments, we run 30 trials on all algorithms to find the average regret.

Synthetic Data (Linear Decomposition with Discrete Sample Space):

For synthetic data, we randomly draw $J = 10$ square exponential kernels with different hyper-parameters and then sample random functions from these kernels to compose the entire target function. The sample noise is set to be 10^{-4}. The sample space $\mathcal{X} = [0, 1]$ is uniformly discretized into



(a) Synthetic data

(b) Synthetic data (zoom in)

(c) Flu prevention

(d) Perceived temperature

Figure 7.4: Comparison of average regret: D-GPUCB, GP-UCB, and various heuristics on synthetic (a, b) and real data (c, d)

1000 points. We follow the recommendation in [292] to scale down β_t by a factor 5 for both GP-UCB and D-GPUCB algorithm. We run each algorithm for 100 iterations with $\delta = 0.05$ for 30 trials (different kernels and target functions each trial), where the cumulative regrets are shown in Figure 7.3(a), 7.3(b), and average regret in Figure 7.4(a), 7.4(b).

Flu Prevention (Linear Decomposition with Continuous Sample Space):

We consider a flu age-stratified SIR model [79] as our target function. The population is stratified into several age groups: young (0-19), adult (20-49), middle aged (50-64), senior (65-69), elder (70+). The SIR model allows the contact matrix and susceptibility of each age group to vary. Our input here is the vaccination rate $\boldsymbol{x} \in [0,1]^5$ with respect to each age group. Given a vaccination rate $\boldsymbol{x}$, the

SIR model returns the average sick days per person $f(\boldsymbol{x})$ within one year. The model can also return the contribution to the average sick days from each age group j, which we denote as $f_j(\boldsymbol{x})$. Therefore we have $f(\boldsymbol{x}) = \sum_{j=1}^{5} f_j(\boldsymbol{x})$, a linear decomposition. The goal is to find the optimal vaccination policy which minimizes the average sick days subject to budget constraints. Since we do not know the covariance kernel functions in advance, we randomly draw 1000 samples and fit a mixture of square exponential kernel and Matérn kernel by tuning the hyper-parameters. We run all algorithms and compare their cumulative regret in Figure 7.3(c) and average regret in Figure 7.4(c).

Perceived Temperature (General Decomposition with Discrete Sample Space):

The perceived temperature is a combination of actual temperature, humidity, and wind speed. When the actual temperature is high, higher humidity reduces the body's ability to cool itself, resulting a higher perceived temperature; when the actual temperature is low, the air motion accelerates the rate of heat transfer from a human body to the surrounding atmosphere, leading to a lower perceived temperature. All of these are nonlinear function compositions. We use the weather data collected from 2906 sensors in United States provided by OpenWeatherMap. Given an input location $\boldsymbol{x} \in \mathcal{X}$, we can access to the actual temperature $f_1(\boldsymbol{x})$, humidity $f_2(\boldsymbol{x})$, and wind speed $f_3(\boldsymbol{x})$. In each test, we randomly draw one third of the entire data to learn the covariance kernel functions. Then we run generalized D-GPUCB and all the other algorithms on the remaining sensors to find the location with highest perceived temperature. The result is averaged over 30 different tests and is also shown in Figure 7.3(d) and Figure 7.4(d).

Discussion:

In the bandit setting with decomposed feedback, Figure 7.3 shows a $10\% - 20\%$ improvement in cumulative regret for both synthetic (Figure 7.3(a), 7.3(b)) and real data (Figure 7.3(c), 7.3(d)). As

in the regression setting, such improvements are highly significant in policy terms; a 10% reduction in sickness due to flu corresponds to hundreds of thousands of infections averted per year. The benefit to incorporating decomposed feedback is particularly large in the general decomposition case (Figure 7.3(d)), where a single GP is a poor fit to the nonlinearly composed function. Figure 7.4 shows the average regret of each algorithm (as opposed to the cumulative regret). Our algorithm's average regret tends to zero. This allows us to empirically confirm the no-regret guarantee for D-GPUCB in both the linear and general decomposition settings. As with the cumulative regret, D-GPUCB uniformly outperforms the baselines.

7.7 Conclusions

We propose algorithms for nonparametric regression and online learning which exploit the decomposed feedback common in real world sequential decision problems. In the regression setting, we prove that incorporating decomposed feedback improves predictive accuracy (Theorem 8). In the online learning setting, we introduce the D-GPUCB algorithms (Algorithm 7 and Algorithm 8) and prove corresponding no-regret guarantees. We conduct experiments in both real and synthetic domains to investigate the performance of decomposed GP regression, D-GPUCB, and generalized D-GPUCB. All show significant improvement against GP-UCB and other methods that do not consider decomposed feedback, demonstrating the benefit that decision makers can realize by exploiting such information.

8

Online Learning for Restless Bandits*

8.1 Introduction

Restless multi-armed bandits (RMABs)[337] generalize multi-armed bandits by introducing states for each arm. RMABs are commonly used to model sequential scheduling problems with limited resources such as in clinical health[312], online advertising[216], and energy-efficient scheduling[50]. As

*This work is a joint work with Lily Xu with equal contributions.

with stochastic combinatorial bandits[66], the RMAB learner must repeatedly pull K out of N arms at each timestep. Unlike stochastic bandits, the reward distribution of each arm in an RMAB depends on that arm's state, which transitions based on a Markov decision process (MDP) depending on whether the arm is pulled. These problems are called "restless" as arms may change state regardless of whether they are pulled. The reward at each timestep is the sum of rewards across all arms, including arms not acted upon.

Even when the transition dynamics are given, planning an optimal policy for RMABs is PSPACE-hard[244] due to the state-dependent reward and combinatorial action space. To compute an approximate planning solution to RMABs, the *Whittle index policy*[337] defines a "Whittle index" for each arm as an estimate of the future value if acted upon, then acts on the arms with the K largest indices. The Whittle index policy is shown to be asymptotically optimal[334] and is commonly adopted as a scalable solution to RMAB problems[140,160].

However, in many real-world applications of RMABs, transition dynamics are often unknown in advance. The learner must strategically query arms to learn the underlying transition probabilities while simultaneously achieving high reward. Accordingly, in this paper we focus on the challenge of online learning in RMABs with unknown transitions. We focus on the Whittle index policy due to its scalability and consider a fixed-length episodic RMAB setting.

Main contributions We present *UCWhittle*, an upper confidence bound (UCB) algorithm that uses the Whittle index policy to achieve the first sublinear frequentist regret guarantee for RMABs. Our algorithm maintains confidence bounds for every transition probability across all arms based on prior observations. Using these bounds, we define a bilinear program to solve for optimistic transition probabilities — the transition probabilities that yield the highest future reward. These optimistic transition probabilities enable us to compute an *optimistic Whittle index* for each arm to inform a Whittle index policy. Our UCWhittle algorithm leverages the structure of RMABs

and the Whittle index solution to decompose the policy across individual arms, greatly reducing the computational cost of finding an optimistic solution compared to other UCB-based solutions[25,148].

Theoretically, we analyze the frequentist regret of UCWhittle. The *frequentist regret* is the worst-case regret incurred from unknown transition dynamics; in contrast, the *Bayesian regret* is the regret averaged over all possible transitions from a prior distribution. In this paper, we define *regret* in terms of the relaxed Lagrangian of the RMAB — to make the objective tractable — which upper bounds the primal RMAB problem. We show that UCWhittle achieves sublinear frequentist regret $O(H\sqrt{T \log T})$ where T is the number of episodes of interaction with the RMAB instance and H is a sufficiently large per-episode time horizon. Our result extends the analysis of Bayesian regret in RMABs[159] to frequentist regret by removing the need to assume a prior distribution. Finally, we evaluate UCWhittle against other online RMAB approaches on real maternal and child healthcare data[211] and two synthetic settings, showing that UCWhittle achieves lower frequentist regret empirically as well.

8.2 Related Work

Offline planning for RMABs When the transition dynamics are given, an RMAB is an optimization problem in a sequential setting. Computing the optimal policy in RMABs is PSPACE-hard[244] due to the state-dependent reward distribution and combinatorial action space. The Whittle index policy[337] approximately solves the planning problem by estimating the value of each arm state. The indexability condition[6,328] guarantees asymptotic optimality[334] of the Whittle index policy with an infinite time horizon. Nakhleh et al.[231] use deep reinforcement learning to estimate Whittle indices for episodic finite-horizon RMABs, which requires the environment to be differentiable and transitions known.

Online learning for RMABs When the transition dynamics are unknown, an RMAB becomes an online learning problem in which the learner must simultaneously learn the transition probabilities (exploration) and execute high-reward actions (exploitation), with the objective of minimizing regret with respect to a chosen benchmark. Dai et al. [78] achieve a regret bound of $O(\log T)$ benchmarked against an optimal policy from a finite number of potential policies. Xiong et al. [347] use a Lagrangian relaxation and index-based algorithm, but require access to an offline simulator to generate samples for any given state–action pair. Tekin & Liu [300] define a weaker benchmark of the best single-action policy — the optimal policy that continues to play the same arm — and use a UCB-based algorithm to achieve $O(\log T)$ frequentist regret.

Recent works introduce oracle-based policies for the non-combinatorial setting in which the learner pulls a single arm in each round, receiving bandit feedback and observing only the state of the pulled arm. Jung & Tewari [159] use a Thompson sampling–based algorithm which achieves a Bayesian regret bound $O(\sqrt{T \log T})$ under a given prior distribution. Wang et al. [329] use separate exploration and exploitation phases to achieve frequentist regret $O(T^{2/3})$. These works assume some policy oracle is given, thus benchmark regret with the policy given by the oracle with knowledge of the true transitions. In contrast to the meta-algorithms they propose, *we design an optimal approach custom-tailored to one specific oracle — based on the Whittle index policy — which enables us to achieve a tighter frequentist regret bound of* $O(H\sqrt{T \log T})$ with a constant horizon H.

Online reinforcement learning RMABs are a special case of Markov decision processes (MDPs) with combinatorial state and action spaces. Q-learning algorithms are popular for solving large MDPs and have been applied to standard binary-action RMABs [27,110,48] and extended to the multi-action setting [169]. However, these works do not provide regret guarantees. Significant work has explored online learning for stochastic multi-armed bandits [233,143,107,30,348], but these do not allow arms to change state.

Some papers study online reinforcement learning by using the optimal policy as the benchmark to bound regret in MDPs[25,148] and RMABs[242]. These works use UCB-based algorithms (UCRL and UCRL2) to obtain a regret of $O(\sqrt{T \log T})$. However, evaluating regret with respect to the optimal policy requires computing the optimal solution to the RMAB problem, which is intractable due to the combinatorial space and action spaces. To overcome this difficulty, we restrict the benchmark for computing regret to the class of Whittle index threshold policies, and leverage the weak decomposability of the Whittle index threshold policy to establish a new regret bound.

Frequentist versus Bayesian regret The regret definition that we consider is *frequentist* regret, measuring worst-case regret under unknown transition probabilities. The other regret notion is Bayesian regret: the expected regret over a prior distribution over possible transition functions. Bayesian regret, such as from Thompson sampling–based methods, relies on a prior and does not provide worst-case guarantees[159,158].

8.3 Restless Bandits and Whittle Index Policy

An instance of a restless multi-armed bandit problem is composed of a set of N arms. Each arm $i \in [N]$ is modeled as an independent Markov decision process (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, R, P_i)$. The state space $\mathcal{S}$, action space $\mathcal{A}$, and reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ are shared across arms; the transition probability $P_i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ may be unique per arm i.

We denote the state of the RMAB instance at timestep $h \in \mathbb{N}$ by $\boldsymbol{s}_h \in \mathcal{S}^N$, where $s_{h,i}$ denotes the state of arm $i \in [N]$. We assume the state is fully observable. The initial state is given by $\boldsymbol{s}_1 = \boldsymbol{s}_{\text{init}} \in \mathcal{S}^N$. The action (a set of "arm pulls") at time h is denoted by a binary vector $\boldsymbol{a}_h \in \mathcal{A}^N = \{0, 1\}^N$ and is constrained by budget K such that $\sum_{i \in [N]} a_{h,i} \leq K$.

After taking action $a_{h,i}$ on arm i, the state $s_{h,i}$ transitions to the next state $s_{h+1,i}$ with transition probability $P_i(s_{h,i}, a_{h,i}, s_{h+1,i}) \in [0, 1]$. We denote the set of all transition probabilities by $\boldsymbol{P} =$

$[P_i]_{i\in[N]}$. The learner receives reward $R(s_{h,i}, a_{h,i})$ from each arm i (including those not acted upon) at every timestep h; we assume the reward function R is known.

The learner's actions are described by a deterministic policy $\pi : \mathcal{S}^N \to \mathcal{A}^N$ which maps a given state $\boldsymbol{s} \in \mathcal{S}^N$ to an action $\boldsymbol{a} \in \mathcal{A}^N$. The learner's goal is to optimize the total discounted reward, with discount factor $\gamma \in (0, 1)$:

$$\begin{aligned} \max_{\pi} \quad & \mathop{\mathbb{E}}_{(\boldsymbol{s},\boldsymbol{a})\sim(\boldsymbol{P},\pi)} \sum_{h\in\mathbb{N}} \gamma^{h-1} \sum_{i\in[N]} R(s_{h,i}, a_{h,i}) \\ \text{s.t.} \quad & \sum_{i\in[N]} (\pi(\boldsymbol{s}))_i \leq K \quad \forall \boldsymbol{s} \in \mathcal{S}^N \end{aligned} \tag{8.1}$$

where $\boldsymbol{s} \sim \boldsymbol{P}$ indicates $s_{h,i} \sim P_i(\cdot \mid s_{h-1,i}, \pi_i(\boldsymbol{s}_{h-1}))$ and $\boldsymbol{a} \sim \pi$ indicates $a_i \sim \pi_i(\boldsymbol{s})$.

8.3.1 Lagrangian Relaxation

Equation 8.1 is intractable to evaluate over all possible policies, thus a poor candidate objective for evaluating online learning performance. Instead, we relax the constraints to use the Lagrangian as the evaluation metric:

$$U_{\pi}^{\boldsymbol{P},\lambda}(\boldsymbol{s}_1) := \mathop{\mathbb{E}}_{(\boldsymbol{s},\boldsymbol{a})\sim(\boldsymbol{P},\pi)} \sum_{h\in\mathbb{N}} \gamma^{h-1} \left(\sum_{i\in[N]} R(s_{h,i}, a_{h,i}) - \lambda \left(\sum_{i\in[N]} (\pi(\boldsymbol{s}_h))_i - K \right) \right) \tag{8.2}$$

which also considers actions that exceed the budget constraint, subject to a given penalty λ. The optimal value of Equation 8.2, which we denote $U_{\star}^{\boldsymbol{P},\lambda}$, is always an upper bound to Equation 8.1. Therefore, we solve Equation 8.2 for candidate penalty values λ and find the infimum $\lambda^{\star} = \arg\min_{\lambda} U_{\star}^{\boldsymbol{P},\lambda}$ afterward.

8.3.2 Whittle Index and Threshold Policy

Relaxing the budget constraint enables us to decompose the combinatorial policy into a set of N independent policies for each arm. The decoupled policy yields $\pi(\boldsymbol{s}) = [\pi_i(\boldsymbol{s}_i)]_{i\in[N]}$, where each arm policy $\pi_i : \mathcal{S} \to \mathcal{A}$ specifies the action for arm i at state s_i. The value function is then:

$$V_{\pi_i}^{P_i,\lambda}(s_{1,i}) := \mathop{\mathbb{E}}_{(s_{1,i},a_{1,i},s_{2,i},a_{2,i},\ldots)\sim(P_i,\pi_i)} \sum_{h\in\mathbb{N}} \gamma^{h-1}\left(R(s_{h,i},a_{h,i}) - \lambda\left(\pi_i(s_{h,i}) - K\right)\right). \tag{8.3}$$

Equation 8.3 can be interpreted as adding a penalty λ to the pulling action $a = 1$, which motivates the definition of Whittle index[337] as the smallest penalty for an arm such that pulling that arm is as good as not pulling it:

Definition 9. *Given transition probabilities P_i and state s_i, the* Whittle index *W_i of arm i is defined as:*

$$W_i(P_i, s_i) = \inf_{m_i}\{m_i : Q^{m_i}(s_i, 0) = Q^{m_i}(s_i, 1)\} \tag{8.4}$$

where the Q-function $Q^{m_i}(s_i, a_i)$ and value-function $V^{m_i}(s_i)$ are the solutions to the Bellman equation with penalty m_i for pulling action $a_i = 1$:

$$Q^{m_i}(s,a) = -m_i a + R(s,a) + \gamma \sum_{s'\in\mathcal{S}} P_i(s,a,s')V^{m_i}(s')$$

$$V^{m_i}(s) = \max_{a\in\mathcal{A}} Q^{m_i}(s,a)\,.$$

When the Whittle index $W_i(P_i, s_i)$ for an arm is higher than the chosen global penalty λ — that is, $m_i > \lambda$ — the optimal policy for Equation 8.3 is to pull that arm, i.e., $\pi_i(s_i) = 1$. We denote the Whittle indices of all arms and all states by $W(\boldsymbol{P}) = [W_i(P_i, s_i)]_{i\in[N],s_i\in\mathcal{S}} \in \mathbb{R}^{N\times|\mathcal{S}|}$.

Definition 10 (Whittle index threshold policy)**.** *Given a chosen global penalty λ and the Whittle indices $W(\boldsymbol{P})$ computed from transitions $\boldsymbol{P}$, the threshold policy is defined by:*

$$\pi_{W(\boldsymbol{P}),\lambda}(\boldsymbol{s}) = [1_{W_i(P_i,s_i)\geq\lambda}]_{i\in[N]} \in \mathcal{A}^N\,, \tag{8.5}$$

which pulls all arms with Whittle indices larger than λ.

The Whittle index threshold policy maximizes the relaxed Lagrangian in Equation 8.2 under penalty λ, but may violate the budget constraints in Equation 8.1. In practice, we pull only the arms with the top K Whittle indices to respect the strict budget constraint.

8.4 Problem Statement: Online Learning in RMABs

We consider the online setting where the true transition probabilities $\boldsymbol{P}^\star$ are unknown to the learner. The learner interacts with an RMAB instance across multiple episodes, and only requires observations for the first H timesteps of each episode to estimate transition probabilities.

At the beginning of each episode $t \in [T]$, the learner starts the RMAB instance (timestep $h = 1$) from $\boldsymbol{s}_1 = \boldsymbol{s}_{\text{init}}$ and selects a new policy $\pi^{(t)}$. We consider the following setting:

- Each episode has an infinite horizon with discount factor γ.
- In each episode t, the learner proposes a policy $\pi^{(t)}$. The learner observes the first H timesteps[†], but receives the infinite discounted reward $U^{\boldsymbol{P},\lambda}_{\pi^{(t)}}(\boldsymbol{s}_1)$ to account for the long-term effect of $\pi^{(t)}$.
- We assume the MDP associated with each arm is *ergodic*. That is, starting from the given initial state, we assume H is large enough such that after H timesteps, there is at least $\varepsilon > 0$ probability of reaching any state $\boldsymbol{s} \in \mathcal{S}$.

[†]In practice, infinite time horizon means a large horizon that is much larger than H.

To evaluate the performance of our policy $\pi^{(t)}$, we compute *regret* against a full-information benchmark: the Whittle index threshold policy $\pi_{W(\boldsymbol{P}^\star),\lambda}$ with knowledge of the true transitions $\boldsymbol{P}^\star$. This offline benchmark measures the advantage gained from knowing the true transitions $\boldsymbol{P}^\star$.

Definition 11 (Frequentist regret of the Lagrangian objective)**.** *Given a penalty λ and the true transitions $\boldsymbol{P}^\star$, we define the* regret *of the policy $\pi^{(t)}$ in episode t relative to the optimal policy $\pi^\star = \pi_{W(\boldsymbol{P}^\star),\lambda}$:*

$$
\begin{aligned}
Reg_\lambda^{(t)} &:= U_{\pi^\star}^{\boldsymbol{P}^\star,\lambda}(\boldsymbol{s}_1) - U_{\pi^{(t)}}^{\boldsymbol{P}^\star,\lambda}(\boldsymbol{s}_1)\,, \\
Reg_\lambda(T) &:= \sum_{t\in[T]} Reg_\lambda^{(t)}\,.
\end{aligned}
\tag{8.6}
$$

However, the relaxed Lagrangian in Equation 8.2 with a randomly chosen penalty λ may not be a good proxy to the primal RMAB problem in Equation 8.1. Therefore, we define the Lagrangian using the optimal Lagrangian multiplier $\lambda^\star$ as the tightest upper bound of Equation 8.1.

Definition 12 (Frequentist regret of the optimal Lagrangian objective)**.** *Given $\boldsymbol{P}^\star$, we denote the optimal penalty by $\lambda^\star = \arg\min_\lambda U_{\pi^\star}^{\boldsymbol{P}^\star,\lambda}(\boldsymbol{s}_1)$. The regret of the* optimal Lagrangian objective *is defined by:*

$$
\begin{aligned}
Reg_{\lambda^\star}^{(t)} &:= U_{\pi^\star}^{\boldsymbol{P}^\star,\lambda^\star}(\boldsymbol{s}_1) - U_{\pi^{(t)}}^{\boldsymbol{P}^\star,\lambda^\star}(\boldsymbol{s}_1)\,, \\
Reg_{\lambda^\star}(T) &:= \sum_{t\in[T]} Reg_{\lambda^\star}^{(t)}\,.
\end{aligned}
\tag{8.7}
$$

The expected regret is approximated using the regret from the relaxed Lagrangian in Equation 8.2 as defined in Definition 11 and Definition 12.

8.5 UCWhittle: Optimistic Whittle Index Threshold Policy

A key challenge to UCB-based online learning in RMABs is that the estimated transitions impact estimates of future reward, so optimistic estimates of transition probabilities do not correspond to optimistic estimates of reward. We introduce a method, UCWhittle, to compute optimistic Whittle indices that account for highest future value.

8.5.1 Confidence Bounds of Transition Probabilities

To compute confidence bounds for every unknown transition probability in the RMAB instance, we maintain counts $N_i^{(t)}(s, a, s')$ for every state, action, and next state transition observed by episode t.

Given a chosen small constant $\delta > 0$, we estimate each transition probability $P_i(s, a, s')$ with the empirical mean

$$\hat{P}_i^{(t)}(s, a, s') := \frac{N_i^{(t)}(s, a, s')}{N_i^{(t)}(s, a)} \tag{8.8}$$

and confidence radius

$$d_i^{(t)}(s, a) := \sqrt{\frac{2|\mathcal{S}| \log(2|\mathcal{S}||\mathcal{A}|N\frac{t^4}{\delta})}{\max\{1, N_i^{(t)}(s, a)\}}} \tag{8.9}$$

where $N_i^{(t)}(s, a) := \sum_{s' \in \mathcal{S}} N_i^{(t)}(s, a, s')$. With these confidence bounds, the ball $\boldsymbol{B}$ of possible values for transition probabilities $\boldsymbol{P}$ is

$$\boldsymbol{B}^{(t)} = \left\{ \boldsymbol{P} \mid \left\| P_i(s, a, \cdot) - \hat{P}_i^{(t)}(s, a, \cdot) \right\|_1 \leq d_i^{(t)}(s, a) \, \forall i, s, a \right\}.$$

8.5.2 Optimistic Transitions and Whittle Indices

To translate confidence bounds in transition probabilities to the actual reward, we define an optimization problem ($\mathcal{P}_V$) to find for each arm i the *optimistic* transition probability $P_i^\dagger$, the value within the confidence bound that yields the *highest future value* from the starting state s_i:

$$\max_{V,Q,P_i \in \boldsymbol{B}_i^{(t)}} V(s_i) \qquad \text{s.t. } V(s) = \max_{a \in \mathcal{A}} Q(s,a) \tag{$\mathcal{P}_V$}$$
$$Q(s,a) = -\lambda a + R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P_i(s,a,s') V(s')$$

We prove Equation ($\mathcal{P}_V$) to be optimal in Section 8.6.

We use the optimistic transition $P_i^\dagger$ to compute the corresponding *optimistic Whittle index* $W_i^\dagger = W(P_i^\dagger, s_i)$. The Whittle index threshold policy $\pi_i^\dagger = \pi_{W_i^\dagger,\lambda}$ achieves the same value function derived from the transition $P_i^\dagger$, which maximizes Equation ($\mathcal{P}_V$). Aggregating all the arms together, optimistic policy $\pi^\dagger$ with optimistic transitions $\boldsymbol{P}^\dagger$ maximizes the future value of the current state $\boldsymbol{s}$.

8.5.3 UCWhittle Algorithm

After computing optimistic transitions and the corresponding optimistic Whittle indices ($\mathcal{P}_m$), we execute the optimistic Whittle index threshold policy. The full algorithm is outlined in Algorithm 9, and implementation details — including novel techniques for speeding up the computation of the Whittle index — are given in Appendix F.5.1.

8.5.4 Alternative Formulation for Whittle Index Upper Bound

Equation ($\mathcal{P}_V$) provides optimistic transition probabilities but requires separately solving for optimistic Whittle indices afterwards. Computing a Whittle index involves binary search, solving value iteration at every step, so is quite computationally expensive. We thus formulate a heuristic which

Algorithm 9: UCWhittle

Input: N arms, budget K, episode horizon H.
Initialization: counts $N_i^{(t)}(s, a, s') = 0$ for all s, a, s'. Randomly initialize penalty $\lambda^{(1)}$.
for *episode* $t \in \{1, 2, \ldots\}$ **do**
  Reset $h = 1$ and $\boldsymbol{s} = \boldsymbol{s}_{\text{init}}$; `// Reset RMAB instance`
  $P_i^{\dagger} = \mathcal{P}_V(s_i, N_i^{(t)}, \lambda^{(t)})$ for all $i \in [N]$; `// Compute an optimistic transition for each arm`
  $W_i = \textsc{ComputeWI}(P_i^{\dagger}, s_i)$ for all $i \in [N]$; `// Compute Whittle indices using Def. 9`
  Execute $\pi^{(t)}$ for H steps by pulling arms with the top K Whittle indices.
  Observe transitions $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$
  Update counts $N_i^{(t)}$, empirical means $\hat{\boldsymbol{P}}^{(t)}$, and confidence regions $\boldsymbol{B}^{(t)}$
  Set $\lambda^{(t+1)}$ to be the K-th highest Whittle index. ; `// Update penalty`

solves for the highest *Whittle index* directly (instead of highest *future value*) at the current state $s_{h,i}$:

$$\max_{m_i, V, Q, P_i, \in \boldsymbol{B}_i^{(t)}} \quad m_i \tag{$\mathcal{P}_m$}$$

$$\text{s.t. } V(s) = \max_{a \in \mathcal{A}} Q(s, a), \;\; Q(s, a = 0) = Q(s, a = 1)$$

$$Q(s, a) = -m_i a + R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P_i(s, a, s') V(s')$$

Solving Equation ($\mathcal{P}_m$) directly yields the maximal Whittle index estimate within the confidence bound. We thus save computation cost while maintaining a valid upper bound to the optimistic Whittle index from Equation ($\mathcal{P}_V$). The theoretical analysis does not hold for ($\mathcal{P}_m$), but empirically, we show that this heuristic achieves comparable performance with significantly lower computation.

8.6 Regret Analysis

We analyze the regret of our UCWhittle algorithm to provide the first frequentist regret analysis for RMABs. In this section, we use the Lagrangian objective as a proxy to the reward received from the proposed policy. Section 8.6.1 first assumes an arbitrary penalty λ is given to define the regret (Definition 11). Section 8.6.2 generalizes by defining the regret of the optimal Lagrangian objective based on the unknown optimal penalty $\lambda^\star$ (Definition 12). Section 8.6.3 provides an update rule for updating the penalty $\lambda^{(t)}$ after each episode. Full proofs are given in Appendix F.4.

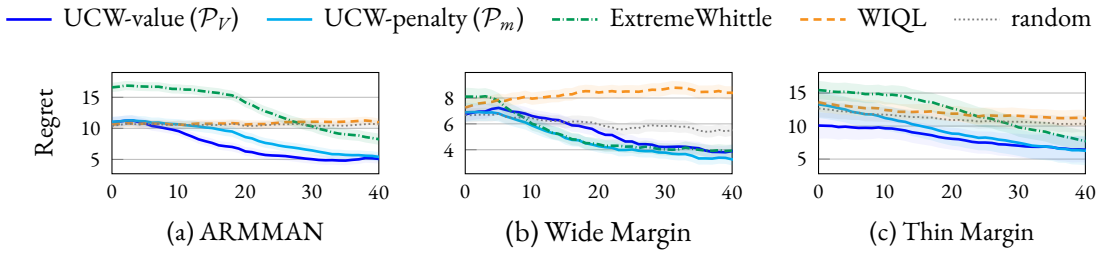


Figure 8.1: Cumulative discounted regret (lower is better) in each episode (x-axis) incurred by our UCWhittle approaches compared to baselines across the three domains with $N = 8$ arms, budget $B = 3$, episode length $H = 20$, and $T = 40$ episodes.

8.6.1 Regret Bound with Known Penalty

By the Chernoff bound, we know that with high probability the true transition $\boldsymbol{P}^\star$ lies within $\boldsymbol{B}^{(t)}$:

Proposition 6. *Given $\delta > 0$ and $t \geq 1$, we have:* $\Pr\left(\boldsymbol{P}^\star \in \boldsymbol{B}^{(t)}\right) \geq 1 - \frac{\delta}{t^4}$.

This bound can be used to bound the regret incurred, even when the confidence bound fails. In the following theorem, we bound the regret in the case where the confidence bound holds and when the penalty λ is given.

Theorem 11 (Regret decomposition)**.** *Given the penalty λ and $\boldsymbol{P}^\star \in \boldsymbol{B}^{(t)}$ for all t, we have:*

$$Reg_\lambda(T) = \sum_{t\in[T]} U_{\pi^\star}^{\boldsymbol{P}^\star,\lambda}(\boldsymbol{s}_1) - U_{\pi^{(t)}}^{\boldsymbol{P}^\star,\lambda}(\boldsymbol{s}_1) \leq \sum_{t\in[T]} U_{\pi^{(t)}}^{\boldsymbol{P}^{(t)},\lambda}(\boldsymbol{s}_1) - U_{\pi^{(t)}}^{\boldsymbol{P}^\star,\lambda}(\boldsymbol{s}_1)\,. \tag{8.10}$$

Proof. By optimality of Equation ($\mathcal{P}_V$) to enable $(P_i^{(t)}, \pi_i^{(t)}) = \arg\max_{P_i\in B_i^{(t)},\pi_i} V_{\pi_i}^{P_i,\lambda}(s_{1,i})$ and the assumption that the true transition lies within the confidence region $P_i^\star \in B_i^{(t)}$, we show that:

$$U_{\pi^\star}^{\boldsymbol{P}^\star,\lambda}(\boldsymbol{s}_1) = \sum_{i\in[N]} V_{\pi_i^\star}^{P_i^\star,\lambda}(s_{1,i}) \leq \sum_{i\in[N]} V_{\pi_i^{(t)}}^{P_i^{(t)},\lambda}(s_{1,i}) = U_{\pi^{(t)}}^{\boldsymbol{P}^{(t)},\lambda}(\boldsymbol{s}_1)\,. \qquad \square$$

Theorem 11 enables us to bound our regret by the difference between two future values under the same policy $\pi^{(t)}$.

Definition 13 (Bellman operator)**.** *Define the* Bellman operator *as:*

$$\mathcal{T}_{\pi_i}^{P_i} V(s) = \mathop{\mathbb{E}}_{a\sim\pi_i}\left[-\lambda a + R(s,a) + \gamma \sum_{s'\in S} P_i(s,a,s')V(s')\right]$$

Using Theorem 11 and the Bellman operator, we can further decompose the regret as:

Theorem 12 (Per-episode regret decomposition in the fully observable setting)**.** *For an arm i, fix $P_i^{(t)}$, $P_i^\star$, λ, and the initial state $s_{1,i}$. We have:*

$$V_{\pi_i^{(t)}}^{P_i^{(t)},\lambda}(s_{1,i}) - V_{\pi_i^{(t)}}^{P_i^\star,\lambda}(s_{1,i}) = \mathop{\mathbb{E}}_{P_i^\star,\pi_i^{(t)}}\left[\sum_{h=1}^{\infty} \gamma^{h-1}\left(\mathcal{T}_{\pi_i^{(t)}}^{P_i^{(t)}} - \mathcal{T}_{\pi_i^{(t)}}^{P_i^\star}\right) V_{\pi_i^{(t)}}^{P_i^{(t)},\lambda}(s_{h,i})\right]\,. \tag{8.11}$$

Theorem 12 further decomposes the regret in Equation 8.10 into individual differences in Bellman operators. The next theorem bounds the differences in Bellman operators by differences in transition probabilities.

Theorem 13. *Assume the penalty term $\lambda^{(t)} = \lambda$ is given and the RMAB instance is ε-ergodicity after H timesteps. Then with probability $1-\delta$, the cumulative regret in T episodes is:*

$$Reg_{\lambda}(t) \leq O\left(\frac{1}{\varepsilon}|S||A|^{\frac{1}{2}}NH\sqrt{T\log T}\right) . \tag{8.12}$$

Proof sketch. We focus on bounding the regret when the confidence bounds hold. By Theorem 11 and Theorem 12, we estimate the right-hand side of Equation 8.11 to bound the total regret by the L^1-difference in the transition probability:

$$\sum_{h=1}^{\infty}\gamma^{h-1}\left(\mathcal{T}^{P_i^{(t)}}_{\pi_i^{(t)}} - \mathcal{T}^{P_i^{\star}}_{\pi_i^{(t)}}\right)V^{P_i^{(t)}}_{\pi_i^{(t)}}(s_{h,i}) \leq \sum_{h=1}^{\infty}\gamma^{h-1}\left\|P_i^{(t)}(s_{h,i},a_{h,i},\cdot) - P_i^{\star}(s_{h,i},a_{h,i},\cdot)\right\|_1 V_{\max}. \tag{8.13}$$

We bound the regret outside of the horizon H by the ergodic assumption of the MDPs. For the regret inside the horizon H, we use the confidence radius to bound the L^1-norm of transition probability differences and count the number of observations for each state–action pair to express the regret as a sequence of random variables, whose sum can be bounded by Lemma 7 to conclude the proof. □

When the penalty term λ is given, Theorem 13 bounds the frequentist regret with a constant term depending on the ergodicity ε of the underlying true MDPs.

8.6.2 Regret Bound with Unknown Optimal Penalty

The analysis in Theorem 11 assumes a fixed and given penalty λ. Now, we generalize to regret defined in terms of the optimal but unknown penalty $\lambda^{\star}$ (Definition 12). We show that updating penalty $\lambda^{(t)}$ in Algorithm 9 achieves the same regret bound without requiring knowledge of the true transitions $\boldsymbol{P}^{\star}$ or optimal penalty $\lambda^{\star}$:

Theorem 14 (Regret bound with optimal penalty)**.** *Assume the penalty $\lambda^{(t)}$ in Algorithm 9 is updated by a saddle point $(\lambda^{(t)}, \boldsymbol{P}^{(t)}, \pi^{(t)}) = \arg\min_\lambda \max_{\boldsymbol{P},\pi} U_\pi^{\boldsymbol{P},\lambda}(\boldsymbol{s}_1)$ subject to constraints in Equation $(\mathcal{P}_V)$. The cumulative regret of the optimal Lagrangian objective is bounded with probability $1-\delta$:*

$$Reg_{\lambda^\star}(t) \leq O\left(\frac{1}{\varepsilon}|S||A|^{\frac{1}{2}}NH\sqrt{T\log T}\right) . \tag{8.14}$$

Proof sketch. The main challenge of an unknown penalty term $\lambda^\star$ is that the optimality of the chosen transition $\boldsymbol{P}^{(t)}$ and policy $\pi^{(t)}$ does not hold in Theorem 11 due to the misalignment of the penalty $\lambda^{(t)}$ used in solving Equation $(\mathcal{P}_V)$ and the penalty $\lambda^\star$ used in the regret.

Surprisingly, the optimality of $(\lambda^{(t)}, \boldsymbol{P}^{(t)}, \pi^{(t)}) = \arg\min_\lambda \max_{\boldsymbol{P},\pi} U_\pi^{\boldsymbol{P},\lambda}(\boldsymbol{s}_1)$ and $\lambda^\star = \inf_\lambda U_{\pi^\star}^{\boldsymbol{P}^\star,\lambda}(\boldsymbol{s}_1)$ is sufficient to show Theorem 11 by:

$$\begin{aligned}
&\underbrace{U_{\pi^\star}^{\boldsymbol{P}^\star,\lambda^\star} \leq}_{\lambda^\star \text{ minimizes } U_{\pi^\star}^{\boldsymbol{P}^\star,\lambda}} \underbrace{U_{\pi^\star}^{\boldsymbol{P}^\star,\lambda^{(t)}} \leq U_{\pi^{(t)}}^{\boldsymbol{P}^{(t)},\lambda^{(t)}}}_{\boldsymbol{P}^{(t)},\,\pi^{(t)} \text{ maximizes } U_\pi^{\boldsymbol{P},\lambda^{(t)}}} \underbrace{\leq U_{\pi^{(t)}}^{\boldsymbol{P}^{(t)},\lambda^\star}}_{\lambda^{(t)} \text{ minimizes } U_{\pi^{(t)}}^{\boldsymbol{P}^{(t)},\lambda}} \\
&\implies \mathrm{Reg}_{\lambda^\star}^{(t)} = U_{\pi^\star}^{\boldsymbol{P}^\star,\lambda^\star} - U_{\pi^{(t)}}^{\boldsymbol{P}^\star,\lambda^\star} \leq U_{\pi^{(t)}}^{\boldsymbol{P}^{(t)},\lambda^\star} - U_{\pi^{(t)}}^{\boldsymbol{P}^\star,\lambda^\star} .
\end{aligned} \tag{8.15}$$

where we omit the dependency on $\boldsymbol{s}_1$.

After taking summation over $t \in [T]$, Equation 8.15 leads to the same result as Theorem 11 without requiring knowledge of the optimal penalty $\lambda^\star$. The rest of the proof follows the same argument in Theorem 12 and Theorem 13 with the same regret bound. □

8.6.3 Penalty Update Rule

Theorem 14 suggests that the penalty term $\lambda^{(t)}$ should be defined by solving a minimax problem $(\lambda^{(t)}, \boldsymbol{P}^{(t)}, \pi^{(t)}) = \arg\min_\lambda \max_{\boldsymbol{P},\pi} U_\pi^{\boldsymbol{P},\lambda}(\boldsymbol{s}_1)$. However, the bilinear objective of $\mathcal{P}_V$ — where

the transition probability and value function variables are being multiplied together — is difficult to solve in a minimax problem. A heuristic solution is to solve the maximization problem using the previous penalty $\lambda^{(t-1)}$ to determine $\boldsymbol{P}^{(t)}$ and $\pi^{(t)}$ (Equation ($\mathcal{P}_V$)). We update $\lambda^{(t)}$ based on the current policy, set equal to the Kth largest Whittle index pulled at time t to minimize the Lagrangian. This update rule mimics the minimax update rule required by Theorem 14.

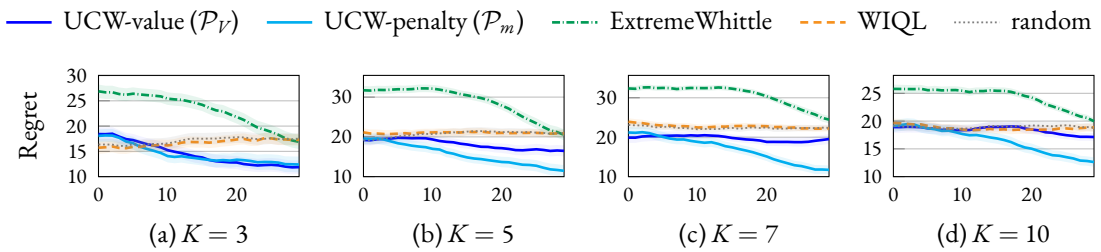


Figure 8.2: Varying budget ratio K/N, with $N = 15$ arms, on the ARMMAN domain. Our UCWhittle approaches perform stronger than baselines, particularly in the challenging low-budget scenarios.

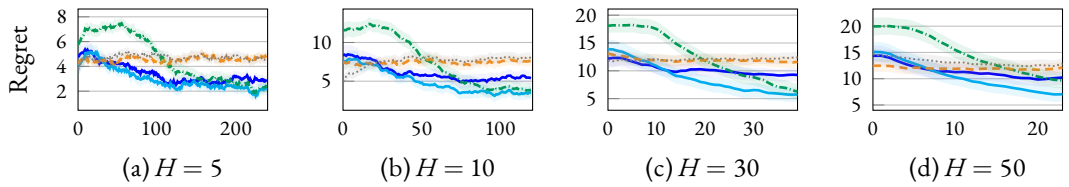


Figure 8.3: Changing episode length H on the ARMMAN domain. We run each setting for 1,200 total timesteps. *UCW-penalty* performs best with longer horizons. At shorter horizons, *UCW-value* converges in fewer timesteps, but more episodes are necessary: around episode $t = 100$ with a horizon $H = 5$ compared to episode $t = 16$ with horizon $H = 50$.

8.7 Experiments

We show that UCWhittle achieves consistently low regret across three domains, including one generated from real-world data on maternal health. Additional details about the dataset and data usage

are in Appendix B.3, and details about implementation (including novel techniques to speed up computation) and experiments are in Appendix F.5.[‡]

8.7.1 Preliminaries

Domains We consider three binary-action, binary-state settings. Across all domains, the binary states are *good* or *bad*, with reward 1 and 0 respectively. We impose two assumptions: that acting is always beneficial (more likely to transition to the good state), and that it is always better to start from the good state (more likely to stay in good state).

ARMMAN is a non-profit based in India that disseminates health information to pregnant women and mothers to reduce maternal mortality. Twice a week, ARMMAN sends automated voice messages to enrolled mothers relaying critical preventative health information. To improve listenership, the organization provides service calls to a subset of mothers; the challenge is selecting which subset to call to maximize engagement. We use real, anonymized data of the engagement behavior of 7,656 mothers from a previous RMAB field study[211]. We construct instances of RMAB problem with transition probabilities randomly sampled from the real dataset.

Wide Margin We randomly generate transition probabilities with high variance, while respecting the constraints specified above.

Thin Margin For a more challenging setting, we consider a synthetic domain with probabilities of transitioning to the good state constrained to the interval $[0.2, 0.4]$ to test the ability of each approach to discern smaller differences in transition probabilities.

Algorithms We evaluate both variants of UCWhittle (Algorithm 9) introduced in this paper. *UCWhittle-value* uses the value-maximizing bilinear program ($\mathcal{P}_V$) while *UCWhittle-penalty* uses the penalty-maximizing bilinear program ($\mathcal{P}_m$).

[‡]Code available at https://github.com/lily-x/online-rmab

In this paper, we focus on frequentist regret, thus we exclude the Bayesian regret baselines, e.g., Thompson sampling[159], because their regret bounds are averaged over a prior. We consider the following three regret baselines: *ExtremeWhittle* is similar to the the approach by Wang et al.[328]: estimate Whittle indices from the extreme points of the unknown transition probabilities, using UCBs of active transition probabilities and lower confidence bounds (LCB) for passive transition probabilities to estimate the gap between the value of acting versus not acting. We then solve a Whittle index policy using these estimates. *WIQL*[48] uses Q-learning to learn the value function of each arm at each state by interacting with the RMAB instance. *Random* takes a random action at each step, serving as a baseline for expected reward without using any strategic learning algorithm. Lastly, we evaluate an *optimal* policy which computes a Whittle index policy with access to the true transition probabilities.

Experiment setup We evaluate the performance of each algorithm across T episodes of length H. The per-episode reward is the cumulative discounted reward with discount rate $\gamma = 0.9$. We then compute regret by subtracting the reward earned by each algorithm from the reward of the *optimal* policy. Results are averaged over 30 random seeds and smoothed using exponential smoothing with a weight of 0.9. We ensure consistency by enforcing, across all algorithms, identical populations (transition probabilities for each arm) and initial state for each episode.

8.7.2 Results

The performance results across all three domains are shown in Figure 8.1. Our UCWhittle algorithm using the value-maximizing bilinear program (*UCW-value*) achieves consistently strong performance and generally converges by 600 timesteps (across varying episode lengths). In Figures 8.2 and 8.3 we evaluate performance while varying the budget K and episode length H, as the regret of UCWhittle (Theorem 13) has dependency on both the budget as a ratio of total number of arms

Method	Time (s)
UCWhittle-value	1090.92
UCWhittle-penalty	177.57
ExtremeWhittle	109.44
WIQL	3.39
random	1.32

Table 8.1: Average runtime of the different approaches across 500 timesteps with $N = 30$ arms and budget $B = 6$

(K/N) and episode length H. We see that UCW-value performs comparatively stronger than the baselines in the challenging low-budget settings, in which each arm pull has greater impact.

Our heuristic approach *UCW-penalty* — the penalty-maximizing bilinear program we present in Equation ($\mathcal{P}_m$) — shows strong performance. UCW-penalty performs even better than UCW-value in some settings, particularly in the ARMMAN domain with $N = 15$ arms (Figure 8.2). Notably in Table 8.1 we see this heuristic approach performs dramatically faster than UCW-value — a 6.1× speedup. Therefore while are able to establish regret guarantees only for UCW-value, we also propose UCW-penalty as a strong candidate for its strong performance and quick execution.

In Figures 8.2 and 8.3 we see *ExtremeWhittle* has poor performance particularly in the early episodes, consistently achieving higher regret than the random policy. Additionally, *WIQL* is slow to converge, performing similarly to the random baseline across the time horizons that we consider.

8.8 Conclusion

We propose the first online learning algorithm for RMABs based on the Whittle index policy, using an upper confidence bound–approach to learn transition dynamics. We formulate a bilinear program to compute optimistic Whittle indices from the confidence bounds of transition dynamics, enabling online learning using an optimistic Whittle index threshold policy. Theoretically, our work pushes the boundary of existing frequentist regret bounds in RMABs while enabling scalability

using the Whittle index threshold policy to decompose the solution approach.

9

Smoothed Online Combinatorial Optimization Using Imperfect Predictions*

9.1 Introduction

We consider the *smoothed online combinatorial optimization* problem, which is an extension of online convex optimization [132,279,362,133] and smoothed online convex optimization [197,198]. In the

smoothed online combinatorial optimization problem, an online learner is repeatedly optimizing a cost function with unknown changing parameter. In every time step, the learner chooses a feasible decision from a combinatorial feasible region before observing the parameter of the cost function. After the learner chooses the decision, the learner receives (i) the cost function parameter and the associated cost (ii) an additional known switching cost function dependent on the chosen decision and the previous decision. The goal of the learner is to minimize the cumulative cost in T time steps, including cost produced by the cost function and the switching cost.

Smoothed online combinatorial optimization is commonly seen in applications with online combinatorial decisions and switching penalty, including ride sharing with combinatorial driver-customer assignment[149], distributed streaming system with bipartite data-to-server assignment[117,302], and A-B testing in advertisement[45]. All these examples incur a potential switching cost when the decisions are changed, e.g., reassigning drivers or data to different locations or servers is costly, and changing advertisement campaign requires additional human resources. The challenge of online combinatorial decision-making and the presence of hidden switching cost motivate the study of smoothed online combinatorial optimization.

In this paper, we study the smoothed online combinatorial optimization where an imperfect predictive model is available. We assume that the predictive model can forecast the future cost parameters with uncertainties, and the uncertainties can evolve over time. We measure the performance of online algorithms by *dynamic regret*, which assumes a dynamic offline benchmark, i.e., the optimal performance when the cost function parameters are given a priori and the sequential decisions are allowed to change. The same use of predictions and dynamic regret are also studied in receding horizon control[212,60] in smoothed online convex optimization under different assumptions on the predictions[64,29,65,194,195]. In our case, the challenges of bounding dynamic regret inherit from smoothed online convex optimization, while the additional combinatorial structure further compli-

*This work was done during an internship at Adobe Research.

cates the analysis.

Main Contribution

Our main contribution is an online algorithm that plans ahead using the imperfect predictions within a dynamic planning window determined based on the predictive uncertainty of the predictive model. We summarize our contributions as follows:

- Given imperfect predictions with uncertainties, we show that planning based on predictions within a finite time horizon leads to a regret bound that is a function of the total predictive uncertainty with an additional potential switching cost. This bound quantifies one source of regret corresponding to the imperfectness of the predictions, while the other source comes from the additional switching cost (Theorem 15).

- Our regret bound in finite time horizon suggests using a dynamic planning window to optimally balance two sources of regret coming from predictive uncertainty and the switching cost, respectively. Iteratively selecting a dynamic planning window to plan ahead leads to a regret bound in infinite time horizon (Theorem 16).

- Specifically, when the uncertainties converge to 0 when more data is collected, we show that the cumulative regret is always sublinear (Theorem 17), which guarantees the no-regretness of Algorithm 10. We also quantify the dependency of the cumulative regret on the convergence rate of the uncertainty in some special cases (Corollary 1).

- Lastly, we show a lower bound on the total regret for any randomized online algorithm when predictive uncertainty is present. The order of the lower bound matches to the order of the upper bound in some special cases, which guarantees the tightness of our online algorithm and the corresponding regret bounds (Corollary 2).

Lastly, given predictions and dynamic planning windows, the smoothed online combinatorial optimization problem reduces to an offline combinatorial problem. We use an iterative algorithm to find an approximate solution to the offline problem efficiently, which largely reduces the computation cost compared to solving the large combinatorial problem using mixed-integer linear program.

Empirically, we evaluate our algorithm on the online distributed streaming problem motivated from Apache Kafka with synthetic traffic. We compare our algorithm using predictions and dynamic planning windows with various baselines. Our algorithm using predictions outperforms baselines without using predictions. Our experiments show an improvement of choosing the right dynamic planning windows against algorithms using fixed planning window, which demonstrates the importance of balancing uncertainty and the switching cost. The use of iterative algorithm also largely reduces the computation cost while keeping a comparable performance, leading to an effective scalable online algorithm that can be applied to real-world problems.

9.2 Related Work

Online convex optimization Online convex optimization [119,132,279,362] assumes the objective function is convex and no switching cost. In online convex optimization, *static regret* is most commonly used, which assumes a static benchmark with full information but the decisions over the entire time steps have to be static. Various variants of online gradient descents [362,133,134,291,103] were proposed with bounds on the static regret. However, the gradient-based approaches and the regret bounds do not directly generalize to the combinatorial setting due to the discreteness of the feasible region.

Smoothed online convex optimization with predictions Smoothed online convex optimization generalizes online convex optimization by assuming a switching cost that defines the cost of moving from the previous decision to the current one. [17] showed that smoothed online con-

vex optimization can achieve the same static regret bound using the algorithms in online convex optimization without switching cost. In terms of dynamic regret, receding horizon control[212] was proposed to leverage the predictions of future time step to make decision. Perfect[198,197] and imperfect[64,65,194,195] predictions are used to bound the performance of receding horizon control with fixed planning window size. Separately, chasing convex bodies[278,57,56,109] shares the same challenge of smoothed online convex optimization but focuses on the competitive ratio.

Nonetheless, the analyses in the convex objectives and feasible regions do not apply to the combinatorial setting. The planning window in receding horizon control is also restricted to be fixed across different time steps.

Online combinatorial optimization and metrical task system Online combinatorial optimization assumes a discrete feasible region that the learner can choose from *before* seeing the cost function. Existing results[23,175] focus on bounding dynamic regret in the case of linear objectives without switching cost. On the other hand, metrical task system assumes n discrete states that the learner can choose *after* seeing the cost function, and there is a metrical switching cost associated to every switch. Existing results focus on bounding *competitive ratio*, where the competitive ratio is lower bounded by $\Omega(\frac{\log n}{\log \log n})$[37,38] and upper bounded by $O(\log^2 n)$[55]. In contrast, *dynamic regret* is a stronger additive guarantee and is more challenging to analyze.

Our work shows that analyzing dynamic regret in an arbitrary smoothed online combinatorial optimization problem becomes tractable when an imperfect predictive model is given.

9.3 Problem Statement

An instance of smoothed online combinatorial optimization is composed of a cost function $f : \mathcal{Z} \times \Theta \to \mathbb{R}_{\geq 0}$ where $z \in \mathcal{Z}$ denotes all the feasible decisions that can be taken and $\theta \in \Theta$ denotes all the possible unknown parameters of the cost function, and a metric $d : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}_{\geq 0}$ that is

used to measure the distance of different decisions. At each time step t, the learner receives a feature $x_t \in \mathcal{X}$ that is correlated to the unknown parameters in the future. Based on the given feature x_t, the learner can predict the future parameters and choose a feasible decision $z_t \in \mathcal{Z}$ without seeing the future parameter θ_t. The parameter θ_t is revealed after the decision is executed and the learner receives an objective cost $f(z_t, \theta_t)$ with a switching cost $d(z_{t-1}, z_t)$ which measures the movement of the decisions made by time step t and $t-1$. The total cost of an online algorithm ALG up to time T is the summation of both the objective cost and the switching cost across all time steps:

$$\text{cost}_T(\text{ALG}) = \sum_{t=1}^{T} f(z_t, \theta_t) + d(z_t, z_{t-1}).$$

We want to compare to the offline benchmark OPT in time T that knows all the parameters in advance, which minimizes the total cost defined below:

$$\text{cost}_T(\text{OPT}) = \min_{z_t \in \mathcal{Z}\ \forall t} \sum_{t=1}^{T} f(z_t, \theta_t) + d(z_t, z_{t-1})$$

Definition 14. *An online algorithm* ALG *has a dynamic regret* $\rho(T)$ *if we have:*

$$\text{Reg}_T := \text{cost}_T(\text{ALG}) - \text{cost}_T(\text{OPT}) \leq \rho(T) \quad \forall T.$$

The goal of the learner is to design an online algorithm with a small dynamic regret bound $\rho(T)$.

9.3.1 Example: Online Distributed Streaming Systems

One application of smoothed online combinatorial optimization problems is the online load balancing problem in the distributed streaming system known as Apache Kafka [117,302]. The system is composed of k topics of streaming data and m servers as shown in Figure 9.1. At each time step t, the system maintains a bipartite assignment z_t between k topics and m servers so that the servers

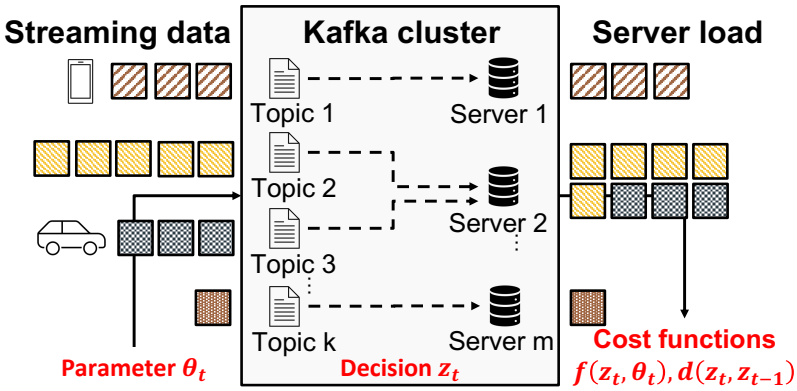
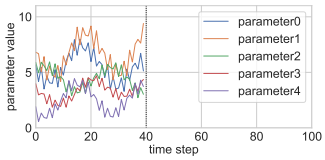


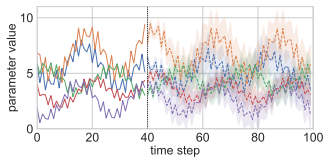
Figure 9.1: Apache Kafka maintains a bipartite assignment z_t between k topics and m servers to prepare for processing the streaming data. The streaming traffic θ_t comes later and gets routed to the corresponding servers. A server imbalance cost $f(z_t, \theta_t)$ and a switching cost $d(z_t, z_{t-1})$ due to assignment change are received.

can process the streaming data in real time. Specifically, each topic must be assigned to exactly one server. We use $z_t \in \mathcal{Z}_t \subseteq \{0,1\}^{k \times m}$ with $z_{t,i,j} = 1$ to denote assigning the topic i to server j at time t. The learner can use the parameters in the prior H time steps as the feature x_t that is correlated to the unknown future parameters. After the assignment is chosen, a new traffic vector $\theta_t \in \mathbb{R}^k$ arrives with each entry representing the number of incoming messages associated to the topic. Figure 9.1 illustrates how the data-to-server assignment works. A commonly used server imbalance cost is defined as makespan $f(z_t, \theta_t) = \left\| z_t^\top \theta_t \right\|_\infty$, the largest load across all servers.

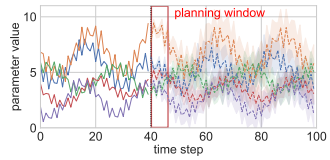
Paper structure We first discuss how planning based on predictions works and how to bound the associated dynamic regret using predictive uncertainty. Second, we discuss two different sources of regret, predictive uncertainty and the number of planning windows used. We propose to use a dynamic planning window to balance the tradeoff with a regret bound derived. Third, we propose an iterative algorithm to solve an offline problem by decoupling the temporal dependency caused by switching cost. Lastly, an application in distributed streaming system and Apache Kafka is discussed and used in our experiments.



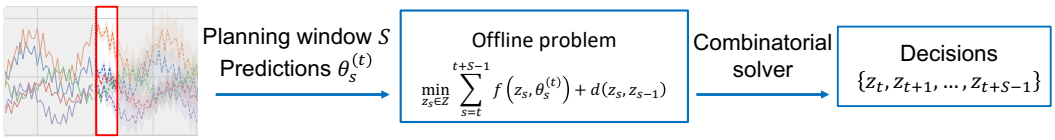
(a) The learner has access to the historical parameters $\{\theta_s \in \mathbb{R}^k\}_{s<t}$. We plot each entry of the parameter prior to time t as a time series to visualize the trend.



(b) The learner predicts the future parameters with uncertainty. Each entry of the parameter corresponds to a time series prediction problem.



(c) Given the predictions, we choose a dynamic planning window such that the total uncertainty within the window is of the same order of the switching cost.



(d) Given the predictions and the planning window, the planning problem reduces to an offline combinatorial problem. We can use any combinatorial solver to find a solution to the offline problem. The solution is executed in the planning window.

Figure 9.2: This flowchart summarizes how predictions are used to derive planning decisions. Fig. 9.2(a) shows the historical data prior to time t as multiple time series. Fig. 9.2(b) visualizes the predictions and uncertainty intervals learned from the historical parameters. Fig. 9.2(c) demonstrates how to determine the dynamic planning window. Fig. 9.2(d) solves an offline problem and executes accordingly.

9.4 Planning Using Predictions

Motivated by the use of predictions in smoothed online convex optimization[65,194,19], this section studies the connection of predictions and predictive uncertainties to the dynamic regret. To conduct the regret analysis below, we require the following assumptions to hold:

Assumption 1. *The cost function $f(z, \theta)$ is Lipschitz in $\theta \in \Theta$ with Lipschitz constant L, i.e., $\|\frac{\partial f(z,\theta)}{\partial \theta}\| \leq L$ for all $z \in \mathcal{Z}$ and $\theta \in \Theta$.*

Assumption 2. *The switching cost is upper bounded in the feasible region $\mathcal{Z}$ by $B = \sup_{z,z' \in \mathcal{Z}} d(z, z')$.*

Assumption 1 quantifies the change of the cost function with respect to the parameter. Assumption 2 quantifies the upper bound of switching cost.

9.4.1 Predictions with Uncertainty

Assumption 3. *We assume there is a predictive model that is trained based on the revealed parameters prior to time t. At time t, the predictive model takes the feature x_t and produces a sequence of predicted future parameters $\{\theta_s^{(t)}\}_{s\in\mathbb{N},s\geq t}$ with uncertainty $\{\varepsilon_s^{(t)}\}_{s\in\mathbb{N},s\geq t}$, where the distance between the prediction $\theta_s^{(t)}$ and the true parameter θ_s at time s is bounded by $\|\theta_s - \theta_s^{(t)}\| \leq \varepsilon_s^{(t)}$.*

We also assume that the predictive uncertainty $\varepsilon_s^{(t)}$ increases in s due to the difficulty of predicting further future parameters, while the predictive uncertainty decreases in t due to more training data available to train the predictive model.

9.4.2 Planning in Fixed Time Horizon

We first analyze the regret in fixed time horizon when we use the predictions to plan accordingly. More precisely, at time t, given the previous decision z_{t-1} at time $t-1$ and the prediction $\{\theta_s^{(t)}\}_{s\in\mathbb{N},s\geq t}$ of the future time steps, the learner selects a planning window $S \in \mathbb{N}$ to plan for the next S time steps by solving a minimization problem:

$$\{z_s\}_{s\in\{t,t+1,\cdots,t+S-1\}} = \underset{z_s\in\mathcal{Z}\ \forall s}{\operatorname{argmin}} \sum_{s=t}^{t+S-1} f(z_s, \theta_s^{(t)}) + d(z_s, z_{s-1}). \tag{9.1}$$

Solving the above finite time horizon optimization problem suggests a solution $\{z_s\}_{s\in\{t,t+1,\cdots,t+S-1\}}$ in the next S time steps to execute starting from time t. This process is summarized in Fig. 9.2.

However, since the predictions are not perfect, the suggested solution might not be the true optimal solution when the true cost function parameters are present. To compare with the true offline optimal solution using the true cost function parameters, we express the offline solution by:

$$\{z'_s\}_{s\in\{t,t+1,\cdots,t+S-1\}} = \underset{z_s\in\mathcal{Z}_s\ \forall s}{\operatorname{argmin}} \sum_{s=t}^{t+S-1} f(z_s, \theta_s) + d(z_s, z_{s-1}). \tag{9.2}$$

Algorithm 10: Dynamic Future Planning

Input: Total time steps T. Maximal switching cost B. A predictive model that can produce predictions $\{\theta_{t+s}^{(t)}\}_{s\in\mathbb{N}}$ at time t.
Initialization $t = 1$, # of planning windows $I = 0$.
while $t \leq T$ **do**
  Get predictions $\{\theta_s^{(t)}\}_{s\in\mathbb{N}, s\geq t}$ and predictive uncertainty $\{\varepsilon_s^{(t)}\}_{s\in\mathbb{N}, s\geq t}$ from the model.
  Find the largest S s.t. $2L \sum_{s=t}^{t+S-1} \varepsilon_s^{(t)} \leq B$.
  Solve the optimization problem in Equation (9.1) with starting time t and planning window S to get $\{z_s\}_{s\in\{t,t+1,\cdots,t+S-1\}}$.
  Execute z_s and receive θ_s with cost $f(z_s, \theta_s) + d(z_s, z_{s-1})$ at time $s \in \{t, \cdots, t+S-1\}$.
  Set $t = t + S, I = I + 1$.

The only difference between Equation (9.1) and Equation (9.2) is that Equation (9.2) has full access to the future cost parameters, while Equation (9.1) uses the predictions instead. We can define the difference by the following regret:

$$\mathrm{Reg}_t^{t+S-1}(z_{t-1}) = \left(\sum_{s=t}^{t+S-1} f(z_s, \theta_s) + d(z_s, z_{s-1})\right) - \left(\sum_{s=t}^{t+S-1} f(z'_s, \theta_s) + d(z'_s, z'_{s-1})\right). \tag{9.3}$$

We have the following bound on the regret:

Theorem 15. *Under Assumption 1, the regret from time step t to $t + S - 1$ in Equation 9.3 is upper bounded by:* $\mathrm{Reg}_t^{t+S-1}(z_{t-1}) \leq 2L \sum_{s=t}^{t+S-1} \varepsilon_s^{(t)}$. *where L is the Lipschitz constant in Assumption 1.*

Theorem 15 links the dynamic regret with the total predictive uncertainty in finite time horizon. Notice that the switching cost terms in Equation (9.3) are misaligned. Therefore, the proof requires not only the Lipschitzness of the objective function f but also the optimality conditions of both the offline and online planning problems to bound the total cumulative regret.

9.4.3 Infinite Time Horizon and Dynamic Planning Window

In the inifinite time horizon problem, the main idea is to reduce the problem to multiple finite time horizon problems with different planning window sizes.

Recall that the predictive uncertainty often increases when we try to predict the parameters in the far future, i.e., $\varepsilon_s^{(t)}$ is increasing in s. Since the regret in Theorem 15 directly relates to the predictive uncertainty in the planning window, it suggests keeping the planning window small to reduce the regret.

On the other hand, Theorem 15 assumes an identical initial decision z_{t-1} in the online problem (Equation (9.1)) and offline problem (Equation (9.2)). In the infinite time horizon case, two algorithms may start from different initial decisions, which may create an additional regret upper bounded by the maximum switching cost B due to the misalignment of the initial decision. This observation suggests using larger planning windows to avoid changing between different planning windows.

Therefore, we propose to balance two sources of regret by choosing the largest planning window S such that:

$$2L \sum_{s=t}^{t+S-1} \varepsilon_s^{(t)} \leq B \tag{9.4}$$

The choice of the dynamic planning window can ensure that the total excessive predictive uncertainty is upper bounded by cost B, while we also plan as far as possible to reduce the number of planning windows incurred during switching between different finite time horizons. The algorithm is described in Algorithm 10.

Theorem 16. *Given Lipschitzness L in Assumption 1 and the maximal switching cost B in Assumption 2, in T time steps, Algorithm 10 achieves cumulative regret upper bounded by $2BI$, where I is the*

total number of planning windows used in Algorithm 10.

Proof sketch. The regret of our algorithm comes from two parts: (i) regret from the discrepancy of the initial decision z_{t-1} and the initial decision of the offline optimal z^*_{t-1} at time t, the start of every planning window, and (ii) the incorrect predictions used in the optimization, which is bounded by Theorem 15.

The regret in part (i) is bounded by $d(z_{t-1}, z^*_{t-1}) \leq B$ for every planning window because it would take at most the maximal switching cost B to align different initial decisions before we can compare. Thus the total regret in part (i) is bounded by BI, where I is the number of planning windows executed in Algorithm 10.

The regret in part (ii) is bounded by Theorem 15 and the choice of the dynamic planning window in Equation (9.4). We have $\mathrm{Reg}_t^{t+S-1}(z^*_{t-1}) \leq 2L \sum_{s=t}^{t+S_i-1} \varepsilon_s^{(t)} \leq B$ for the i-th window. We can take summation over all planning windows to bound the total regret in part (ii) by: $\sum_{i=1}^{I} B = BI$. where combining two bounds concludes the proof. □

Theorem 16 links the excessive dynamic regret to I, the number of planning windows that Algorithm 10 uses. The next step is to bound the number of planning windows I by the total time steps T. In Theorem 17, we first show that the cumulative regret is always sublinear in T when the predictive uncertainty converges to 0 when more data is collected.

Theorem 17. *Under Assumption 1 and 2, if $\varepsilon_{t+s-1}^{(t)} = o(1)$ in t for all $s \in \mathbb{N}$, i.e., $\varepsilon_{t+s-1}^{(t)} \to 0$ when $t \to \infty$, then the cumulative regret of Algorithm 10 is sublinear in T.*

Proof. When the predictive uncertainty $\varepsilon_s^{(t)} \to 0$ when $t \to \infty$, the window size S_t that satisfies $2L \sum_{s=t}^{t+S_i-1} \varepsilon_s^{(t)} \leq B$ at time t converges to ∞ when $t \to \infty$. This suggests that the number of windows I required in total number of time steps T is strictly smaller than $\Theta(T)$, i.e., $I = o(T)$. By Theorem 16, the cumulative regret is upper bounded by $2BI = o(T)$, which is sublinear in T. □

Theorem 17 guarantees that the cumulative regret of Algorithm 10 in Theorem 16 is sublinear when the uncertainty converges to 0. This establishes the no-regretness of Algorithm 10 in dynamic regret, which is only known to be possible in the smoothed online *convex* optimization but not known in the smoothed online *combinatorial* optimization.

In some special cases of the predictive uncertainty, we can further provide a more precise bound on the cumulative regret in the following corollary.

Corollary 1. *If the uncertainty satisfies* $\varepsilon_{t+s-1}^{(t)} = O(\frac{s^a}{t^b})$, $\forall s, t \in \mathbb{N}$ *with* $a, b \in \mathbb{R}_{\geq 0}$, *we have:*

$$\mathrm{Reg}_T \leq \begin{cases} O(T^{1-\frac{b}{a+1}}) & \text{if } b < a+1 \\ O(\log T) & \text{if } b = a+1 \\ O(\log\log T) & \text{if } b > a+1 \end{cases} \cdot$$

Corollary 1 is proved by providing a more concrete bound on the number of planning windows I in Theorem 16. Corollary 1 also quantifies the dependency of the cumulative regret on the convergence rate of predictive uncertainty. When $b > 0$, the cumulative regret is always sublinear, which matches our result in Theorem 17.

9.4.4 Lower Bound on The Cumulative Regret

In this section, we provide a lower bound on the expected cumulative regret, showing that no randomized algorithm can achieve an expected cumulative regret lower than a term similar to the upper bound.

Corollary 2. *Given* $\varepsilon_{t+s-1}^{(t)} = \Omega(\frac{s^a}{t^b})$ *for all* $t, s \in \mathbb{N}$ *with* $0 \leq b$, *there exist instances such that for any*

randomized algorithm, the expected regret is at least:

$$\mathbb{E}[\mathrm{Reg}_T] \geq \begin{cases} \Omega(T^{1-b}) & if\, b < 1 \\ \Omega(\log T) & if\, b = 1 \\ \Omega(1) & if\, b > 1 \end{cases} \cdot$$

The lower bound suggests that there is no online learning algorithm that can achieve a cumulative regret that is smaller than the regret in Corollary 2. Specifically, we can see that the lower bound matches to the upper bound up to a logarithm factor when $a = 0$, which guarantees the tightness of our upper bound in Corollary 1 and Theorem 16 in the case of $a = 0$.

9.4.5 Extension to Probabilistic Bounds

In this paper, we primarily focus on the deterministic uncertainty bounds of the predictive model. The same analyses in Section 9.4 also generalize to probabilistic bounds of the predictive model that hold with high probability, e.g., with probability $1 - \delta_i$ for each prediction in the i-th planning window with size S_i. This kind of probabilistic bounds is commonly seen in the literature of probably approximately correct (PAC) learning, where the predictive error bound can be bounded by the number of training samples used in fitting the underlying hypothesis class. In this case, the regret analysis in Theorem 15 needs to additionally consider the event when the uncertainty bounds do not hold, which leads to an additional regret term with order $O(S_i\delta_i)$ in Theorem 15, leading to a linear term $\sum_{i=1}^{I} S_i\delta_i$ in Theorem 16.

Fortunately, we can also select a decreasing failure probability δ_i in the later planning windows when more samples are collected. As long as we can guarantee that the choice of uncertainty bound $\varepsilon_s^{(t)}$ and the failure probability δ_i at time t converge to 0 when more samples are collected, we can obtain a similar result as Theorem 17 showing the cumulative regret bound is sublinear in T. This

generalizes our results of deterministic bounds to probabilistic bounds.

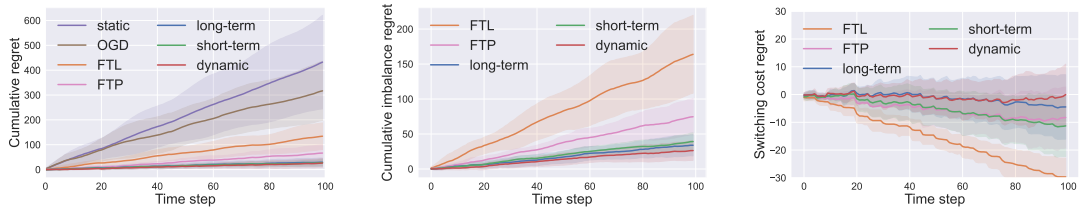
9.5 Experiment Setup

In our experiment, we use the distributed streaming system problems with synthetic data to compare our algorithm with other baselines.

Cost function and switching cost In the distributed streaming system, the learner maintains a bipartite assignment $\boldsymbol{z}_t \in \mathcal{Z}_t \subseteq \{0,1\}^{k\times m}$ between k topics and m servers at time step t to process the streaming data, where $\boldsymbol{z}_{t,i,j} = 1$ denotes that topic i is assigned to server j at time t to process the incoming traffic. Once the decision $\boldsymbol{z}_t$ is chosen at time t, a traffic vector $\theta_t \in \Theta \subseteq \mathbb{R}^k$ is revealed.

Given traffic θ_t and the chosen assignment z_t, we define the cost function by $f(z_t, \theta_t) = \|z_t^\top \theta_t\|_\infty$ as the resulting server imbalance cost, which is also known as makespan, i.e., the maximal number of messages a server needs to process across all servers. Minimizing makespan is a well-studied strongly NP-complete problem [116] with various approximation algorithms [136,191]. Additionally, we define the switching cost by $d(z, \boldsymbol{y}) := \mathbf{1}_k^\top |z - \boldsymbol{y}|\boldsymbol{u}$, where $|z - \boldsymbol{y}| \in \mathbb{R}_{\geq 0}^{k\times m}$ represents the number of switches of each pair of topic and server, and each entry of $\boldsymbol{u} \in \mathbb{R}^m$ denotes the unit switching cost associated to the corresponding server, which is randomly drawn from a uniform distribution $U[0, 2]$.

Data generation We assume that there are $k = 10$ topics to be assigned to $m = 3$ servers. We generate k time series, where each represents the trend of incoming traffic $\{\theta_{t,i}\}_{t\in[T]}$ of topic $i \in [k]$ as the cost function parameter. Each time series is generated by a composition of sine waves, an autoregressive process, and a Gaussian process to model the seasonality, trend, and the random process. We use sine waves with periods of 24 and 2 with amplitudes drawn from $U[1, 2]$ and $U[0.5, 1]$ to model the daily and hourly changes. We use an autoregressive process AR(1) that takes the weighted



(a) Average cumulative regret that includes the imbalance cost and switching cost.

(b) Average cumulative imbalance regret compared to the offline benchmark.

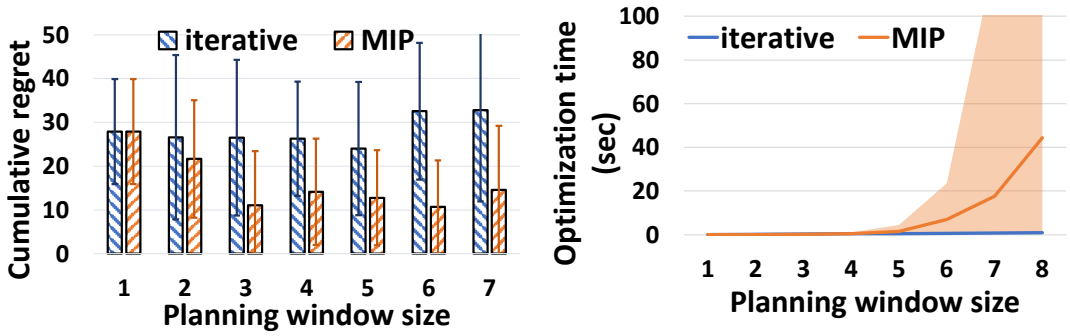
(c) Average cumulative switching cost regret compared to the offline benchmark.

Figure 9.3: We compare the performance of our approaches with various baselines without using predictions. The first takeaway is that methods using predictions largely outperform the methods without using predictions in Fig. 9.3(a). Secondly, choosing the right planning window can achieve a better imbalance cost in Fig. 9.3(b) with a small increase in the amount of switching cost in Fig. 9.3(c). All the algorithms are compared with an offline benchmark with full information. The shaded area refers to the region within first standard deviation.

sum of 0.9 of the previous signal and a 0.1 of a white noise to generate the next signal. Lastly, we use a rational quadratic kernel as the Gaussian process kernel.

Predictive model At time step t, to predict the incoming traffic $\theta_s \in \Theta \subseteq \mathbb{R}^k$ for all $s \geq t$, we collect all the historical data $\{\theta_{s'}\}_{s'<t}$ prior to time t and apply Gaussian process regression using the same rational quadratic kernel on the historical data to generate predictions $\{\theta_s^{(t)}\}_{s\geq t}$ of the future time steps. We use the standard deviation learned from Gaussian process regression as the uncertainty $\{\varepsilon_s^{(t)}\}_{s\geq t}$.

Experimental setup For each instance of the load balancing problem, we assume 50 historical data have been collected a priori to stabilize Gaussian process regression. We run different online algorithms for another 100 time steps with hidden incoming traffic to measure the performance of online algorithms. For each setup, we run 10 independent trials with different random seeds to estimate the average performance. All the results are plotted with average value and the corresponding standard deviation.



(a) Cumulative regret of methods using different planning window sizes and different optimization approaches.

(b) Average running time per optimization of different optimization methods with different planning window sizes.

Figure 9.4: Comparison of different methods of solving Equation (9.1) and different planning window sizes.

9.6 Experimental Results

We compare with our algorithm with baselines in the literature of online convex optimization:

- The **static** approach uses the initial assignment and never adjusts dynamically.
- The Online Gradient Descent (**OGD**) updates the assignment by running gradient descent on the cost function received previously and project back to the discrete feasible region.
- The Follow-The-Leader (**FTL**) aggregates all the cost functions received in the past and finds the optimal decision that optimizes the historical cost functions with switching cost.
- The Follow-The-Previous (**FTP**) optimizes the cost function in the last time step.
- The **short-term** algorithm and the **long-term** algorithm both use predictions but with deterministic planning window sizes 1 and 10, respectively.
- The **dynamic** algorithm refers to our algorithm using a dynamic planning window determined by the predictive uncertainty.

All the algorithms compare with an offline benchmark with full information. Since the offline problem is NP-hard to solve, we split the offline problem into chunks of size 5 and solve each of them optimally using mixed integer program to get the offline performance.

Effect of predictions In Fig. 9.3, we compare the performance of baselines (static, OGD, FTL, FTP) with approaches using predictions with different planning window sizes (short-term, long-term, dynamic). We first notice that OGD and FTL perform worse than FTP, which simply follows the previous cost function to update solution. Due to the smoothness of the cost function parameters, optimizing over the previous cost function can be a strong baseline.

Secondly, the methods using predictions further improve the solution quality. Using predictions can help leverage the seasonality and trend information, and leave the uncertainty to the planning part. On the other hand, the OGD and the FTL algorithms are designed to deal with the case without predictable pattern and switching cost. The different purposes of algorithm design make our algorithm more applicable to our problem.

Lastly, in Fig. 9.3(a), we can see that the dynamic algorithm achieves the smallest cumulative regret compared to the short-term algorithm and the long-term algorithm using planning window with size 1 and 10, respectively. Fig. 9.3(b) and Fig. 9.3(c) further compare different performance metrics. We can see that our approach of choosing proper planning window can achieve much smaller server imbalance performance while requiring slightly more switching cost only. Methods considering less future effect (FTL, FTP, short-term) can be reluctant to switch and underestimate the benefit of switching, which results in a smaller switching cost but larger imbalance cost. In contrast, the long-term algorithm using larger planning window instead can be harmed by the increasing predictive uncertainty, which leads to incorrect planning decision due to the uncertainty. This result justifies the benefit of predictions and the right planning window to balance between uncertainty and the switching cost.

Effect of planning window size In Fig. 9.4(a), we compare the performance of different choices of planning window size and different ways of solving the offline problem in Equation (9.1). First, if we use mixed integer program (**MIP**), we can see a clear improvement by using a larger planning window and a slightly degraded performance after window size exceeds 3. This empirical result matches to our analysis of shorter and longer planning windows, where the dynamic planning window suggests a planning window with size around 3. We also compare with an **iterative** algorithm (Algorithm 16 in Appendix G.6) that is used to approximately solve the NP-hard offline problem in Equation (9.1). The effect of planning window size is less significant due to the suboptimality of the iterative algorithm. But we can still see a similar benefit while using an appropriate planning window size.

Fig. 9.4(b) compares the runtime of solving Equation (9.1) using different approaches and planning window sizes. Runtime of solving the optimization problem is important because decisions have to be made in real time. We can see that MIP requires an exponentially increasing runtime because the combinatorial structure and the linearly increasing number of binary variables when the window size grows. On the other hand, the iterative algorithm solves the problem approximately and more efficiently. In short, the MIP algorithm achieves the best performance but with an expensive computation, while the iterative algorithm scales better but with a loss in the solution quality.

9.7 Conclusion

This paper studies the smoothed online combinatorial optimization problem with switching cost. We show that when predictions with uncertainty are available, we can bound the dynamic regret by the convergence of the predictive uncertainty, which links the bound on dynamic regret to the predictability of the incoming cost function parameters. Our analysis suggests using a dynamic planning window dependent on the sequence of predictive uncertainties. Our dynamic planning win-

dow can optimize the regret, where we empirically show in our experiments that using a predictive model and an appropriate planning window can further improve the performance.

Part III

Optimization in Multi-agent Systems

10

End-to-End Gradient Descent for Stackelberg Games

10.1 Introduction

Stackelberg games are commonly adopted in many real-world applications, including security [150,114], wildlife conservation [94], and commercial decisions made by firms [228,26,358]. Moreover, many real-

istic settings involve a single leader with multiple self-interested followers such as wildlife conservation efforts with a central coordinator and a team of defenders [115,114]; resource management in energy [26] with suppliers, aggregators, and end users; or security problems with a central insurer and a set of vulnerable agents [228,154]. Solving Stackelberg games with multiple followers is challenging in general [39,70]. Previous work often reformulates the followers' best response as stationary and complementarity constraints in the leader's optimization [280,40,39,70,59], casting the entire Stackelberg problem as a single optimization problem. This reformulation approach has achieved significant success in problems with linear or quadratic objectives, assuming a unique equilibrium or a specific equilibrium concept, e.g., followers' optimistic or pessimistic choice of equilibrium [141,40,39]. The reformulation approach thoroughly exploits the structure of objectives and equilibrium to conquer the computation challenge. However, when these conditions are not met, reformulation approach may get trapped in low-quality solutions.

In this paper, we propose an end-to-end gradient descent approach to solve multi-follower Stackelberg games. Specifically, we run gradient descent by back-propagating through a sampled Nash equilibrium reached by followers to update the leader's strategy. Our approach overcomes weaknesses of reformulation approaches as (i) we decouple the leader's optimization problem from the followers', casting it as a learning problem to be solved by end-to-end gradient descent through the followers' equilibrium; and (ii) back-propagating through a sampled Nash equilibrium enables us to work with arbitrary equilibrium selection procedures and multiple equilibria.

In short, we make several contributions. First, we provide a procedure for differentiating through a Nash equilibrium assuming uniqueness (later we relax the assumption). Because each follower must simultaneously best respond to every other follower, the Karush–Kuhn–Tucker (KKT) conditions [182] for each follower must be simultaneously satisfied. We can thus differentiate through the system of KKT conditions and apply the implicit function theorem to obtain the gradient. Second, we relax the uniqueness assumption and extend our approach to an arbitrary, potentially stochastic,

equilibrium selection oracle. We first show that given a stochastic equilibrium selection procedure, using optimistic or pessimistic assumptions to solve Stackelberg games with stochastic equilibria can yield payoff to the leader that is arbitrarily worse than optimal. To address the issue of multiple equilibria and stochastic equilibria, we formally characterize stochastic equilibria with a concept we call *equilibrium flow*, defined by a partial differential equation. The equilibrium flow ensures the stochastic gradient computed from the sampled Nash equilibrium is unbiased, allowing us to run stochastic gradient descent to differentiate through the stochastic equilibrium. We also discuss how to compute the equilibrium flow either from KKT conditions under certain sufficient conditions or by solving the partial differential equation. This paper is the first to guarantee that the gradient computed from an arbitrary stochastic equilibrium sampled from multiple equilibria is a differentiable, unbiased sample. Third, to address the challenge that the feasibility of the leader's strategy may depend on the equilibrium reached by the followers (e.g., when a subsidy paid to the followers is conditional on their actions as in [270,224]), we use an augmented Lagrangian method to convert the constrained optimization problem into an unconstrained one. The Lagrangian method combined with our unbiased Nash equilibrium gradient estimate enables us to run stochastic gradient descent to optimize the leader's payoff while also satisfying the equilibrium-dependent constraints.

We conduct experiments to evaluate our approach in three different multi-follower Stackelberg games: normal-form games with a leader offering subsidies to followers, Stackelberg security games with a planner coordinating multiple defenders, and cyber insurance games with an insurer and multiple customers. Across all three examples, the leader's strategy space is constrained by a budget constraint that depends on the equilibrium reached by the followers. Our gradient-based method provides a significantly higher payoff to the leader evaluated at equilibrium, compared to existing approaches which fail to optimize the leader's utility and often produce large constraint violations. These results, combined with our theoretical contributions, demonstrate the strength of our end-to-end gradient descent algorithm in solving Stackelberg games with multiple followers.

10.2 Related Work

Stackelberg models with multiple followers Multi-follower Stackelberg problems have received a lot of attention in domains with a hierarchical leader-follower structure [230,358,202,289,286]. For example, mechanism and auction design can be formulated as a Stackelberg game where the mechanism design is the leader and all the participating agents are the followers [120,162,359,345]. The leader can also leverage machine learning, data, and predictions to design strategy to play against the followers [5,35,34] and optimize the leader's payoff. However, although single-follower normal-form Stackelberg games can be solved in polynomial time [176,49], the problem becomes NP-hard when multiple followers are present, even when the equilibrium is assumed to be either optimistic or pessimistic [40,70]. Existing approaches [40,26] primarily leverage the leader-follower structure in a bilevel optimization formulation [69], which can be solved by reformulating the followers' best response into non-convex stationary and complementarity constraints in the leader's optimization problem [287]. Various optimization techniques, including branch-and-bound [70] and mixed-integer programs [40], are adopted to solve the reformulated problems. However, these reformulation approaches highly rely on well-behaved problem structure, which may encounter large mixed integer non-linear programs when the followers have non-quadratic objectives. Additionally, these approaches mostly assume uniqueness of equilibrium or a specific equilibrium concept, e.g., optimistic or pessimistic, which may not be feasible [115]. Previous work on the stochastic equilibrium drawn from multiple equilibria in Stackelberg problems [199] mainly focuses on the existence of an optimal solution, while our work focuses on actually solving the Stackelberg problems to identify the best action for the leader.

In contrast, our approach solves the Stackelberg problem by differentiating through the equilibrium reached by followers to run gradient descent in the leader's problem. Our approach also applies to any stochastic equilibrium drawn from multiple equilibria by establishing the unbiasedness

of the gradient computed from a sampled equilibrium using a partial differential equation.

Differentiable optimization When there is only a single follower optimizing his utility function, differentiating through a Nash equilibrium reduces to the framework of differentiable optimization [251,11,3,32]. When there are two followers with conflicting objectives (zero-sum), differentiating through a Nash equilibrium reduces to a differentiable minimax formulation [200,201]. Lastly, when there are multiple followers, Li et al. [192] follow the sensitivity analysis and variational inequalities (VIs) literature [215,306,77,245] to express a unique Nash equilibrium as a fixed-point to the projection operator in VIs to differentiate through the equilibrium. Li et al. [196] further extend the same approach to structured hierarchical games. Nonetheless, these approaches rely on the uniqueness of Nash equilibrium. In contrast, our approach generalizes to multiple equilibria.

10.3 Stackelberg Games With a Single Leader and Multiple Followers

In this paper, we consider a Stackelberg game composed of one leader and n followers. The leader first chooses a strategy $\pi \in \Pi$ that she announces, then the followers observe the leader's strategy and respond accordingly. When the leader's strategy π is determined, the followers form an n-player simultaneous game with n followers, where the i-th follower minimizes his own objective function $f_i(z_i, z_{-i}, \pi)$, which depends on his own action $z_i \in \mathcal{Z}_i$, other followers' actions $z_{-i} \in \mathcal{Z}_{-i}$, and the leader's strategy $\pi \in \Pi$. We assume that each strategy space is characterized by linear constraints: $\mathcal{Z}_i = \{z_i \mid A_i z_i = b_i, G_i z_i \leq h_i\}$. We also assume perfect information—all the followers know other followers' utility functions and strategy spaces.

10.3.1 Nash Equilibria

We call $\boldsymbol{z}^* = \{z_1^*, z_2^*, \ldots, z_n^*\}$ a Nash equilibrium if no follower has an incentive to deviate from their current strategy, where we assume each follower *minimizes*[*] his objective:

$$\forall i : f_i(z_i^*, z_{-i}^*, \pi) \leq f_i(z_i, z_{-i}^*, \pi) \qquad \forall z_i \in \mathcal{Z}_i. \tag{10.1}$$

As shown in Figure 10.1, when the leader's strategy π is chosen and passed to an n-player game composed of all followers, we assume the followers converge to a Nash equilibrium $\boldsymbol{z}^*$.

In the first section, we assume there is a unique Nash equilibrium returned by an oracle $\boldsymbol{z}^* = \mathcal{O}(\pi)$. We later generalize to the case where there are multiple equilibria with a stochastic equilibrium selection oracle which randomly outputs an equilibrium $\boldsymbol{z} \sim \mathcal{O}(\pi)$ drawn from a distribution with probability density function $p(\cdot, \pi) : \mathcal{Z} \to \mathbb{R}_{\geq 0}$.

10.3.2 Leader's Optimization Problem

When the leader chooses a strategy π and all the followers reach an equilibrium $\boldsymbol{z}^*$, the leader receives a payoff $f(\boldsymbol{z}^*, \pi)$ and a constraint value $g(\boldsymbol{z}^*, \pi)$. The goal of the Stackelberg leader is to choose an optimal π to maximize her utility while satisfying the constraint.

Definition 15 (Stackelberg problems with multiple followers and unique Nash equilibrium)**.** *The leader chooses a strategy π to maximize her utility function f subject to constraints g evaluated at the unique equilibrium $\boldsymbol{z}^*$ induced by an equilibrium oracle $\mathcal{O}$, i.e.,:*

$$\max_{\pi} f(\boldsymbol{z}^*, \pi) \qquad \textit{s.t.} \quad \boldsymbol{z}^* = \mathcal{O}(\pi), \quad g(\boldsymbol{z}^*, \pi) \leq 0. \tag{10.2}$$

[*]We use minimization formulation to align with the convention in convex optimization. In our experiments, examples of maximization problems are used, but the same approach applies.

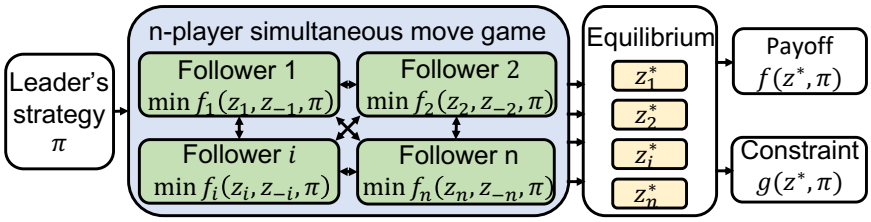


Figure 10.1: Given leader's strategy π, followers respond to the leader's strategy and reach a Nash equilibrium $\boldsymbol{z}^*$. The leader's payoff and the constraint depend on both the leader's strategy π and the equilibrium $\boldsymbol{z}^*$.

This problem is hard because the objective $f(\boldsymbol{z}^*, \pi)$ depends on the Nash equilibrium $\boldsymbol{z}^*$ reached by the followers. Moreover, notice that the feasibility constraint $g(\boldsymbol{z}^*, \pi)$ also depends on the equilibrium, which creates a complicated feasible region for the leader's strategy π.

10.3.3 Gradient Descent Approach

To solve the leader's optimization problem, we propose to run gradient descent to optimize the leader's objective. This requires us to compute the following gradient:

$$\frac{df(\boldsymbol{z}^*, \pi)}{d\pi} = f_\pi(\boldsymbol{z}^*, \pi) + f_{\boldsymbol{z}}(\boldsymbol{z}^*, \pi) \cdot \frac{d\boldsymbol{z}^*}{d\pi}. \tag{10.3}$$

The terms $f_\pi, f_{\boldsymbol{z}}$ above are easy to compute since the payoff function f is explicitly given. The main challenge is to compute $\frac{d\boldsymbol{z}^*}{d\pi}$ because it requires estimating how the Nash equilibrium $\boldsymbol{z}^*$ reached by followers responds to any change in the leader's strategy π.

10.4 Gradient of Unique Nash Equilibrium

In this section, we assume a unique Nash equilibrium reached by followers. Motivated by the technique proposed by Amos & Kolter [11], we show how to differentiate through multiple KKT conditions to derive the derivative of a Nash equilibrium.

10.4.1 Differentiating Through KKT Conditions

Given the leader's strategy π, we express the KKT conditions of follower i with dual variables λ_i^* and ν_i^* by:

$$\begin{cases} \nabla_{z_i} f_i(z_i^*, z_{-i}^*, \pi) + G_i^\top \lambda_i^* + A_i^\top \nu_i^* = 0 \\ \mathrm{Diag}(\lambda_i^*)(G_i z_i^* - h_i) = 0 \\ A_i z_i^* = b_i. \end{cases} \tag{10.4}$$

We want to estimate the impact of π on the resulting Nash equilibrium $\boldsymbol{z}^*$. Supposing the objective functions $f_i \in C^2$ are twice-differentiable, we can compute the total derivative of the the KKT system in Equation 10.4 written in matrix form:

$$\begin{bmatrix} \nabla^2_{z_i z_i} f_i & \nabla^2_{z_{-i} z_i} f_i & G_i^\top & A_i^\top \\ \mathrm{Diag}(\lambda_i^*) G_i & 0 & \mathrm{Diag}(G_i z_i^* - h_i) & 0 \\ A_i & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} dz_i^* \\ dz_{-i}^* \\ d\lambda_i^* \\ d\nu_i^* \end{bmatrix} = \begin{bmatrix} -\nabla^2_{\pi z_i} f_i d\pi - dG_i^\top \lambda_i^* - dA_i^\top \nu_i^* \\ -\mathrm{Diag}(\lambda_i^*)(dG_i z_i^* - dh_i) \\ db_i - dA_i z_i^* \end{bmatrix}.$$

Since we assume the constraint matrices are constant, dG_i, dh_i, dA_i, db_i can be ignored. We concatenate the linear system for every follower i and move $d\pi$ to the denominator:

$$\begin{bmatrix} \nabla_{\boldsymbol{z}} F & G^\top & A^\top \\ \mathrm{Diag}(\lambda^*) G & \mathrm{Diag}(G\boldsymbol{z}^* - h) & 0 \\ A & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{d\boldsymbol{z}^*}{d\pi} \\ \frac{d\lambda^*}{d\pi} \\ \frac{d\nu^*}{d\pi} \end{bmatrix} = \begin{bmatrix} -\nabla_\pi F \\ 0 \\ 0 \end{bmatrix} \tag{10.5}$$

where $F = [(\nabla_{z_1} f_1)^\top, \ldots, (\nabla_{z_n} f_n)^\top]^\top$ is a column vector, and $G = \mathrm{Diag}(G_1, G_2, \ldots, G_n), A = \mathrm{Diag}(A_1, A_2, \ldots, A_n)$ are the diagonalized placement of a list of matrices. In particular, the KKT

Follower 2 \ Follower 1			
	1	0	0
	0	1	0
	0	0	1

(a) Strategy 1 payoffs

Follower 2 \ Follower 1			
	0	1	0
	0	0	1
	1	0	0

(b) Strategy 2 payoffs

Follower 2 \ Follower 1			
	0	0	1
	1	0	0
	0	1	0

(c) Strategy 3 payoffs

Follower 2 \ Follower 1			
	C	0	$-\varepsilon$
	$C-\varepsilon$	0	0
	0	$C-\varepsilon$	$-C$

(d) Leader payoffs

Figure 10.2: Payoff matrices from Theorem 18 where the leader has 3 strategies. Follower payoffs for each strategy in (a)–(c) where both followers receive the same payoff; leader payoffs in (d).

matrix on the left-hand side of Equation 10.5 matches the sensitivity analysis of Nash equilibria using variational inequalities[92,77].

Proposition 7. *When the Nash equilibrium is unique and the KKT matrix in Equation 10.5 is invertible, the implicit function theorem holds and* $\frac{d\boldsymbol{z}^*}{d\pi}$ *can be uniquely determined by Equation 10.5.*

Proposition 7 ensures the sufficient conditions for applying Equation 10.5 to compute $\frac{d\boldsymbol{z}^*}{d\pi}$. Under these sufficient conditions, we can compute Equation 10.3 using Equation 10.5.

10.5 Gradient of Stochastic Equilibrium

In the previous section, we showed how to compute the gradient of a Nash equilibrium when the equilibrium is unique. However, this can be restrictive because Stackelberg games with multiple followers often have multiple equilibria that the followers can stochastically reach one. For example, both selfish routing games in the traffic setting[271] and security games with multiple defenders[115] can have multiple equilibria that are reached in multiple independent runs.

In this section, we first demonstrate the importance of stochastic equilibrium by showing that optimizing over optimistic or pessimistic equilibrium could lead to arbitrarily bad leader's payoff under the stochastic setting. Second, we generalize our gradient computation to the case with multiple equilibria, allowing the equilibrium oracle $\mathcal{O}$ to stochastically return a sample equilibrium from

a distribution of multiple equilibria. Lastly, we discuss how to compute the gradient of different types of equilibria and its limitation.

10.5.1 Importance of Stochastic Equilibrium

When the equilibrium oracle is stochastic, our Stackelberg problem becomes a stochastic optimization problem:

Definition 16 (Stackelberg problems with multiple followers and stochastic Nash equilibria)**.** *The leader chooses a strategy π to optimize her expected utility and satisfy the constraints in expectation under a given stochastic equilibrium oracle $\mathcal{O}$:*

$$\max_{\pi} \mathop{\mathbb{E}}_{\boldsymbol{z}^* \sim \mathcal{O}(\pi)} f(\boldsymbol{z}^*, \pi) \quad \textit{s.t.} \mathop{\mathbb{E}}_{\boldsymbol{z}^* \sim \mathcal{O}(\pi)} g(\boldsymbol{z}^*, \pi) \leq 0. \tag{10.6}$$

In particular, we show that if we ignore the stochasticity of equilibria by simply assuming optimistic or pessimistic equilibria, the leader's expected payoff can be arbitrarily worse than the optimal one.

Theorem 18. *Assuming the followers stochastically reach a Nash equilibrium drawn from a distribution over all equilibria, solving a Stackelberg game under the assumptions of optimistic or pessimistic equilibrium can give the leader expected payoff that is arbitrarily worse than the optimal one.*

Proof. We consider a Stackelberg game with one leader and two followers (row and column player) with no constraint. The leader can choose 3 different strategies, each corresponding to a payoff matrix in Figure 10.2(a)–10.2(c), where both followers receive the same payoff in the entry when they choose the corresponding row and column. In each payoff matrix, there are three pure Nash equilibria; we assume the followers reach any of them uniformly at random. After the followers reach a Nash equilibrium, the leader receives the corresponding entry in the payoff matrix in Figure 10.2(d).

Under the optimistic assumption, the leader would choose strategy 1, expecting followers to break the tie in favor of the leader, yielding payoff C. Instead, the three followers select a Nash equilibria uniformly at random, yielding expected payoff $\frac{C+0-C}{3} = 0$. Under the pessimistic assumption, the leader chooses strategy 2, anticipating and receiving an expected payoff of zero. Under the correct stochastic assumption, she chooses strategy 3 with expected payoff $\frac{C-\varepsilon+C-\varepsilon-\varepsilon}{3} = \frac{2}{3}C-\varepsilon \gg 0$, which can be arbitrarily higher than the optimistic or pessimistic payoff when $C \to \infty$. □

Theorem 18 justifies why we need to work on stochastic equilibrium when the equilibrium is drawn stochastically in Definition 16. In the following section, we show how to apply gradient descent to optimize the leader's payoff by differentiating through followers' equilibria with a stochastic oracle.

10.5.2 Equilibrium Flow and Unbiased Gradient Estimate

Our goal is to compute the gradient of the objective in Equation 10.6: $\frac{d}{d\pi}\mathbb{E}_{\boldsymbol{z}^*\sim\mathcal{O}(\pi)}f(\boldsymbol{z}^*,\pi)$. However, since the distribution of the oracle $\mathcal{O}(\pi)$ can also depend on π, we cannot easily exchange the gradient operator into the expectation.

To address the dependency of the oracle $\mathcal{O}(\pi)$ on π, we use $p(\boldsymbol{z},\pi)$ to represent the probability density function of the oracle $\boldsymbol{z} \sim \mathcal{O}(\pi)$ for every π. We want to study how the oracle distribution changes as the leader's strategy π changes, which we denote by *equilibrium flow* as defined by the following partial differential equation:

Definition 17 (Equilibrium Flow)**.** *We call $v(\boldsymbol{z},\pi)$ the equilibrium flow of the oracle $\mathcal{O}$ with probability density function $p(\boldsymbol{z},\pi)$ if $v(\boldsymbol{z},\pi)$ satisfies the following equation:*

$$\frac{\partial}{\partial\pi}p(\boldsymbol{z},\pi) = -\nabla_{\boldsymbol{z}}\cdot(p(\boldsymbol{z},\pi)v(\boldsymbol{z},\pi)). \tag{10.7}$$

This differential equation is similar to many differential equations of various conservation laws, where $v(\boldsymbol{z}, \pi)$ serves as a velocity term to characterize the movement of equilibria. In the following theorem, we use the equilibrium flow $v(\boldsymbol{z}, \pi)$ to address the dependency of $\mathcal{O}(\pi)$ on π.

Theorem 19. *If $v(\boldsymbol{z}^*, \pi)$ is the equilibrium flow of the stochastic equilibrium oracle $\mathcal{O}(\pi)$, we have:*

$$\frac{d}{d\pi} \mathop{\mathbb{E}}_{\boldsymbol{z}^* \sim \mathcal{O}(\pi)} f(\boldsymbol{z}^*, \pi) = \mathop{\mathbb{E}}_{\boldsymbol{z}^* \sim \mathcal{O}(\pi)} \left[f_\pi(\boldsymbol{z}^*, \pi) + f_{\boldsymbol{z}}(\boldsymbol{z}^*, \pi) \cdot v(\boldsymbol{z}^*, \pi) \right]. \tag{10.8}$$

Proof sketch. To compute the derivative on the left-hand side, we can expand the expectation by:

$$\begin{aligned} \frac{d}{d\pi} \mathop{\mathbb{E}}_{\boldsymbol{z}^* \sim \mathcal{O}(\pi)} f(\boldsymbol{z}^*, \pi) &= \frac{d}{d\pi} \int f(\boldsymbol{z}, \pi) p(\boldsymbol{z}, \pi) d\boldsymbol{z} \\ &= \int p(\boldsymbol{z}, \pi) \frac{\partial}{\partial \pi} f(\boldsymbol{z}, \pi) + f(\boldsymbol{z}, \pi) \frac{\partial}{\partial \pi} p(\boldsymbol{z}, \pi) d\boldsymbol{z} \\ &= \mathop{\mathbb{E}}_{\boldsymbol{z}^* \sim \mathcal{O}(\pi)} f_\pi(\boldsymbol{z}^*, \pi) + \int f(\boldsymbol{z}, \pi) \frac{\partial}{\partial \pi} p(\boldsymbol{z}, \pi) d\boldsymbol{z}. \end{aligned} \tag{10.9}$$

We substitute the term $\frac{\partial}{\partial \pi} p = -\nabla_{\boldsymbol{z}} \cdot (p \cdot v)$ by the definition of equilibrium flow, and apply integration by parts and Stokes' theorem[†] to the right-hand side of Equation 10.9 to get Equation 10.8. More details can be found in the appendix. □

Theorem 19 extends the derivative of Nash equilibrium to the case of stochastic equilibrium randomly drawn from multiple equilibria. Specifically, Equation 10.9 offers an efficient unbiased gradient estimate by sampling an equilibrium from the stochastic oracle to compute the right-hand side of Equation 10.9. Theorem 19 also matches to Equation 10.3, where the role of equilibrium flow $v(\boldsymbol{z}^*, \pi)$ coincides with the role of $\frac{d\boldsymbol{z}^*}{d\pi}$ in Equation 10.3.

[†]The analysis of integration by parts and Stokes' theorem applies to both Riemann and Lebesgue integral. Lebesgue integral is needed when the set of equilibria forms a measure-zero set.

10.5.3 How to Determine Equilibrium Flow

The only remaining question is how to determine the equilibrium flow. Given the leader's strategy π, there are two types of equilibria: (i) isolated equilibria and (ii) non-isolated equilibria. We first show that the solution to Equation 10.5 matches the equilibrium flow for every equilibrium in case (i) when the probability of sampling the equilibrium is locally fixed.

Theorem 20. *Given the leader's strategy π and a sampled equilibrium $\boldsymbol{z}$, if (1) the KKT matrix at $(\boldsymbol{z}, \pi)$ is invertible and (2) the equilibrium $\boldsymbol{z}$ is sampled with a fixed probability locally, the solution to Equation 10.5 is a homogeneous solution to Equation 10.7 and matches the equilibrium flow $v(\pi, \boldsymbol{z})$ locally.*

Theorem 20 ensures that when the sampled equilibrium behaves like a unique equilibrium locally, the solution to Equation 10.5 matches the equilibrium flow of the sampled equilibrium. In particular, Theorem 20 does not require all equilibria are isolated; it works as long as the sampled equilibrium satisfies the sufficient conditions. In contrast, the study in multiple equilibria requires global isolation for the analysis to work. Together with Theorem 19, we can use the solution to Equation 10.5 as an unbiased equilibrium gradient estimate and run stochastic gradient descent accordingly.

Lastly, when the sufficient conditions in Theorem 20 are not satisfied, e.g., the KKT matrix becomes singular for any non-isolated equilibrium, the solution to Equation 10.5 does not match the equilibrium flow $v(\boldsymbol{z}, \pi)$. In this case, to compute the equilibrium flow correctly, we rely on solving the partial differential equation in Equation 10.7. If the probability density function $p(\boldsymbol{z}, \pi)$ is explicitly given, we can directly solve Equation 10.7 to derive the equilibrium flow. If the probability density function $p(\boldsymbol{z}, \pi)$ is not given, we can use the empirical equilibrium distribution $p'(\boldsymbol{z}, \pi)$ constructed from the historical equilibrium samples of the oracle instead.

In practice, we hypothesize that even if the equilibria are not isolated and the corresponding KKT matrices are singular, solving degenerated version of Equation 10.5 still serves as a good approximation to the equilibrium flow. Therefore, we still use the solution to Equation 10.5 as an approximate of the equilibrium flow in the following sections and algorithms.

10.6 Gradient-Based Algorithm and Augmented Lagrangian Method

To solve both the optimization problems in Definition 15 and Definition 16, we implement our algorithm with (i) stochastic gradient descent with unbiased gradient access, and (ii) augmented Lagrangian method to handle the equilibrium-dependent constraints. We use the relaxation algorithm [307] as our equilibrium oracle $\mathcal{O}$. The relaxation algorithm is a classic equilibrium finding algorithm that iteratively updates agents' strategies by best responding to other agents' strategies until convergence with guarantees [179].

Since the leader's strategy π is constrained by the followers' response, we adopt an augmented Lagrangian method [43] to convert the constrained problem to an unconstrained one with a Lagrangian objective. We introduce a slack variable $\boldsymbol{s} \geq \mathbf{0}$ to convert inequality constraints into equality constraints $\mathbb{E}_{\boldsymbol{z}^* \sim \mathcal{O}(\pi)} g(\boldsymbol{z}^*, \pi) + \boldsymbol{s} = \mathbf{0}$. Thus, the penalized Lagrangian can be written as:

$$\mathcal{L}(\pi, \boldsymbol{s}; \lambda) = - \mathop{\mathbb{E}}_{\boldsymbol{z}^* \sim \mathcal{O}(\pi)} f(\boldsymbol{z}^*, \pi) + \lambda^\top (\mathop{\mathbb{E}}_{\boldsymbol{z}^* \sim \mathcal{O}(\pi)} g(\boldsymbol{z}^*, \pi) + \boldsymbol{s}) + \frac{\mu}{2} \left\| \mathop{\mathbb{E}}_{\boldsymbol{z}^* \sim \mathcal{O}(\pi)} g(\boldsymbol{z}^*, \pi) + \boldsymbol{s} \right\|^2 . \tag{10.10}$$

We run gradient descent on the minimization problem of the penalized Lagrangian $\mathcal{L}(\pi, \boldsymbol{s}; \lambda)$ and update the Lagrangian multipliers λ every fixed number of iterations, as described in Algorithm 11. The stochastic Stackelberg problem with multiple followers can be solved by running stochastic gradient descent with augmented Lagrangian methods, where Theorem 19 ensures the unbiasedness

Algorithm 11: Augmented Lagrangian Method

Initialization: $\pi = \pi_{\text{init}}$, learning rate γ, multipliers $\lambda = \lambda_0$, slack variable $\boldsymbol{s} \geq \mathbf{0}, K = 100$
for *iteration in* $\{1, 2, \dots\}$ **do**
  Define the objective to be Lagrangian $\mathcal{L}(\pi, \boldsymbol{s}; \lambda)$ defined in Equation 10.10
  Compute a sampled gradient of $\mathcal{L}$ by sampling $\boldsymbol{z}^* \sim \mathcal{O}(\pi)$. Compute $\frac{d\boldsymbol{z}^*}{d\pi}$ by Equation 10.5
  Update $\pi = \pi - \gamma(\frac{\partial \mathcal{L}}{\partial \pi} + \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}^*}\frac{d\boldsymbol{z}^*}{d\pi}), \boldsymbol{s} = \max\{\boldsymbol{s} - \gamma\frac{\partial \mathcal{L}}{\partial \boldsymbol{s}}, \mathbf{0}\}$
  if *iteration is a multiple of* K **then**
    Update $\lambda = \lambda - \mu(g(\boldsymbol{z}^*, \pi) + \boldsymbol{s})$
Return: leader's strategy π

of the stochastic gradient estimate under the conditions in Theorem 20.

10.7 Example Applications

We briefly describe three different Stackelberg games with one leader and multiple self-interested followers. Specifically, normal-form games with risk penalty has a unique Nash equilibrium, while other examples can have multiple.

10.7.1 Coordination in Normal-Form Games

A normal-form game (NFG) is composed of n follower players each with a payoff matrix $U_i \in \mathbb{R}^{m_1 \times \cdots \times m_n}$ for all $i \in [n]$, where the i-th player has m_i available pure strategies. The set of all feasible mixed strategies of player i is $z_i \in \mathcal{Z}_i = \{z \in [0,1]^{m_i} \mid \mathbf{1}^\top z = 1\}$. On the other hand, the leader can offer non-negative subsidies $\pi_i \in \mathbb{R}_{\geq 0}^{m_1 \times \cdots \times m_n}$ to each player i to reward specific combinations of pure strategies. The subsidy scheme is used to control the payoff matrix and incentivize the players to change their strategies.

Once the subsidy scheme π is determined, each player i chooses a strategy z_i and receives the ex-

pected payoff $U_i(\boldsymbol{z})$ and subsidy $\pi_i(\boldsymbol{z})$, subtracting a penalty term $H(z_i) = \sum_j z_{ij} \log z_{ij}$, the Gibbs entropy of the chosen strategy $\boldsymbol{z}_i$ to represent the risk aversion of player i. Since the followers' objectives are concave, the risk aversion model yields a unique Nash equilibrium, which is known to be quantal response equilibrium (QRE)[214,200]. Lastly, the leader's payoff is given by the social welfare across all players, which is the summation of the expected payoffs without subsidies: $\sum_{i \in [n]} U_i(\boldsymbol{z})$. The subsidy scheme is subject to a budget constraint B on the total subsidy paid to all players.

10.7.2 Security Games with Multiple Defenders

Stackelberg security games (SSGs) model a defender protecting a set of targets T from being attacked. We consider a scenario with a leader coordinator and n non-cooperative follower defenders each patrolling a subset $T_i \subseteq T$ of the targets[115]. Each defender i can determine the patrol effort spent on protecting the designated targets. We use $0 \leq z_{i,t} \leq 1$ to denote the effort spent on target $t \in T_i$ and the total effort is upper bounded by b_i. Defender i only receives a penalty $U_{i,t} < 0$ when target $t \in T_i$ in her protected region is attacked but unprotected by all defenders, and 0 otherwise.

Because the defenders are independent, the patrol strategies $\boldsymbol{z}$ can overlap, leading to a multiplicative unprotected probability $\prod_i (1 - z_{i,t})$ of target t. Given the unprotected probabilities, attacks occur under a distribution $\boldsymbol{p} \in \mathbb{R}^{|T|}$, where the distribution $\boldsymbol{p}$ is a function of the unprotected probabilities defined by a quantal response model. To encourage collaboration, the leader coordinator can selectively provide reimbursement $\pi_{i,t} \geq 0$ to alleviate defender i's loss when target t is attacked but unprotected, which encourages the defender to focus on protecting specific regions, reducing wasted effort on overlapping patrols. The leader's goal is to protect all targets, where the leader's objective is the total return across over all targets $\sum_{t \in T} U_t p_t \prod_i (1 - z_{i,t})$. Lastly, the reimbursement scheme π must satisfy a budget constraint B on the total paid reimbursement.

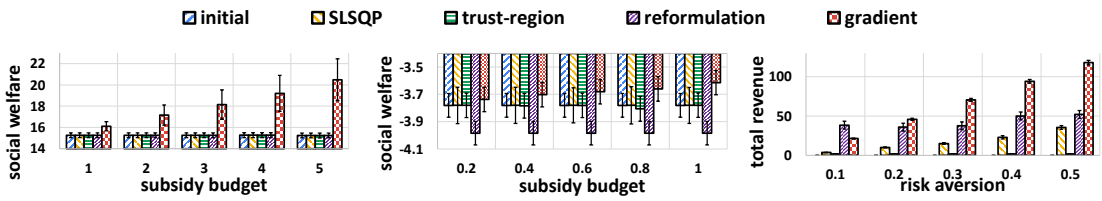
10.7.3 Cyber Insurance Games With Multiple Customers

We adopt the cyber insurance model proposed by Naghizadeh et al.[228] and Johnson et al.[154] to study how agents in an interconnected cyber security network make decisions, where agents' decisions jointly affect each other's risk. There are n agents (followers) facing malicious cyberattacks. Each agent i can deploy an effort of protection $z_i \in \mathbb{R}_{\geq 0}$ to his computer system, where investing in protection incurs a linear cost $c_i z_i$. Given the efforts $\boldsymbol{z}$ spent by all the agents, the joint protection of agent i is $\sum_{j=1}^{n} w_{ij} z_j$ with an interconnected effect parameterized by weights $W = \{w_{ij}\}_{i,j \in [n]}$. The probability of being attacked is modeled by $\sigma(-\sum_{j=1}^{n} w_{ij} z_j + L_i)$, where σ is the sigmoid function and L_i refers to the value of agent i.

The Stackelberg leader is an external insurer who can customize insurance plans to influence agents' protection decisions. The leader can set an insurance plan $\pi = \{I_i, \rho_i\}_{i \in [n]}$ to agent i, where ρ_i is the premium paid by agent i to receive compensation I_i when attacked. Under the insurance plans and the interconnected effect, agents selfishly determine their effort spent on the protection $\boldsymbol{z}$ to maximize their payoff. On the other hand, the leader's objective is the total premium subtracting the compensation paid, while the constraints on the feasible insurance plans are the individual rationality of each customer, i.e., the compensation and premium must incentivize agents to purchase the insurance plan by making the payoff with insurance no worse than the payoff without. These constraints restrict the premium and compensation offered by the insurer.

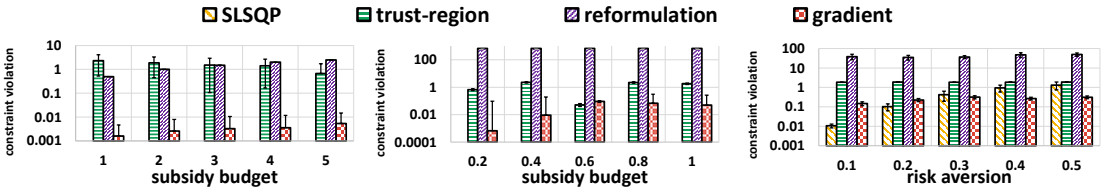
10.8 Experiments and Discussion

We compare our gradient-based Algorithm 11 (**gradient**) against various baselines in the three settings described above. In each experiment, we execute 30 independent runs (100 runs for SSGs) under different randomly generated instances. We run Algorithm 11 with learning rate $\gamma = 0.01$ for



(a) Normal-form games **(b)** Multi-defender SSGs **(c)** Cyber insurance games

Figure 10.3: We plot the solution quality of the Stackelberg problems with multiple followers. In all three domains, our gradient-based method achieves significantly higher objective than all other approaches. In NFGs and SSGs, the baselines cannot meaningfully improve upon the default strategy of the leader's initialization due to the high dimensionality of the parameter π; in cyber insurance games, SLSQP and reformulation both make some progress but still mostly with lower utility.



(a) Normal-form games **(b)** Multi-defender SSGs **(c)** Cyber insurance games

Figure 10.4: We plot the average budget constraint violation. Our gradient-based approach maintains low violation across all settings. SLSQP produces no violation in the first two domains because it fails to provide any meaningful improvement against the leader's initialization. Other baselines violate constraints more despite less performance improvement.

5,000 gradient steps and update the Lagrange multipliers every $K = 100$ iterations. Our gradient-based method completes in about an hour across all settings—refer to the appendix for more details.

Baselines We compare against several baselines that can solve the stochastic Stackelberg problem with multiple followers with equilibrium-dependent objective and constraints. In particular, given the non-convexity of agents' objective functions, SSGs and cyber insurance games can have multiple, stochastic equilibria. Our first baseline is the leader's **initial** strategy π_0, which is a naive all-zero strategy in all three settings. Blackbox optimization baselines include sequential least squares programming (**SLSQP**)[178] and the **trust-region** method[72], where the equilibrium encoded in the optimization problem is treated as a blackbox that needs to be repeatedly queried. **Reformulation**-based

algorithm [40,26] is the state-of-the-art method to solve Stackelberg games with multiple followers. This approach reformulates the followers' equilibrium conditions into non-linear complementary constraints as a mathematical program with equilibrium constraints [205], then solves the problem using branch-and-bound and mixed integer non-linear programming (we use a commercial solver, Knitro [238]). The reformulation-based approach cannot handle arbitrary stochastic equilibria but can handle optimistic or pessimistic equilibria. We implement the optimistic version of the reformulation as our baseline, which could potentially suffer from a performance drop as exemplified in Theorem 18.

10.8.1 Solution Quality

In Figure 10.3(a) and 10.3(b), we plot the leader's objective (y-axis) versus various budgets for the paid subsidy (x-axis). Figure 10.3(c), shows the total revenue to the insurer (y-axis) versus the risk aversion of agents (x-axis). Denoting the number of agents by n and the number of actions per agent by m, we have $n = 3, 5, 10$ and $m = 10, 50, 1$ in NFGs, SSGs, and cyber insurance games, respectively.

Our optimization baselines perform poorly in Figure 10.3(a) and 10.3(b) due to the high dimensionality of the environment parameter π in NFGs ($\dim(\pi) = nm^n$) and SSGs ($\dim(\pi) = nm$), respectively. In Figure 10.3(c), the dimensionality of cyber insurance games ($\dim(\pi) = 2n$) is smaller, where we can see that SLSQP and reformulation-based approaches start making some progress, but still less than our gradient-based approach. The main reason that blackbox methods do not work is due to the expensive computation of numerical gradient estimates. On the other hand, reformulation method fails to handle the mixed-integer non-linear programming problem reformulated from followers' best response and the constraints within a day.

10.8.2 Constraint Violation

In Figure 10.4, we provide the average constraint violation across different settings. Blackbox optimization algorithms either become stuck at the initial point due to the inexact numerical gradient estimate or create large constraint violations due to the complexity of equilibrium-dependent constraints. The reformulation approach also creates large constraint violations due to the difficulty of handling large number of non-convex followers' constraints under high-dimensional leader's strategy. In comparison, our method can handle equilibrium-dependent constraints by using an augmented Lagrangian method with an ability to tighten the budget constraint violation under a tolerance as shown. Although Figure 10.4 only plots the budget constraint violation, in our algorithm, we enforce that the equilibrium oracle runs until the equilibrium constraint violation is within a small tolerance 10^{-6}, whereas other algorithms sometimes fail to satisfy such equilibrium constraints.

10.9 Conclusion

In this paper, we present a gradient-based approach to solve Stackelberg games with multiple followers by differentiating through followers' equilibrium to update the leader's strategy. Our approach generalizes to stochastic gradient descent when the equilibrium reached by followers is stochastically chosen from multiple equilibria. We establish the unbiasedness of the stochastic gradient by the equilibrium flow derived from a partial differential equation. To our knowledge, this work is the first to establish the unbiasedness of gradient computed from stochastic sample of multiple equilibria. Empirically, we implement our gradient-based algorithm on three different examples, where our method outperforms existing optimization and reformulation baselines.

11

Equilibrium Refinement in Security Games with Arbitrary Scheduling Constraints

11.1 Introduction

Stackelberg Security Games (SSG) have been successfuly applied in a variety of domains to optimize the use of limited security resources against a strategic adversary, with examples such as ARMOR

for airport security[252], IRIS for security of flights[146], ports[281] and border[58,173] patrolling, traffic enforcement[269,268], and transit network[310]. In SSG, the defender (security agencies) protects targets using limited security resources, but allocation of resources to targets must obey many scheduling constraints. For example, some resources may be prohibited from being assigned to certain targets or may be able to cover several targets at the same time. After conducting surveillance of the defender strategy, the strategic attacker (terrorists/criminals) may respond with an optimal attack.

The standard solution concept adopted by SSG is the Strong Stackelberg Equilibrium (SSE)[189,315]. Significant research in SSG has focused on providing efficient algorithms to compute SSE under various constraints[146,282]. Recently, significant research efforts have focused on devising strategies that perform well even under uncertainty in the adversary behavior. For example,[351,236] investigate adversary bounded rationality,[151] considers execution uncertainty, and[353] focuses on observational uncertainty. In most of these frameworks, the defender either pays a price or slightly sacrifices her first priority target to ensure robustness against unpredictability in the adversary's behavior. However, *equilibrium refinement* is an attractive alternative to provide robustness at no cost by choosing, among all SSEs, the one that performs best in all possible events although it has not received as much attention in the security game literature.

In most real-world applications, security resources must be allocated in the presence of scheduling constraints. This is the case for example of the Federal Air Marshal Service[146], cyber security[283,235], network security[168,272], and more generally in domains where security resources exhibit protection externalities[113,84]. Yet, existing algorithms for equilibrium refinement in security games do not apply in the presence of such constraints. The presence of scheduling constraints complicates the problem of equilibrium refinement significantly, since multiple equilibria are the norm for security games with schedules, and even finding an arbitrary SSE[176] is already a challenging task and prevents the adoption of existing techniques[15] in our problem. To the best of our knowledge, the only paper to investigate the problem of equilibrium refinement under scheduling constraints is[93], wherein a

heuristic algorithm is proposed to conduct equilibrium refinement in the spatio-temporal domain. While the paper provides a significant step in our research direction, it only addresses a special case of scheduling constraints. In fact, we are not aware of any algorithm that can cater for *arbitrary scheduling constraints* in security games to provide an optimal refined equilibrium.

In this paper, we focus on the equilibrium refinement on Security Problems with ARbitrary Schedules (SPARS) [146] , where we assign each resource to cover one schedule and each schedule can cover multiple targets. We follow the same dominance criteria mentioned in [15] and introduce a counterexample showing that in the presence of scheduling constraints, their method fails to return a non-dominated equilibrium. We propose a new method to analyze the topology of the attacker's best response. This analysis provides us with key insights into the structure of multiple equilibria. Leveraging these insights, we introduce a new iterative (resp. recursive) algorithm that successfully returns the non-dominated solution of zero-sum (resp. general-sum) SPARS. We show that in the worst case, our iterative algorithm only necessitates $O(n^3)$ calls to an LP oracle, where n corresponds to the number of targets and an LP oracle could be either a linear program solver or a column generation method used to approximate the optimal solution. For the general-sum games, our recursive algorithm successfully provides the optimal solution with $O(n^3)$ oracle calls for each subproblem.

Our experimental results demonstrate significant improvement on the robustness of our computed solution over existing approaches which also serves to showcase the benefit of equilibrium refinement on SPARS. Moreover, our computations show the average number of oracle calls is $O(n^2)$ in both zero-sum and general-sum cases, illustrating practical scalability of our approach.

11.2 Security Games with Arbitrary Schedules

In this work, we consider SPARS [146]. This is a two-player Stackelberg game played between an attacker and a defender. The attacker's pure strategy space is the set of targets T that could be attacked,

$T = \{t_1, \ldots, t_n\}$. The attacker's corresponding mixed strategy $\mathbf{a} = \langle a_i \rangle_{i=1}^n$ is a vector where a_i represents the probability of attacking t_i. To protect targets, the defender has at her disposal a collection of resources indexed by $r \in R$, where the set R collects all resources. Each resource r can be assigned to a *schedule* $s \subseteq T$ that covers multiple targets. Associated with each resource r is the set of all possible schedules $S_r \subseteq \mathcal{P}(T)$ to which it can be assigned. For notational convenience, we assume that $\emptyset \in S_r$ so that a resource that is assigned to $\emptyset$ is effectively unused.

The defender's pure strategy space J is the set of all joint schedules that assign each resource to exactly one schedule. Thus,

$$J = \{j \subseteq T \,:\, j = \cup_{r \in R} s_r,\ s_r \in S_r\}$$

and target $t \in T$ is covered by the joint schedule $j \in J$ if and only if $t \in j$. For any joint schedule, a target can be covered by more than one schedule, and a target is considered covered (or protected) whenever the total number of resources allocated to a schedule that covers the target equals or exceeds one (1).

Associated with each joint schedule $j \in J$ is a vector $\mathbf{P}_j = \langle P_{jt} \rangle \in \{0,1\}^n$, where P_{jt} indicates whether target t is covered in joint schedule j, i.e., $P_{jt} = \mathbb{I}(t \in j)$. The defender's mixed strategy $\mathbf{x}$ specifies the probabilities of playing each $j \in J$, where $x_j \geq 0, \sum_{j \in J} x_j = 1$. Let $\mathbf{c} = \langle c_t \rangle_{t=1}^n$ be the vector of coverage probabilities corresponding to $\mathbf{x}$, where $c_t = \sum_{j \in J} P_{jt} x_j$, is the marginal probability of covering t and we can write $\mathbf{c} = \mathbf{P}^\top \mathbf{x}$.

The payoffs of players are decided by the target chosen by the attacker and whether the target is protected by the defender. The defender's payoff for an uncovered attack on target t is denoted by $U_d^u(t)$ and for a covered attack $U_d^c(t)$. Similarly, $U_a^u(t)$ and $U_a^c(t)$ are the attacker's payoffs for the uncovered and covered cases, respectively. A widely adopted assumption in security games is that $U_d^c(t) > U_d^u(t)$ and $U_a^u(t) > U_a^c(t)$. In other words, covering an attack is beneficial for the defender,

while hurts the attacker. Given a strategy profile $\langle \mathbf{x}, \mathbf{a} \rangle$, $\mathbf{c} = \mathbf{P}^\top \mathbf{x}$, the expected utilities for both players are denoted as follows:

$$U_d(\mathbf{c}, \mathbf{a}) = \sum_{t \in T} a_t U_d(\mathbf{c}, t), \text{ where } U_d(\mathbf{c}, t) = c_t U_d^c(t) + (1 - c_t) U_d^u(t)$$
$$U_a(\mathbf{c}, \mathbf{a}) = \sum_{t \in T} a_t U_a(\mathbf{c}, t), \text{ where } U_a(\mathbf{c}, t) = c_t U_a^c(t) + (1 - c_t) U_a^u(t)$$

We adopt a Stackelberg model in which the defender acts first and the attacker chooses a strategy after observing the defender's mixed strategy. Stackelberg games are common in security domains where attackers can surveil the defender strategy. The standard solution concept is SSE [189,315], in which the leader selects an optimal mixed strategy based on the assumption that the follower will choose an optimal response, breaking ties in favor of the leader. There always exists an optimal pure-strategy response for the attacker, so we restrict our attention to this set in this paper.

11.3 Refinement of Strong Stackelberg Equilibrium in Security Games

A well-known property of SSE is that all SSEs give the same expected payoff for the leader (defender) [52,189]. The refinement of SSEs in security games is first discussed in [15]. They indicate that multiple equilibria exist frequently (especially when there are resources, scheduling constraints) and in many of these solutions, a portion of the resources are not efficiently used since they can be abandoned without affecting the expected utility. We follow the same dominance criteria in [15]. The defender assumes there is an infinitesimal probability that the attacker will deviate from his first choice to his second or other preferable targets due to some unexpected events. But, even when the attacker is forced to deviate, he still behaves intelligently by choosing the next-best alternative rather than acting randomly. Therefore, the defender will still need to efficiently arrange the remaining resources to achieve her highest defender utilities, sequentially, on the secondary targets.

Based on this model, our equilibrium concept can be written as following: Given an SSE $\langle \mathbf{x}, \mathbf{a} \rangle$

and its coverage vector $\mathbf{c}$, an ordering over targets is defined such that target $t(1)$ is the target that will be attacked by the unconstrained attacker, and $t(i)$ is the target that will be attacked by the constrained attacker who cannot attack targets $t(1), \ldots, t(i-1)$. Utility vector $\mathbf{v} = \langle v_i \rangle_{i=1}^{n}$ represents the defender's utilities where v_i is the defender's utility if target $t(i)$ is attacked, i.e., $v_i = c_{t(i)} U_d^c(t(i)) + (1 - c_{t(i)}) U_d^u(t(i))$. They define a dominance relation between SSEs based on the utility vectors (if there is no ambiguity, we will consistently use coverage vector $\mathbf{c}$ to refer the defender's strategy $\mathbf{x}$).

Definition 18. *Given two SSEs $\langle \mathbf{c}, \mathbf{a} \rangle, \langle \mathbf{c}', \mathbf{a}' \rangle$ and their utility vectors $\mathbf{v}$ and $\mathbf{v}'$. We say that SSE $\langle \mathbf{c}, \mathbf{a} \rangle$* **dominates** *SSE $\langle \mathbf{c}', \mathbf{a}' \rangle$ if there exists i such that* i) $v_i > v_i'$ *and* ii) $v_j = v_j'$ *for all j such that* $1 \leq j < i$.

There is an iterative algorithm [15] which can find the non-dominated SSE in the security games without scheduling constraints. In those cases, the multiple SSEs only exist when the best response target of the attacker is fully covered. In the security games with scheduling constraints, multiple SSEs are more common which motivates further needs for refinement. Unfortunately, in the presence of scheduling constraints, the method in [15] may return a dominated SSE, as illustrated by the following example.

Example 1 (Dominated SSEs in zero-sum SPARS games)**.** *Consider a zero-sum game with one resource $R = \{r_1\}$, three targets $T = \{t_1, t_2, t_3\}$, three schedules $S_1 = \{s_1, s_2, s_3\}$:*

$$s_1 = \{t_1, t_3\}, s_2 = \{t_2\}, s_3 = \{t_3\}$$

and with the following payoffs: $U_a^c(t) = U_d^c(t) = 0 \, \forall \, t \in T$

$$U_d^u(t_1) = -3, U_d^u(t_2) = -3, U_d^u(t_3) = -6, \qquad U_a^u(t) = -U_d^u(t) \, \forall t \in T$$

There are infinite SSE solutions. One possible SSE could be $\mathbf{x}^1 = \langle \frac{1}{3}, \frac{1}{3}, \frac{1}{3} \rangle$ *with the corresponding coverage vector* $\mathbf{c}^1 = \langle \frac{1}{3}, \frac{1}{3}, \frac{2}{3} \rangle$*. The unsorted defender's utility vector is given by* $\mathbf{d}^1 = \langle -2, -2, -2 \rangle$ *for targets* t_1, t_2, t_3*. Accordingly, the sorted utility vector is given by* $\mathbf{v}^1 = \langle -2, -2, -2 \rangle$*. In this case,* $\mathbf{v}^1$ *and* $\mathbf{d}^1$ *are the same because the attacker feels indifferent between all of the targets. Applying the iterative algorithm from* [15]*, given the arbitrary SSE* $\mathbf{x}_1$*, first we fix the coverage of one target among those with the highest attacker expected utility (in this case* $\{t_1, t_2, t_3\}$*). We assume the algorithm chooses target* t_1 *with* $c_1 = \frac{1}{3}$ *fixed and solves it iteratively, which returns the same strategy* $\mathbf{x}^1$*. However, the strategy* $\mathbf{x}^1$ *is dominated by strategy* $\mathbf{x}^2 = \langle \frac{2}{3}, \frac{1}{3}, 0 \rangle$ *with coverage* $\mathbf{c}^2 = \langle \frac{2}{3}, \frac{1}{3}, \frac{2}{3} \rangle$ *providing a better defender's utility vector* $\mathbf{d}^2 = \langle -1, -2, -2 \rangle$ *and* $\mathbf{v}^2 = \langle -2, -2, -1 \rangle$ *sorted by the attacker's preference. Both* $\mathbf{x}^1, \mathbf{x}^2$ *provide the highest defender's utility* $d^* = -2$*.*

Example 1 shows that a non-dominated solution can perform significantly better than an arbitrary chosen SSE. In this case, if the attacker deviates from his best response (target t_2, t_3) to the third preferable target (target t_1), the defender's utility will be -2 and -1 for strategies $\mathbf{x}^1$ and $\mathbf{x}^2$, respectively, yielding a 50% difference between refined and arbitrary SSE. *

11.4 Zero-sum Games

In this paper, we start with zero-sum games where the attacker is completely opposite to the defender. We define the idea of minimal attack set, prove its uniqueness, and show the SSE with minimal attack set is better than all the other SSEs. We also show that the minimal attack set can be computed by a polynomial number of calls to an oracle that solves linear programs. Accordingly, we propose an algorithm which iteratively solves the minimal attack set of restricted instance and fixes

*One intuitive heuristic algorithm of refinement, in the presence of constraints, is to eliminate those inefficient schedules [93]. E.g., in the context of Example 1, schedule $s_1 = \{t_1\}$ is dominated by $s_3 = \{t_1, t_3\}$. However, in the experiment part, we will show that the heuristic method provides only a little improvement in the zero-sum games and it does not work in the general-sum games.

the coverage on the minimal attack set. We prove that our algorithm requires at most $O(n^3)$ oracle calls and returns a non-dominated SSE.

11.4.1 Uniqueness of Minimal Attack Set of SSE

Definition 19. *Given a feasible SSE coverage vector* $\mathbf{c}$*, the* ***Attack Set*** $\Gamma(\mathbf{c}) := \arg\max_{t \in T} U_a(\mathbf{c}, t)$ *is the best response of the attacker.*

Definition 20. *Let* $\Psi := \{T' \subseteq T \mid \exists\, SSE\ \langle \mathbf{c}, \mathbf{a} \rangle : \Gamma(\mathbf{c}) = T'\}$ *be the set of all possible attack sets of SSEs.*

In zero-sum games, the less optimal choices of the attacker (attack set) imply the less targets that he can achieve his highest utility. Thus, for the defender, the SSE with a smaller attack set is always better than the SSE with a larger attack set.

Definition 21. *A* ***Minimal Attack Set*** *is a set* $M \in \Psi$ *such that any proper subset* V *of* M *is not an element of* Ψ*, i.e.,* $V \notin \Psi$ *for all* $V \subset M$*.*

Example 2. *Consider a zero-sum game with one resource* $R = \{r_1\}$*,* $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$*, and four schedules* $S_1 = \{s_1, s_2, s_3, s_4\}$*:*

$$s_1 = \{t_1, t_2, t_3\}, s_2 = \{t_2, t_3, t_4\}, s_3 = \{t_3, t_4, t_5\}, s_4 = \{t_6\}$$

with the following payoffs: $U_d^c(t) = 0\ \forall\, t \in T$

$$U_d^u(t_1) = -4, U_d^u(t_2) = -4, U_d^u(t_3) = -12$$
$$U_d^u(t_4) = -4, U_d^u(t_5) = -2, U_d^u(t_6) = -4$$

There are infinite possible SSE solutions. For example, one possible SSE is $\mathbf{x}^1 = \langle \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4} \rangle$ *with coverage* $\mathbf{c}^1 = \langle \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4} \rangle$ *which gives the defender's utility vector* $\mathbf{d}^1 = \langle -3, -2, -3, -2, -1.5, -3 \rangle$ *with* $d^* = -3$*. In this case, the best response of the attacker is* $\Gamma(\mathbf{c}^1) = \{t_1, t_3, t_6\}$*. The following mixed strategies also apply.*

$$\mathbf{x}^2 = \langle \frac{1}{2}, \frac{1}{6}, \frac{1}{12}, \frac{1}{4} \rangle, \qquad \mathbf{c}^2 = \langle \frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \frac{1}{4}, \frac{1}{12}, \frac{1}{4} \rangle$$

$$\mathbf{d}^2 = \langle -2, -\frac{4}{3}, -3, -3, -\frac{11}{6}, -3 \rangle, \qquad \Gamma(\mathbf{c}^2) = \{t_3, t_4, t_6\}$$

$$\mathbf{x}^3 = \langle \frac{3}{8}, \frac{5}{24}, \frac{1}{6}, \frac{1}{4} \rangle, \qquad \mathbf{c}^3 = \langle \frac{3}{8}, \frac{7}{12}, \frac{3}{4}, \frac{3}{8}, \frac{1}{6}, \frac{1}{4} \rangle$$

$$\mathbf{d}^3 = \langle -2.5, -\frac{5}{3}, -3, -2.5, -\frac{5}{3}, -3 \rangle, \qquad \Gamma(\mathbf{c}^3) = \{t_3, t_6\}$$

$$\mathbf{x}^4 = \langle \frac{1}{4}, 0, \frac{1}{2}, \frac{1}{4} \rangle, \qquad \mathbf{c}^4 = \langle \frac{1}{4}, \frac{1}{4}, \frac{3}{4}, \frac{1}{2}, \frac{1}{2}, \frac{1}{4} \rangle$$

$$\mathbf{d}^4 = \langle -3, -3, -3, -2, -1, -3 \rangle, \qquad \Gamma(\mathbf{c}^4) = \{t_1, t_2, t_3, t_6\}$$

Clearly, all the above strategies are SSE solutions. But the strategy $\mathbf{x}^3$ *dominates all the others since the defender's utility on the third-preferable target of the attacker is* -2.5*, which is higher than all the others' utility* -3*. Actually,* $\mathbf{x}^3$ *is the non-dominated strategy.*

If we explore all of the possible SSEs in Example 2, we will find that the above attack sets are exactly all the possible attack sets:

$$\Psi = \{\{t_3, t_6\}, \{t_1, t_3, t_6\}, \{t_3, t_4, t_6\}, \{t_1, t_2, t_3, t_6\}\}$$

Therefore, the only minimal attack set is $\Gamma(\mathbf{c}^3) = \{t_3, t_6\}$*.*

Theorem 21 (Intersection Property in Zero-sum Games)**.** *For any two attack sets* $T^1, T^2 \in \Psi$*, we have* $T^1 \cap T^2 \neq \emptyset$ *and* $T^1 \cap T^2 \in \Psi$*.*

Proof. Given $T^1 = \Gamma(\mathbf{c})$, $T^2 = \Gamma(\mathbf{c}')$, there are two cases:

(1) $\Gamma(\mathbf{c}) \cap \Gamma(\mathbf{c}') \neq \emptyset$. Consider another strategy $\mathbf{c}^* = \alpha\mathbf{c} + (1-\alpha)\mathbf{c}'$ with $\alpha \in (0,1)$. Since $\mathbf{c}^* = \alpha\mathbf{c} + (1-\alpha)\mathbf{c}' = \alpha\mathbf{P}^\top\mathbf{x} + (1-\alpha)\mathbf{P}^\top\mathbf{x}' = \mathbf{P}^\top(\alpha\mathbf{x} + (1-\alpha)\mathbf{x}')$, $\mathbf{c}^*$ is a feasible coverage vector of strategy $\mathbf{x}^*$. It is easy to verify that $\Gamma(\mathbf{c}^*) = \Gamma(\mathbf{c}) \cap \Gamma(\mathbf{c}')$ as follows:

$$U_a(\mathbf{c}, t) \begin{cases} = v & \text{if } t \in \Gamma(\mathbf{c}) \\ < v & \text{otherwise} \end{cases} \qquad U_a(\mathbf{c}', t) \begin{cases} = v & \text{if } t \in \Gamma(\mathbf{c}') \\ < v & \text{otherwise} \end{cases}$$

$$U_a(\mathbf{c}^*, t) = \alpha U_a(\mathbf{c}, t) + (1-\alpha) U_a(\mathbf{c}', t) \begin{cases} = v & \text{if } t \in \Gamma(\mathbf{c}) \cap \Gamma(\mathbf{c}') \\ < v & \text{otherwise} \end{cases}$$

where v is the expected attacker's utility. Thus, we obtain an SSE strategy $\mathbf{c}^*$ with a smaller attack set $\Gamma(\mathbf{c}) \cap \Gamma(\mathbf{c}')$.

(2) $\Gamma(\mathbf{c}) \cap \Gamma(\mathbf{c}') = \emptyset$. Similarly, we consider the feasible strategy $\mathbf{c}^* = \alpha\mathbf{c} + (1-\alpha)\mathbf{c}'$ with $\alpha \in (0,1)$. It is easy to verify that $U_a(\mathbf{c}^*, t) < v$ for any $t \in T$. In other words, $U_d(\mathbf{c}^*, t) > -v$ where $-v$ is the highest expected utility of defender in SSE. This contradicts the optimality of SSE. That means this case will never happen. □

Consider the SSE strategy $\mathbf{x}^3$ in Example 2. It can be written as the combination of SSE strategies $\mathbf{x}^1, \mathbf{x}^2$ by $\mathbf{x}^3 = \frac{1}{2} \cdot \mathbf{x}^1 + \frac{1}{2} \cdot \mathbf{x}^2$. As Theorem 21 states, the attack set $\Gamma(\mathbf{c}^3) = \Gamma(\mathbf{c}^1) \cap \Gamma(\mathbf{c}^2)$.

Theorem 22. *The minimal attack set M exists and is unique. Moreover, for each $T' \in \Psi$, $M \subseteq T'$.*

Proof. (1) Existence: Clearly, $M = \bigcap_{T' \in \Psi} T' \neq \emptyset$ is a minimal attack set. (2) Uniqueness: If there are two different minimal attack sets, then by Theorem 21, their intersection will be non-empty and is a smaller attack set, which is a contradiction. □

In Example 2, the minimal attack set is exactly $\Gamma(\mathbf{c}^3) = \{t_3, t_6\}$ which is the attack set of the non-dominated solution $\mathbf{x}^3$.

11.4.2 An Iterative Algorithm

In zero-sum games, an SSE with a smaller attack set is better (for the defender) than an SSE with a larger attack set. This motivates us to find the minimal attack set. Moreover, the minimal attack set is included in every attack set, which implies that we can fix the common coverage on the minimal attack set and solve the remaining subproblem. For this aim, we define the *restricted* SPARS.

Definition 22. *Given a SPARS instance g, we denote by $g^{\mathbf{c},T'}$ the restricted game with respect to coverage vector $\mathbf{c}$ and $T' \subset T$. The* **restricted SPARS** *instance $g^{\mathbf{c},T'}$ is the same as SPARS instance g except the following rules:*

(R1) The attacker is forbidden to attack targets in T'.

(R2) The defender's coverage on $t \in T'$ is fixed to be c_t.

(R3) The defender must cover targets $t \in T \setminus T'$ enough such that the attacker utility on these targets is at most $\min_{t' \in T'} U_a(\mathbf{c}, t')$.

The SSE in a restricted game follows the same definition as in the original SPARS. Rule (R1) guarantees that the attacker will only focus on targets $T \setminus T'$. Rule (R2) guarantees that solving the restricted SPARS will not alter the existing coverage on T' which is already known. Rule (R3) requires the defender to cover targets $t \in T \setminus T'$ enough such that the targets in $T \setminus T'$ are not more preferable for the attacker than those in T'. In addition, we define the **restricted attack set** by $\Gamma(\mathbf{c}', T') = \operatorname{argmax}_{t \in T \setminus T'} U_a(\mathbf{c}', t)$, where c' is a feasible solution to $g^{c,T'}$. Note that the restricted attack set is the attack set for the restricted instance $g^{\mathbf{c},T'}$, thus they share the same properties of attack sets. Accordingly, we can define the **minimal restricted attack set** for the restricted instance.

Algorithm 12: Iterative Algorithm for Zero-sum Games

```
Parameter: SPARS instance g, T' ← ∅, c ← 0
while |T'| < |T| do
    c ← a restricted SSE strategy in the instance g^{c,T'}
    M ← the minimal restricted attack set of instance g^{c,T'}
    T' ← T' ∪ M
Return: non-dominated SSE strategy c
```

Algorithm 12 depicts the procedure of equilibrium refinement in zero-sum games. We compute an arbitrary SSE strategy, fix the coverage on the minimal restricted attack set (we will discuss how to find the minimal attack set in Section 11.4.3), and iteratively solve the remaining restricted sub-problem. The following theorems guarantee the correctness of Algorithm 12.

Proposition 8. *Given a restricted SPARS instance $g^{\mathbf{c},T'}$, its minimal restricted attack set M, and an SSE $\mathbf{c}^*$ of $g^{\mathbf{c},T'}$, we have the following statement: the strategy $\mathbf{c}'$ is a feasible defender coverage of $g^{\mathbf{c}^*,T'\cup M}$ (satisfies Rules (R2), (R3)) if and only if $\mathbf{c}'$ is an SSE of $g^{\mathbf{c}^*,T'}$, which provides a mapping between two different restricted instances.*

Proof. ($\Leftarrow$) Since both $\mathbf{c}'$ and $\mathbf{c}^*$ are SSE strategies and M is the minimal restricted attack set of $g^{\mathbf{c},T'}$, both $\mathbf{c}'$ and $\mathbf{c}^*$ share the same value on $T' \cup M$, which satisfies the Rule (R2) of $g^{\mathbf{c}^*,T'\cup M}$.

Since $\mathbf{c}'$ is an SSE of $g^{\mathbf{c},T'}$, the attacker's utility on $t \in T'$ with SSE strategy $\mathbf{c}'$ must be greater than all the others $t \notin T'$. By the definition of minimal restricted attack set M, the best response of the attacker, it implies that target $t \in M$ must have the highest attacker's utility among $T \backslash T'$. Therefore, the attacker utility on $t \in T' \cup M$ is no less than the others' utilities, which satisfies Rule (R3) of $g^{\mathbf{c}^*,T'\cup M}$.

($\Rightarrow$) Assume that $\mathbf{c}'$ is a solution of $g^{\mathbf{c}^*,T'\cup M}$. By the definition of restricted instance $g^{\mathbf{c}^*,T'\cup M}$, the coverage $\mathbf{c}'$ on targets in $T' \cup M$ has been fixed to be the same as $\mathbf{c}^*$, and Rule (R3) forces all the other targets outside of $T' \cup M$ to have a smaller attacker's utility. It implies that the strategy $\mathbf{c}'$

achieves the highest attacker's utility on minimal attack set M in the restricted instance $g^{\mathbf{c},T'}$, thus an SSE in the restricted instance $g^{\mathbf{c},T'}$. □

Theorem 23. *The output of Algorithm 12 is a non-dominated SSE.*

Proof. Denote the sequences of minimal restricted attack sets and updated coverage in Algorithm 12 as $M_1, \ldots, M_k$ and $\mathbf{c}^1, \ldots, \mathbf{c}^k$, respectively (W.L.O.G let $M_0 = \emptyset, \mathbf{c}^0 = 0$). According to Algorithm 12, $\mathbf{c}^i$ is an SSE, M_i is the minimal attack set of $g^{\mathbf{c}^{i-1}, M_1 \cup \ldots \cup M_{i-1}}$.

By Proposition 8, $\forall i \in \{1, 2, \ldots, k\}$ we have: given a restricted instance $g^{\mathbf{c}^{k-i}, M_1 \cup M_2 \cup \ldots \cup M_{k-i}}$, its minimal restricted attack set M_{k-i+1} and its SSE $\mathbf{c}^{k-i+1}$, the strategy $\mathbf{c}'$ is a feasible coverage of the instance $g^{\mathbf{c}^{k-i+1}, M_1 \cup M_2 \cup \ldots \cup M_{k-i} \cup M_{k-i+1}}$ if and only if $\mathbf{c}'$ is an SSE of the instance $g^{\mathbf{c}^{k-i+1}, M_1 \cup M_2 \cup \ldots \cup M_{k-i}}$.

According to the above argument, $\forall i \in \{1, 2, \ldots, k\}$ we have: $\mathbf{c}^k$ is the non-dominated solution of $g^{\mathbf{c}^{k-i+1}, M_1 \cup M_2 \cup \ldots \cup M_{k-i+1}} \Leftrightarrow \mathbf{c}^k$ is the non-dominated SSE of $g^{\mathbf{c}^{k-i+1}, M_1 \cup M_2 \cup \ldots \cup M_{k-i}} \Leftrightarrow \mathbf{c}^k$ is the non-dominated SSE of $g^{\mathbf{c}^{k-i}, M_1 \cup M_2 \cup \ldots \cup M_{k-i}}$ (since $\mathbf{c}^{k-i}$ and $\mathbf{c}^{k-i+1}$ share the same coverage on $M_1 \cup M_2 \cup \ldots \cup M_{k-i}$) $\Leftrightarrow \mathbf{c}^k$ is the non-dominated solution of $g^{\mathbf{c}^{k-i}, M_1 \cup M_2 \cup \ldots \cup M_{k-i}}$ (non-dominated solution must be an SSE). When $i = 1$, $\mathbf{c}^k$ is the only solution (thus non-dominated) solution of $g^{\mathbf{c}^k, M_1 \cup M_2 \cup \ldots \cup M_k}$ (since $M_1 \cup M_2 \cup \ldots \cup M_k = T$). By induction, $\mathbf{c}^k$ is the non-dominated solution of $g^{\mathbf{c}^{k-i}, M_1 \cup M_2 \cup \ldots \cup M_{k-i}}$. By letting $i = k$, the statement is exactly our conclusion: $\mathbf{c}^k$ is the non-dominated solution of g. □

11.4.3 Computing the Minimal Attack Set

In the previous section, we showed that a non-dominated SSE strategy can be obtained by iteratively computing SSE strategies of restricted SPARS instances and their corresponding minimal restricted

attack sets. We now propose a method for finding the unique minimal attack set.

$$\max_{\mathbf{c},\mathbf{x}} \quad U_d(\mathbf{c}, t) \tag{11.1a}$$

$$\text{s.t.} \quad U_a(\mathbf{c}, t) \geq U_a(\mathbf{c}, t') \quad \forall t' \neq t \tag{11.1b}$$

$$\mathbf{P}^\top \mathbf{x} = \mathbf{c} \tag{11.1c}$$

$$\sum\nolimits_{j \in J} x_j = 1. \tag{11.1d}$$

First, multiple LPs method[71] is commonly used to compute an SSE to security games. Each LP (11.1) corresponds to one target t and maximizes the defender's expected utility on this target under the restriction that t is in the best response for the attacker.

Definition 23. *Given target t, let N_t be the* **smallest tight constraint set** *with $N_t := \{t' \in T \mid \forall$ SSE strategies $\mathbf{c}$ with $t \in \Gamma(\mathbf{c}), U_a(\mathbf{c}, t) = U_a(\mathbf{c}, t'), U_d(\mathbf{c}, t) = U_d(\mathbf{c}, t')\}$.*

Given target t and its LP (11.1), we are interested in which constraints are necessary and always active for all optimal solutions (SSEs), which is the smallest tight constraint set N_t. Our main idea is to slightly alter the constraint of target t' in LP (11.1) to

$$U_a(\mathbf{c}, t) \geq U_a(\mathbf{c}, t') + \varepsilon$$

where ε is a small positive number (e.g., constant times of numerical error). If the modified version of the linear program still provides the same maximum objective value (up to numerical error), then the constraint with respect to t' is not active, which means $t' \notin N_t$. If it provides a smaller objective value or the linear program is infeasible, that means the constraint with respect to t' is always active, which implies $t' \in N_t$.

The procedure of Algorithm 13 is to find out the smallest tight constraint set N_t under the restriction that t is the best response of attacker. Every N_t can be solved by at most n modified linear

programs. We will show that the intersection of all the smallest tight constraint sets is exactly the minimal attack set.

Algorithm 13: Algorithm for Finding Minimal Attack Set

Parameter: SPARS instance $g^{\mathbf{c},T'}$
solve an SSE $\mathbf{c}^*$ using the multiple linear program method and record the primal, dual solution of each LP
for $t \in \Gamma(\mathbf{c}^*)$ **do**
  given the dual solution $\mathbf{d}'$ and primal solution $\mathbf{c}'$ of LP (11.1)
  $N_t \leftarrow A_t \leftarrow \{t' \mid d'_{t'} \neq 0\} \cup \{t\}, B_t \leftarrow \Gamma(\mathbf{c}')\backslash A_t$
  for $t' \in B_t$ **do**
    solve modified LP (11.1) with one more constraint $U_a(\mathbf{c}, t) \geq U_a(\mathbf{c}, t') + \varepsilon$
    if *the objective value changes* **then**
      $N_t \leftarrow N_t \cup \{t'\}$
Return: minimal restricted attack set $\bigcap_{t \in \Gamma(\mathbf{c}^*)} N_t$, coverage $\mathbf{c}^*$

Proposition 9. *Given the dual solution* $\mathbf{d}'$ *of LP* (11.1)*, the set* $\{t' | \mathbf{d}'_{t'} \neq 0\}$ *is contained in the smallest tight constraint set* N_t*.*

Proposition 10. *Given a primal solution* $\mathbf{c}'$ *of LP* (11.1)*, every target* $t \notin \Gamma(\mathbf{c}')$ *is not contained in the smallest tight constraint set.*

Proposition 9 and 10 help eliminate unnecessary enumerations in Algorithm 13. In the average case, there are only a constant number of targets in B_t (in Algorithm 13) needed to be enumerated But in the worst case, we still need to run through at most n targets. The following theorems guarantee correctness of Algorithm 13.

Proposition 11. *Given target t,* $N_t = \bigcap_{T' \in \Psi : t \in T'} T'$*. Moreover, given arbitrary SSE coverage* $\mathbf{c}'$*,* $\bigcap_{t \in \Gamma(\mathbf{c}')} N_t = \bigcap_{T' \in \Psi} T'$*, which is the minimal attack set M.*

Proof. First, for each SSE solution $\mathbf{c}$, the targets in $\Gamma(\mathbf{c})\backslash\{t\}$ are exactly the targets which make the constraints (11.1b) tight. Thus, the smallest tight constraints are the same as the intersection of attack sets containing t as the best response for the attacker, which implies $N_t = \bigcap_{T' \in \Psi : t \in T'} \Gamma(\mathbf{c})$. Second, since $\Gamma(\mathbf{c}')$ contains at least one target t in the minimal attack set, the minimal attack set M must appear in one of the $T' \in \Psi, t \in T'$, which implies $\bigcap_{t \in \Gamma(\mathbf{c}')} N_t = \bigcap_{t \in \Gamma(\mathbf{c}')} \bigcap_{T' \in \Psi : t \in T'} \Gamma(\mathbf{c}) = M = \bigcap_{T' \in \Psi} \Gamma(\mathbf{c})$. □

Theorem 24. *Algorithm 13 correctly returns the minimal restricted attack set of $g^{\mathbf{c},T'}$.*

We can employ Algorithm 13 in Algorithm 12 to find the minimal attack set. This provides our iterative algorithm for finding a non-dominated SSE strategy in zero-sum games. In order to find every smallest constraint set N_t, we need to enumerate all target pairs (t, t'). Therefore, the number of oracle calls of each iteration is $O(n^2)$, where oracles are used to solve variants of LP (11.1). There are at most n iterations, thus the overall runtime is $O(n^3)$ oracle calls.

Theorem 25. *Algorithm 12 correctly solves the non-dominated SSE in $O(n^3)$ oracle calls.*

11.5 General-sum Games

In this section, we discuss the refinement of SSEs in general-sum games. The method is similar to the zero-sum case, but one of the crucial difficulties is that there is no longer a direct relation between the defender and attacker utilities. Several useful properties of zero-sum games do not hold either. For example, in the general setting the intersection of two attack sets may not be an attack set, leading to non-uniqueness of the minimal attack set. This implies a significant growth of time complexity.

11.5.1 Non-Uniqueness of Minimal Attack Set

In Section 11.4, Theorem 21 tells us that in zero-sum games the intersection of two attack sets is still an attack set. The following example shows that it is not necessarily true in general-sum games.

Example 3. *Consider a game with one resource $R = \{r_1\}$, five targets $T = \{t_1, t_2, t_3, t_4, t_5\}$, and three schedules $S_1 = \{s_1, s_2, s_3\}$:*

$$s_1 = \{t_1, t_2\}, s_2 = \{t_3, t_4\}, s_3 = \{t_3, t_4, t_5\}$$

We have the following payoffs:

$$t_1 : U_d^c(t_1) = 10, U_d^u(t_1) = -10, U_a^u(t_1) = 10, U_a^c(t_1) = -10$$

$$t_2 : U_d^c(t_2) = 0,\ \ U_d^u(t_2) = -5,\ \ U_a^u(t_2) = 5,\ \ U_a^c(t_2) = -5$$

$$t_3 : U_d^c(t_3) = 6,\ \ U_d^u(t_3) = -4,\ \ U_a^u(t_3) = 3,\ \ U_a^c(t_3) = -7$$

$$t_4 : U_d^c(t_4) = 3,\ \ U_d^u(t_4) = -2,\ \ U_a^u(t_4) = 4,\ \ U_a^c(t_4) = -8.5$$

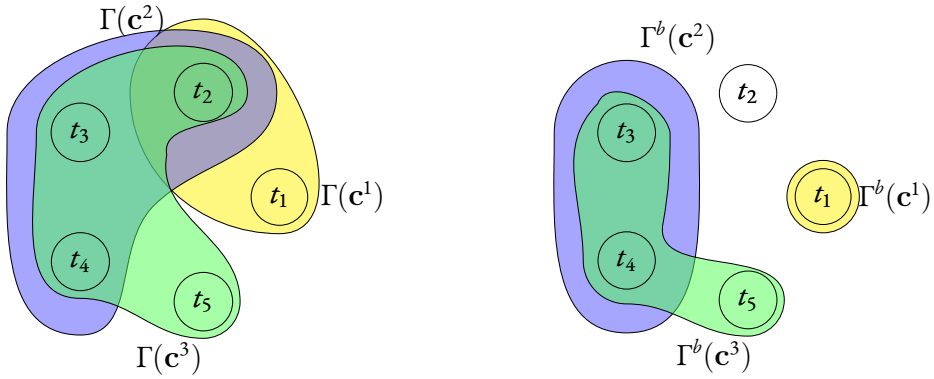
$$t_5 : U_d^c(t_5) = 4,\ \ U_d^u(t_5) = -1,\ \ U_a^u(t_5) = 0,\ \ U_a^c(t_5) = -5$$

Since the schedule s_2 is completely contained in the schedule s_3, the intuition tells us choosing s_3 will always be better than choosing s_2. However, this is wrong in this case. In order to show that, we list some SSE solutions with unsorted attacker utility $\mathbf{f}$*, unsorted defender utility* $\mathbf{d}$*, and defender utility* $\mathbf{v}$ *sorted in attacking order:*

$$\mathbf{x}^1 = \langle 0.5, 0.1, 0.4 \rangle, \qquad \mathbf{f}^1 = \langle 0, 0, -2, -2.25, -2 \rangle$$

$$\mathbf{d}^1 = \langle 0, -2.5, 1, 0.5, 1 \rangle, \qquad \mathbf{v}^1 = \langle 0, -2.5, 1, 1, 0.5 \rangle$$

$$\mathbf{x}^2 = \langle 0.6, 0, 0.4 \rangle, \qquad \mathbf{f}^2 = \langle -2, -1, -1, -1, -2 \rangle$$



(a) the set of attack sets Ψ

(b) the set of best attack sets Ψ^b

Figure 11.1: The attack sets in Example 3

$$\mathbf{d}^2 = \langle 2, -2, 0, 0, 1 \rangle, \qquad \mathbf{v}^2 = \langle 0, 0, -2, 2, 1 \rangle$$

$$\mathbf{x}^3 = \langle 0.6, 0.2, 0.2 \rangle, \qquad \mathbf{f}^3 = \langle -2, -1, -1, -1, -1 \rangle$$

$$\mathbf{d}^3 = \langle 2, -2, 0, 0, 0 \rangle, \qquad \mathbf{v}^3 = \langle 0, 0, 0, -2, 2 \rangle$$

$$\Gamma(\mathbf{c}^1) = \{t_1, t_2\}, \Gamma(\mathbf{c}^2) = \{t_2, t_3, t_4\}, \Gamma(\mathbf{c}^3) = \{t_2, t_3, t_4, t_5\}$$

It can be verified that these are all the possible attack sets. We have that $\mathbf{v}^3$ *dominates* $\mathbf{v}^2$ *and* $\mathbf{v}^1$*, which implies partially using inefficient schedule* s_2 *will result in better performance. Figure 11.1(a) illustrates all the attack sets in Example 3, which shows that the minimal attack set is not unique in general-sum games.*

11.5.2 Best Attack Set

We introduce the notion of **best attack set**. Similar to Section 11.4, we iteratively fix the coverage on the minimal best attack set: those targets the attacker will actually attack, up to breaking ties.

Definition 24. *Given an SSE coverage vector* $\mathbf{c}$*, the* ***Best Attack Set*** $\Gamma^b(\mathbf{c})$ *is the set of targets in the*

best response of the attacker which also achieves the highest defender utility.

In Example 3, as shown in Figure 11.1(b), the best attack sets are respectively $\Gamma^b(\mathbf{c}^1) = \{t_1\}, \Gamma^b(\mathbf{c}^2) = \{t_3, t_4\}, \Gamma^b(\mathbf{c}^3) = \{t_3, t_4, t_5\}$.

Definition 25. *Let* $\Psi^b = \{T' \subseteq T \mid \exists\, SSE\, \langle \mathbf{c}, \mathbf{a} \rangle : T' = \Gamma^b(\mathbf{c})\}$ *be the set of all possible best attack sets of SSEs.*

Theorem 26 (Intersection Property in General-sum Games)**.** *For any two attack sets* $\Gamma(\mathbf{c}), \Gamma(\mathbf{c}') \in \Psi$ *(Definition 20), if* $\Gamma^b(\mathbf{c}) \cap \Gamma^b(\mathbf{c}') \neq \emptyset$*, we have* $\Gamma(\mathbf{c}) \cap \Gamma(\mathbf{c}') \in \Psi, \Gamma^b(\mathbf{c}) \cap \Gamma^b(\mathbf{c}') \in \Psi^b$*.*

Proof. Given two sets $\Gamma(\mathbf{c}), \Gamma(\mathbf{c}') \in \Psi$, their corresponding SSEs $\langle \mathbf{c}, \mathbf{a} \rangle$ and $\langle \mathbf{c}', \mathbf{a}' \rangle$ with $\Gamma^b(c) \cap \Gamma^b(c') \neq \emptyset$, we follow a similar proof idea as in Theorem 21. Consider another strategy $\mathbf{c}^* = \alpha\mathbf{c} + (1-\alpha)\mathbf{c}'$ with $\alpha \in (0,1)$. $\mathbf{c}^*$ is a feasible coverage vector with strategy $\mathbf{x}^* = \alpha\mathbf{x} + (1-\alpha)\mathbf{x}'$. Moreover, they share some common targets in their best attack sets and thus the same highest attacker's utilities v_a and the highest defender's utility v_d. It is easy to verify that $\Gamma(\mathbf{c}^*) = \Gamma(\mathbf{c}) \cap \Gamma(\mathbf{c}'), \Gamma^b(\mathbf{c}^*) = \Gamma^b(\mathbf{c}) \cap \Gamma^b(\mathbf{c}')$ as follows:

$$U_a(\mathbf{c}, t) \begin{cases} = v_a & \text{if } t \in \Gamma(\mathbf{c}) \\ < v_a & \text{otherwise} \end{cases} \qquad U_a(\mathbf{c}', t) \begin{cases} = v_a & \text{if } t \in \Gamma(\mathbf{c}') \\ < v_a & \text{otherwise} \end{cases}$$

$$U_a(\mathbf{c}^*, t) = \alpha U_a(\mathbf{c}, t) + (1-\alpha) U_a(\mathbf{c}', t) \begin{cases} = v_a & \text{if } t \in \Gamma(\mathbf{c}) \cap \Gamma(\mathbf{c}') \\ < v_a & \text{otherwise} \end{cases} \tag{11.2}$$

$$U_d(\mathbf{c}, t) \begin{cases} = v_d & \text{if } t \in \Gamma^b(\mathbf{c}) \\ < v_d & \text{if } t \in \Gamma(\mathbf{c}) \backslash \Gamma^b(\mathbf{c}) \end{cases} \qquad U_d(\mathbf{c}', t) \begin{cases} = v_d & \text{if } t \in \Gamma^b(\mathbf{c}') \\ < v_d & \text{if } t \in \Gamma(\mathbf{c}') \backslash \Gamma^b(\mathbf{c}') \end{cases}$$

$$U_d(\mathbf{c}^*, t) = \alpha U_d(\mathbf{c}, t) + (1-\alpha) U_d(\mathbf{c}', t)$$

$$\Rightarrow U_d(\mathbf{c}^*, t) \begin{cases} = v_d & \text{if } t \in \Gamma^b(\mathbf{c}) \cap \Gamma^b(\mathbf{c}') \\ < v_d & \text{if } t \in (\Gamma(\mathbf{c}) \cap \Gamma(\mathbf{c}')) \backslash (\Gamma^b(\mathbf{c}) \cap \Gamma^b(\mathbf{c}')) \end{cases} \tag{11.3}$$

Equation (11.2) guarantees the attack set of strategy $\mathbf{c}^*$ is $\Gamma(\mathbf{c}) \cap \Gamma(\mathbf{c}')$. Equation (11.3) guarantees that among the attack set $\Gamma(\mathbf{c}) \cap \Gamma(\mathbf{c}')$, strategy $\mathbf{c}$ achieves the highest defender's utility on target t if and only if the target $t \in \Gamma^b(\mathbf{c}) \cap \Gamma^b(\mathbf{c}')$ which is non-empty. Thus, the best attack set of strategy $\mathbf{c}^*$ is $\Gamma^b(\mathbf{c}^*) = \Gamma^b(\mathbf{c}) \cap \Gamma^b(\mathbf{c}')$. □

Theorem 26 implies that the intersection of attack sets is still an attack set if the intersection of their best attack sets is non-empty. But if the intersection of their best attack sets is empty, the combining strategy $\mathbf{c}^*$ is no longer an SSE, and thus the intersection of attack sets may not be an attack set. Based on Theorem 26, we can define the minimal best attack set:

Definition 26. *A **Minimal Best Attack Set** is a best attack set $M \in \Psi^b$ such that any proper subset $V \subset M$ is not an element of Ψ^b, that is $V \notin \Psi^b$ for all $V \subset M \cap V\}$.*

Proposition 12. *Given any SSE strategy $\mathbf{c}$, its attack set $\Gamma(\mathbf{c})$ must contain one of the minimal best attack sets.*

11.5.3 A Recursive Algorithm

Based on the above theorems and paralleling Algorithm 12, Algorithm 14 iterates through all minimal best attack sets M and finds the non-dominated SSE in each restricted instances $g^{\mathbf{c},T' \cup M}$. After enumerating all the possible solutions, it returns the best one. The following results guarantee correctness of Algorithm 14.

Algorithm 14: ***RefinedSSE***(g)

```
Function RefinedRestrictedSSE(g, c, T')
    Parameter: restricted SPARS instance g^{c,T'}, cList ← []
    for each minimal best attack set M of g^{c,T'} do
        c* ← an SSE of instance g^{c,T'∪M}
        c' ← RefinedRestrictedSSE(g, c*, T' ∪ M)
        add c' into cList
    return non-dominated coverage vector among cList
return RefinedRestrictedSSE(g, c = 0, T' = ∅)
```

Proposition 13. *Assume M is a minimal restricted best attack set of $g^{\mathbf{c},T'}$ and $\mathbf{c}^*$ is an SSE strategy of $g^{\mathbf{c},T'}$ containing M in the attack set. Then, strategy $\mathbf{c}'$ is an SSE of $g^{\mathbf{c},T'}$ containing M if and only if $\mathbf{c}'$ is a solution of $g^{\mathbf{c}^*,T'\cup M}$.*

Theorem 27. *The output of Algorithm 14 is a non-dominated SSE.*

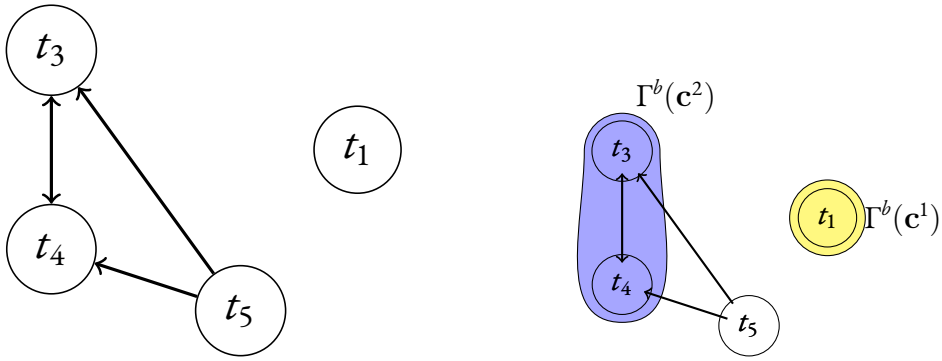
Proofs are similar to those of Proposition 8 and Theorem 23.

11.5.4 Computing Minimal Best Attack Sets

Similar to Section 11.4, we next propose an efficient method to find all the minimal best attack sets. Following the notations in Section 11.4.3, it can be shown that in general-sum games, solving the modified LPs (11.1) with respect to target t will yield the smallest tight constraint set N_t. The following proposition gives an alternative expression of N_t (the proof is similar to that of Theorem 11):

Proposition 14. $N_t = \bigcap_{T' \in \Psi^b : t \in T'} T'$

The set N_t provides the information between targets: if target t is included in the best attack set T', then all the targets in N_t must be included in the best attack set T' too. We can then focus on those targets which could be in the best attack set.



(a) the corresponding graph **(b)** Minimal best attack sets

Figure 11.2: The minimal best attack sets in the Example 3

Definition 27. *Let Q be the set of targets which achieve the best defender utility in some SSE strategies.*

The set Q is equivalent to the set of targets t for which LP (11.1) provides the highest defender utility and that can be derived while solving the n linear programs. We construct a directed graph $G = (V, E)$ to represent the relations between these targets. Let $V = Q$ be the set of all targets which could achieve the highest defender's utility. Let $E = \bigcup_{t \in Q} \{(t, s) | s \in N_t, s \in Q, s \neq t\}$ where (t, s) is the directed edge from t to s.

Example 4 (Continued from Example 3)**.** *With the help of Figure 11.1(b), we can visualize the sets $\{N_t | t \in Q\}$ ($Q = \{t_1, t_3, t_4, t_5\}$):*

$$N_{t_1} = \{t_1, t_2\}, N_{t_3} = \{t_3, t_4\}, N_{t_4} = \{t_3, t_4\}, N_{t_5} = \{t_3, t_4, t_5\}.$$

We can draw a corresponding graph (Figure 11.2(a)) according to these sets. Figure 11.2(b) depicts all of the minimal best attack sets. Notice that the definition of edges implies the inclusion relationship: $e = (t, s) \in E$ if and only if $t \in Q$, and any attack set including target t must also include target s.

Proposition 15. *Directed relations are* **transitive** *in graph $G = (V, E)$, i.e., if (t, u) and $(u, v) \in E$, then $(t, v) \in E$.*

The transitive rule has an intuitive meaning: if a best attack set is such that if t is included, so must u; and if u is included, so must v; then if it includes t, it must also include v.

Lemma 2. *M is a minimal best attack set if and only if M is a maximal clique without outgoing edge directed from M to any other target in $Q \backslash M$.*

Proof. ($\Rightarrow$) $\forall t \in M$, by Theorem 14, $N_t = \bigcap_{T' \in \Psi^b : t \in T'} T'$. Notice that the minimal best attack set M satisfies $M \in \Psi^b, t \in M$, thus $N_t \subseteq M$. Moreover, by Theorem 26, N_t is the intersection of best attack sets, which implies that N_t is a best attack set. But we have $N_t \subseteq M$ and M is a minimal best attack set. By the definition of the minimal best attack set, the only possibility is $N_t = M \, \forall \, t \in M$, which implies that M is a maximal clique without outgoing edges. ($\Leftarrow$) Suppose M is a maximal clique without any outgoing edge. Then $N_t = M \, \forall \, t \in M$. Since N_t is the intersection of best attack sets, $M = N_t$ is also a best attack set. Moreover, if a best attack set V includes any vertex $t \in M$, V must include $N_t = M$ (since $N_t = \bigcap_{T' \in \Psi^b : t \in T'} T'$, V satisfies $V \in \Psi^b : t \in V$). Next, we derive a contradiction. Suppose there is a proper subset $V \subset M$ which is also a best attack set. Then, there exists $t \in V \cap M$. By the above argument, we have $M = N_t \subseteq V$, which contradicts that $V \subset M$. We conclude that M is a minimal best attack set. □

Although the maximal clique problem is generically NP-hard, fortunately, the transitive law in Proposition 15 reduces the maximal clique problem to a variant of the tournament problem $G = (V, E)$ with time complexity $O(|V| + |E|) = O(n^2)$. In Algorithm 15, we leverage the transitive law to propose a random walk method that successfully discovers all the minimal best attack sets in $O(n^2)$.

Theorem 28. *Each subproblem in Algorithm 14 correctly returns the non-dominated solution in $O(n^3)$ oracle calls.*

Both the worst-case runtime of Algorithm 15 and the computation of all the sets $\{N_t | t \in Q\}$ are $O(n^2)$ oracle calls. Therefore, the worst-case runtime of solving each subproblem in the recursive al-

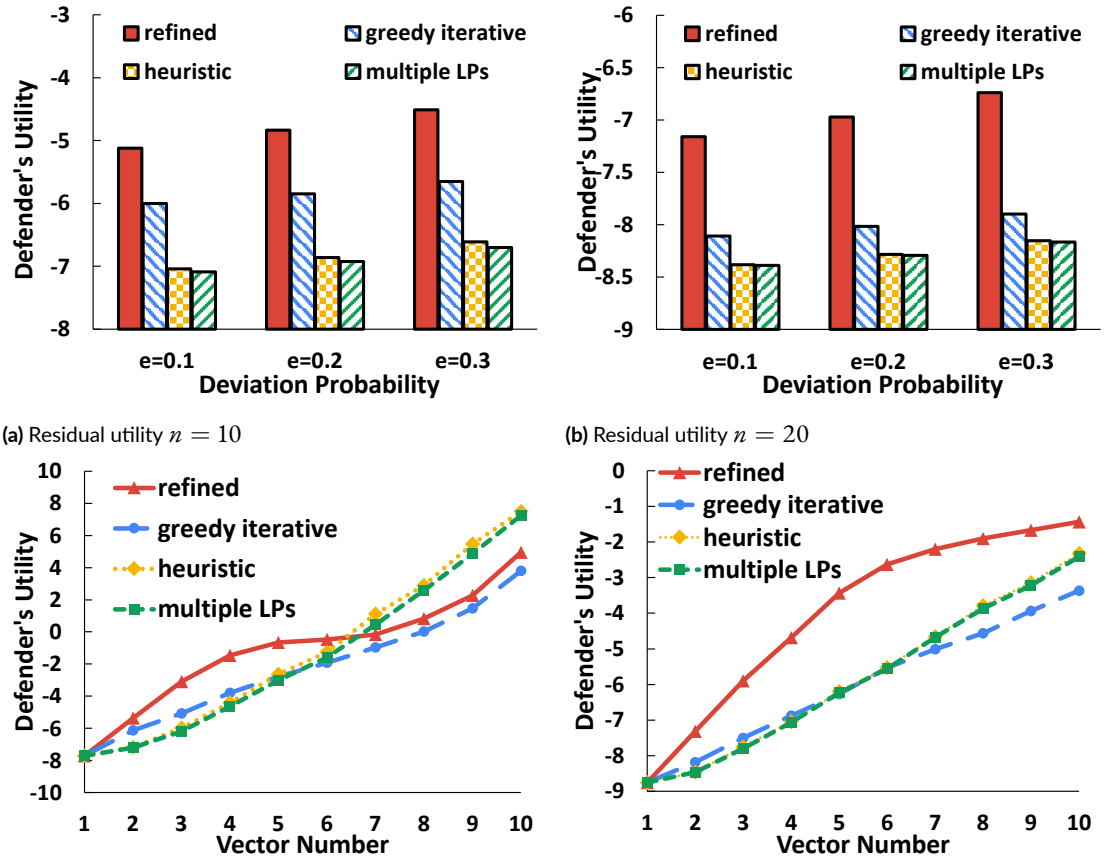
Algorithm 15: Find All the Minimal Best Attack Sets

```
Transitive graph G = (V, E), Mlist ← [], V' ← V, E' ← E
while V' ≠ ∅ do
    Start random walk in G' = (V', E') and record all the nodes we walked through
      until we encounter a duplicate node or we cannot move anymore. Let v be the
      last node.
    N_v ← N(v) ∪ {v} where N(v) is the neighborhood of v
    if N_v is a maximal clique without outgoing edges then
        add N_v into Mlist
    V' ← nodes in V' that have not been passed by
    E' ← edges in E with both endpoints ∈ V'
return Mlist
```

gorithm is still $O(n^3)$ oracle calls, same as the zero-sum cases. The number of subproblems depends on the number of minimal best attack sets. In Example 3, there are two minimal best attack sets: $\{t_1\}$ and $\{t_3, t_4\}$, so we need to compute the non-dominated solutions for both cases and choose the best one. The overall runtime depends on the number of subproblems that need to be solved. Fortunately, while iteratively solving the subproblems, rule (R3) enables us to foresee the defender's utilities on the first few targets, thus prune out a large number of subproblems, which reduces the overall runtime significantly relative to the worst-case (reduce from exponential to polynomial many oracle calls in practice).

11.6 Experimental Results

We run experiments to evaluate the solution quality and scalability of the refined SSE on SPARS. All LPs are solved by CPLEX (version 12.7.1) on a machine with an Intel core i5-7200U CPU and 11.6GB memory. Our experiments use 100 sampled game instances with 2 defender resources, varying the number of targets, and randomly generated payoffs. In zero-sum cases, payoffs $U_a^u(t) = -U_d^u(t)$, $U_d^c(t) = -U_a^c(t)$ are uniformly distributed in the set $\{0, 1, \ldots, 10\}$. In general-sum cases,



(a) Residual utility $n = 10$

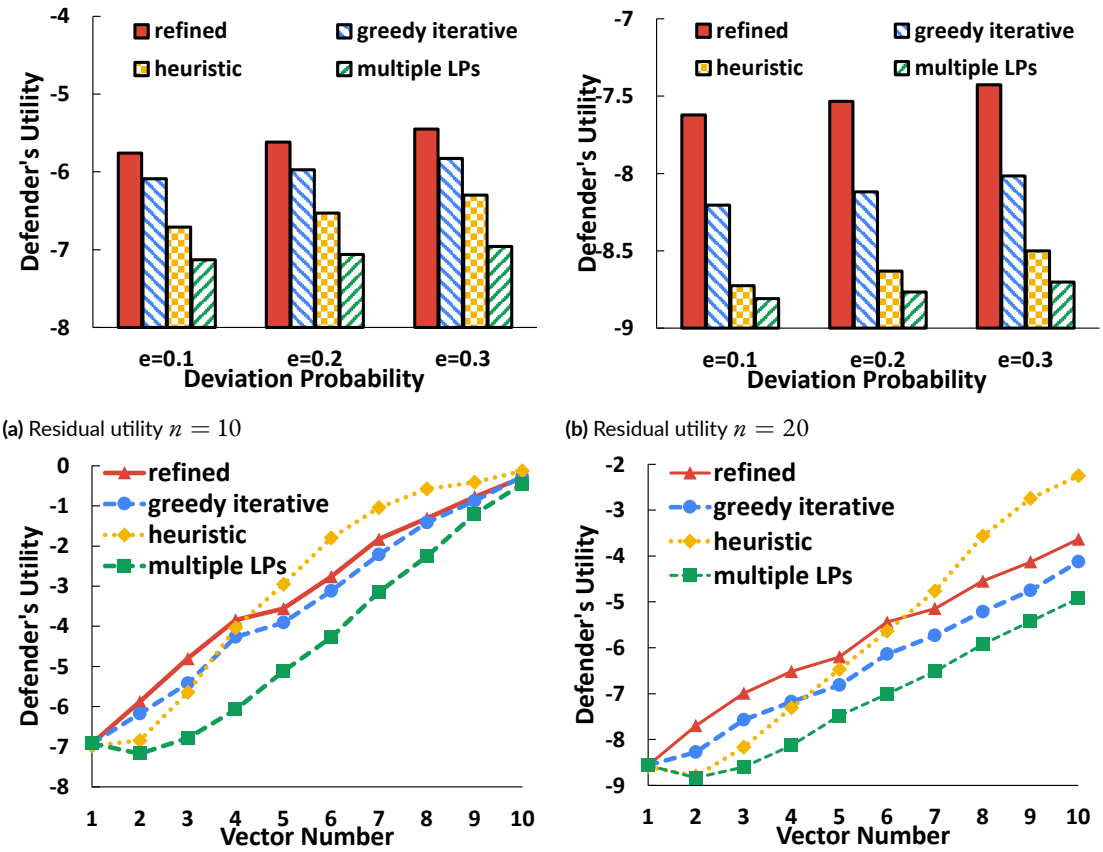
(b) Residual utility $n = 20$

(c) Utility vectors $n = 10$

(d) Utility vectors $n = 20$

Figure 11.3: Performance of equilibrium refinement in zero-sum Stackelberg security games.

we are motivated by ARMOR[252] and adopt the following payoff setting: $U_a^u(t) = -U_d^u(t)$ uniformly distributed in the set $\{0, 1, \ldots, 10\}$ (completely opposite on successful attack), $U_d^c(t) = 0$ (zero reward for successful protect), and $U_a^c(t)$ uniformly distributed in $\{0, 1, \ldots, \lfloor U_a^u(t)/2 \rfloor\}$. Each instance also encompasses $O(n)$ randomly generated scheduling constraints with each schedule covering 2 to 5 targets depending on the number of targets. We employ CPLEX as our oracle to obtain exact solutions to linear programs. We compare the solution quality of our refined SSE to the



(a) Residual utility $n = 10$

(b) Residual utility $n = 20$

(c) Utility vectors $n = 10$

(d) Utility vectors $n = 20$

Figure 11.4: Performance of equilibrium refinement in general-sum Stackelberg security games.

SSEs given by the **multiple LPs** method[71], **heuristic** method[93], and **greedy iterative** method[15]†.

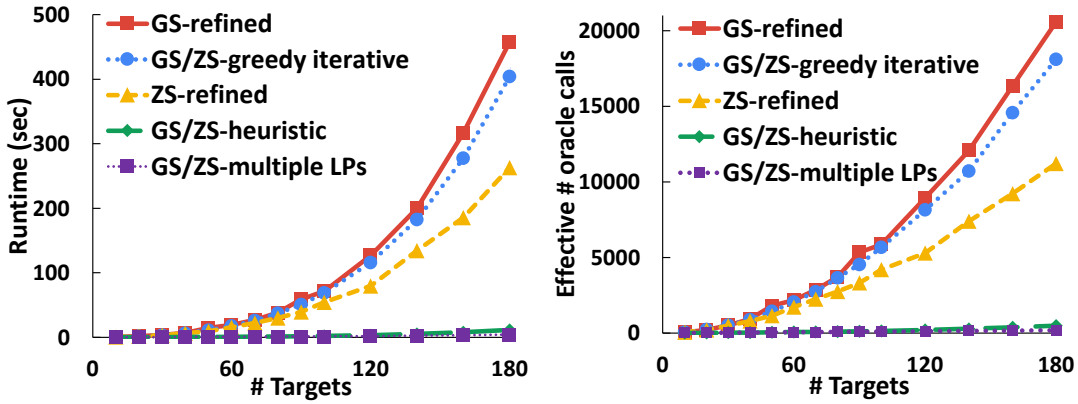
Since the defender utilities on the first preferable target are identical for all SSEs, we display the *residual* expected utility for the remaining targets. Suppose the attacker deviates from his target to the secondary target with probability e. Further assume that the attacker does not attack the first preferable target, then the attacker will attack the second preferable target with probability $1 - e$,

†The **heuristic** method starts from an arbitrary SSE and goes through all of the pure strategies. If there is a strictly better pure strategy than the pure strategy in the current mixed strategy, then move the weight to the better one. The **greedy iterative** method adopts the idea of the iterative algorithm[15] but without finding minimal attack sets. It iteratively fixes the coverage of an arbitrary target in the attack set (best attack set).

third preferable target with $e(1-e)$ and so on. Given the utility vector $\mathbf{v}$ sorted by the attacking order, the residual value is expressible as $\sum_{2\leq i\leq n}(1-e)e^{i-2}v_i$.

Figures 11.3(a), 11.3(b), 11.4(a), 11.4(b) illustrate the residual expected utilities in zero and general-sum games with $n = 10$ and 20, respectively. Without spending additional resources, our refined solution outperforms the other SSE solutions, improving the defender utility by $10-40\%$. Figures 11.3(c), 11.3(d), 11.4(c), 11.4(d) depict the defender utilities in attacking order. The figures show that *(i)* the refined SSE and other SSEs provide the same defender utility on the first preferable target; *(ii)* While the heuristic and multiple LPs methods are a lot faster than ours (Figures 11.5(a), 11.5(b)), they perform significantly worse since they do not refine the solution; *(iii)* The refined SSE gives a much higher defender utility on the following few targets (second and third preferable) by sacrificing those less preferable targets, which are even more unlikely to be attacked than the first few targets.

Figure 11.5(a) (resp. 11.5(b)) compares the runtime (resp. number of oracle calls) of our algorithm relative to other algorithms in zero-sum (ZS) and general-sum (GS) cases. The results show *(i)* the runtime of both zero and general-sum cases is of the same order as the runtime of the greedy iterative algorithm, which requires $O(n^2)$ oracle calls. Thus, the empirical number of oracle calls is significantly lower than our worst-case estimate of $O(n^3)$. This is due to the fact that in random settings, the cardinality of Q (Definition 27) is small (usually under 4), resulting in a small number of enumerations of $N_t, t \in Q$; *(ii)* Our algorithm for zero-sum games is almost two times faster than the greedy iterative algorithm because fixing the minimal attack set can significantly reduce the number of iterations, which speeds up our algorithm and also boosts solution quality; *(iii)* Figure 11.5(a) also shows that the runtime of our optimal algorithm is close to the runtime of the greedy iterative one. Contrary to the greedy iterative approach, our algorithm guarantees optimality and provides a significant improvement in defender utility and robustness, see Figures 11.3(c), 11.3(d), 11.4(c), and 11.4(d) at low computational cost, which provides a more robust solution with further spending



(a) Runtime Comparison **(b)** Effective # oracle calls

Figure 11.5: Computation cost of equilibrium refinement with varying problem sizes.

only little more runtime.

11.7 Conclusion

In summary, we show that the refinement is critical in Security Problems with ARbitrary Schedule (SPARS) domain and existing algorithms may lead to suboptimal performance. We provide theoretical analyses by defining minimal attack set and dominance relationship to design algorithms that computes the non-dominated Strong Stackelberg Equilibrium (SSE) with scalable computation cost in both zero-sum and general-sum games.

12

Conclusion

This thesis presents a set of algorithmic, methodological, and theoretical contributions in the data-to-deployment pipeline to integrate optimization and machine learning problems in public health and conservation. On the technical level, the thesis presents techniques for integrating knowledge from optimization and different decision-making processes to strengthen machine learning performance, including supervised learning, online learning, and multi-agent systems in the face of uncertainty and with limited data.

Part I discusses the integration of optimization in supervised learning to train machine learning models in the presence of optimization. Chapter 2 and Chapter 3 study the integration of sequential problems as a differentiable layer into machine learning to enable the first decision-focused learning in sequential problems with approximate solutions to reduce computation costs. These two works set the foundation for applying decision-focused learning to public health and the deployment to the maternal and child health program as shown in Chapter 4. Lastly, Chapter 5 and Chapter 6 study the integration of non-sequential optimization into machine learning by proposing sampling and surrogate algorithms to reduce computation costs.

Part II focuses on using optimization to design online learning algorithms to collect data and strengthen theoretical guarantees. Chapter 7 studies using optimization to handle additive and independent feedback in multi-armed bandits with continuous action space. Chapter 8 studies using optimization to handle additive but weekly dependent feedback in restless multi-armed bandits. Chapter 9 studies using optimization to leverage non-additive and dependent feedback in online combinatorial optimization problems. All these three works use optimization to design online algorithms with improved theoretical guarantees and empirical results.

Part III focuses on designing scalable and approximate solutions to solve optimization in multi-agent systems using Stackelberg games. Chapter 10 extends the idea from decision-focused learning to multi-agent systems and proposes the first gradient-based algorithm to find the best equilibrium of Stackelberg games with multiple followers. Chapter 11 proposes efficient algorithms to solve the equilibrium refinement problem in Stackelberg security games with arbitrary constraints. These works focus on complexity and the design of scalable algorithms in finding equilibria in multi-agent systems.

From a practical perspective, this thesis introduces how AI algorithms and theory can be applied to public health and conservation challenges. On the public health front, the thesis covers maternal health, tuberculosis, and epidemiology using the data-to-deployment pipeline. Specifically, the

maternal health application includes a real-world field study and deployment to a mobile health program where the proposed decision-focused learning algorithm is currently used by more than 100,000 people to improve engagement with health information. On the conservation front, the thesis covers optimizing patrol strategies in wildlife conservation, determining mechanisms to incentivize collaboration between multiple patrol teams, and interrupting illegal wildlife trade in a physical network. These examples demonstrate the use of machine learning, optimization, and multi-agent systems to design AI solutions in public health and conservation.

From the perspective of using AI to create social impact, the thesis emphasizes the importance of engaging with stakeholders and organizations that possess a deep understanding of societal challenges in order to consolidate optimization and decision-making processes to design suitable AI solutions. On one hand, these optimization formulations provide valuable domain knowledge to enhance machine learning approaches. The optimization problems formulated in collaboration with domain experts reveal the constraints and knowledge pertinent to the societal challenges, which are critical for effectively characterizing societal issues and quantifying uncertainty, especially in scenarios where data is limited. On the other hand, involving stakeholders in the design of AI solutions and the data-to-deployment pipeline helps ensure the AI solutions meet the need of the stakeholders to ultimately convert algorithmic contributions to deployment. By incorporating stakeholders' input, the designed AI solutions can better reflect the constraints and requirements faced in societal challenges, making them more suitable for deployment. For instance, the collaborative work with ARMMAN in Chapter 2 and Chapter 3 to define maternal health decision-making processes was instrumental in the successful deployment of the solutions, largely benefiting from the involvement of the organization and domain experts. In summary, the thesis establishes the algorithmic and theoretical foundation for integrating optimization obtained from stakeholders into machine learning to effectively leverage knowledge from decision-making processes in a computationally efficient manner.

Future vision

A significant amount of work remains to be done in developing AI solutions for public health and conservation. Although we have seen the empirical and theoretical success of decision-focused learning in both simulation and real-world deployment, the computational cost of decision-focused learning algorithms has become a critical concern. The involvement of optimization problems in the learning pipeline makes the training process much more expensive and non-smooth, hindering gradient-based approaches from working efficiently. Additionally, our current understanding of the differentiability of different optimization and decision-making processes is also limited to continuous mathematical optimization and sequential decision problems modeled as Markov decision processes. Many societal challenges involve decision-making processes that are more complex than our algorithmic frontier, which limits the applicability of decision-focused learning. Therefore, the scalability and the applicability to other decision-making processes remain challenging for integrating optimization into machine learning algorithms.

Furthermore, our current understanding of the knowledge embedded in optimization and decision-making processes remains limited. This limitation significantly impacts the explainability and robustness of decision-focused learning, which becomes increasingly important as the methodology of decision-focused learning matures and gets adopted more frequently. It is unclear whether integrating optimization into machine learning is the ultimate solution for incorporating knowledge from optimization and decision-making processes. Historically, machine learning algorithms use regularization terms to indirectly incorporate domain knowledge and insights into machine learning objectives. In contrast, decision-focused learning algorithms integrate optimization and decision-making processes into the learning pipeline to directly incorporate domain knowledge to define machine learning objectives, but in a cost of computation cost and non-smoothness that can impact learning performance, explainability, and robustness. Further investigation into the connec-

tion between regularization and the integration of optimization is needed, and consideration of the truthworhiness is also important in integrating optimization and machine learning.

Lastly, on a broader level, the journey of AI for social impact has only just begun. This thesis lays the foundation for integrating optimization, machine learning, multi-agent systems, and stakeholders' involvement to design AI solutions for deployment. However, there are much more AI techniques and well-established areas that need to be studied and integrated into the research of AI for social impact. As interdisciplinary research continues to flourish, we gain a deeper understanding of how different AI techniques and domain knowledge intersect and interact. With more research on AI for social impact and the application of AI in various fields and societal challenges, AI and computer science gradually define their unique position and responsibility in working with domain experts, non-governmental organizations, and governments. These efforts will serve as the nourishment that fosters algorithmic contributions to social impact and propels AI to thrive in our societies.

References

[1] Abbasi, Y., Kar, D., Sintov, N., Tambe, M., Ben-Asher, N., Morrison, D., & Gonzalez, C. (2016). Know your adversary: Insights for a better adversarial behavioral model. In *CogSci*.

[2] Abbasi, Y. D., Short, M., Sinha, A., Sintov, N., Zhang, C., & Tambe, M. (2015). Human adversaries in opportunistic crime security games: evaluating competing bounded rationality models. In *Proc. of Advances in Cognitive Systems*.

[3] Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., & Kolter, J. Z. (2019a). Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*.

[4] Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., & Kolter, J. Z. (2019b). Differentiable convex optimization layers. In *NeurIPS-19* Vancouver.

[5] Agrawal, P., Balkanski, E., Gkatzelis, V., Ou, T., & Tan, X. (2022). Learning-augmented mechanism design: Leveraging predictions for facility location. In *Proceedings of the 23rd ACM Conference on Economics and Computation* (pp. 497–528).

[6] Akbarzadeh, N. & Mahajan, A. (2019). Restless bandits with controlled restarts: Indexability and computation of whittle index. In *2019 IEEE 58th Conference on Decision and Control (CDC)* (pp. 7294–7300).: IEEE.

[7] Alimohamadi, Y., Khodamoradi, F., Khoramdad, M., Shahbaz, M., Esmaeilzadeh, F., et al. (2019). Human development index, maternal mortality rate and under 5 years mortality rate in west and south asian countries, 1980–2010: an ecological study. *East Mediterr Health J*, 25(3), 189–196.

[8] Allen, L. H. (2000). Anemia and iron deficiency: effects on pregnancy outcome. *The American journal of clinical nutrition*, 71(5), 1280S–1284S.

[9] Alman, J. & Williams, V. V. (2021). A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)* (pp. 522–539).: SIAM.

[10] Amin, M. & Ali, A. (2018). Performance evaluation of supervised machine learning classifiers for predicting healthcare operational decisions. *Wavy AI Research Foundation: Lahore, Pakistan*, 90.

[11] Amos, B. & Kolter, J. Z. (2017). OptNet: Differentiable optimization as a layer in neural networks. In *ICML-17* Sydney.

[12] Amos, B., Rodriguez, I. D. J., Sacks, J., Boots, B., & Kolter, J. Z. (2018). Differentiable mpc for end-to-end planning and control. *arXiv preprint arXiv:1810.13400*.

[13] Amos, B., Xu, L., & Kolter, J. Z. (2017). Input convex neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (pp. 146–155).: JMLR. org.

[14] Amsallem, D., Zahr, M., Choi, Y., & Farhat, C. (2015). Design optimization using hyper-reduced-order models. *Structural and Multidisciplinary Optimization*, 51(4), 919–940.

[15] An, B., Tambe, M., Ordonez, F., Shieh, E., & Kiekintveld, C. (2011). Refinement of strong stackelberg equilibria in security games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25 (pp. 587–593).

[16] Anderson, B. & Jooste, J. (2014). *Wildlife poaching: Africa's surging trafficking threat*. Africa Center for Strategic Studies.

[17] Andrew, L., Barman, S., Ligett, K., Lin, M., Meyerson, A., Roytman, A., & Wierman, A. (2013). A tale of two metrics: Simultaneous bounds on competitiveness and regret. In *COLT* (pp. 741–763).

[18] Angrist, J. D. & Pischke, J.-S. (2008). *Mostly harmless econometrics*. Princeton university press.

[19] Antoniadis, A., Coester, C., Elias, M., Polak, A., & Simon, B. (2020). Online metric algorithms with untrusted predictions. In *ICML* (pp. 345–355).

[20] ARMMAN (2022). ARMMAN helping mothers and children. https://armman.org/. Accessed: 2022-05-19.

[21] Arsovska, J. & Kostakos, P. A. (2008). Illicit arms trafficking and the limits of rational choice theory: the case of the balkans. *Trends in Organized Crime*, 11(4), 352–378.

[22] Atta, C. A., Fiest, K. M., Frolkis, A. D., Jette, N., Pringsheim, T., St Germaine-Smith, C., Rajapakse, T., Kaplan, G. G., & Metcalfe, A. (2016). Global birth prevalence of spina bifida by folic acid fortification status: a systematic review and meta-analysis. *American journal of public health*, 106(1), e24–e34.

[23] Audibert, J.-Y., Bubeck, S., & Lugosi, G. (2014). Regret in online combinatorial optimization. *Mathematics of Operations Research*, 39(1), 31–45.

[24] Auer, P. (2002). Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov), 397–422.

[25] Auer, P. & Ortner, R. (2006). Logarithmic online regret bounds for undiscounted reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 19.

[26] Aussel, D., Brotcorne, L., Lepaul, S., & von Niederhäusern, L. (2020). A trilevel model for best response in energy demand-side management. *EJOR*.

[27] Avrachenkov, K. E. & Borkar, V. S. (2022). Whittle index based Q-learning for restless bandits with average reward. *Automatica*, 139, 110186.

[28] Ayling, J. (2013). What sustains wildlife crime? rhino horn trading and the resilience of criminal networks. *Journal of International Wildlife Law & Policy*, 16(1), 57–80.

[29] Badiei, M., Li, N., & Wierman, A. (2015). Online convex optimization with ramp constraints. In *2015 54th IEEE Conference on Decision and Control (CDC)* (pp. 6730–6736).

[30] Baek, J. & Farias, V. F. (2020). TS-UCB: Improving on Thompson sampling with little to no additional computation. *arXiv preprint arXiv:2006.06372*.

[31] Bagheri, S. & Scaglione, A. (2015). The restless multi-armed bandit formulation of the cognitive compressive sensing problem. *IEEE Transactions on Signal Processing*, 63(5), 1183–1198.

[32] Bai, S., Kolter, J. Z., & Koltun, V. (2019). Deep equilibrium models. In *Advances in Neural Information Processing Systems* (pp. 688–699).

[33] Balghiti, O. E., Elmachtoub, A. N., Grigas, P., & Tewari, A. (2019). Generalization bounds in the predict-then-optimize framework. *arXiv preprint arXiv:1905.11488*.

[34] Balkanski, E., Gkatzelis, V., & Tan, X. (2022). Strategyproof scheduling with predictions. *arXiv preprint arXiv:2209.04058*.

[35] Balkanski, E., Rubinstein, A., & Singer, Y. (2017). The limitations of optimization from samples. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing* (pp. 1016–1027).

[36] Bandler, J. W., Biernacki, R. M., Chen, S. H., Grobelny, P. A., & Hemmers, R. H. (1994). Space mapping technique for electromagnetic optimization. *IEEE Transactions on Microwave Theory and Techniques*, 42(12), 2536–2544.

[37] Bartal, Y., Bollobás, B., & Mendel, M. (2006). Ramsey-type theorems for metric spaces with applications to online problems. *Journal of Computer and System Sciences*, 72(5), 890–921.

[38] Bartal, Y., Linial, N., Mendel, M., & Naor, A. (2003). On metric ramsey-type phenomena. In *STOC* (pp. 463–472).

[39] Basilico, N., Coniglio, S., & Gatti, N. (2017). Methods for finding leader–follower equilibria with multiple followers. *arXiv preprint arXiv:1707.02174*.

[40] Basilico, N., Coniglio, S., Gatti, N., & Marchesi, A. (2020). Bilevel programming methods for computing single-leader-multi-follower equilibria in normal-form and polymatrix games. *EJCO*.

[41] Benamou, J.-D., Carlier, G., Cuturi, M., Nenna, L., & Peyré, G. (2015). Iterative bregman projections for regularized transportation problems. *SIAM Journal on Scientific Computing*, 37(2), A1111–A1138.

[42] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 1798–1828.

[43] Bertsekas, D. P. (2014). *Constrained optimization and Lagrange multiplier methods*. Academic press.

[44] Bertsekas, D. P. & Tsitsiklis, J. N. (1996). *Neuro-dynamic Programming*. Belmont, MA: Athena.

[45] Bhat, N., Farias, V. F., Moallemi, C. C., & Sinha, D. (2020). Near-optimal ab testing. *Management Science*, 66(10), 4477–4495.

[46] Bian, A., Levy, K., Krause, A., & Buhmann, J. M. (2017). Continuous dr-submodular maximization: Structure and algorithms. In *Advances in Neural Information Processing Systems*.

[47] Bian, A. A., Mirzasoleiman, B., Buhmann, J. M., & Krause, A. (2016). Guaranteed non-convex optimization: Submodular maximization over continuous domains. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*.

[48] Biswas, A., Aggarwal, G., Varakantham, P., & Tambe, M. (2021). Learn to intervene: An adaptive learning policy for restless bandits in application to preventive healthcare. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*.

[49] Blum, A., Haghtalab, N., Hajiaghayi, M., & Seddighin, S. (2019). Computing stackelberg equilibria of large general-sum games. In *International Symposium on Algorithmic Game Theory* (pp. 168–182).: Springer.

[50] Borkar, V. S., Kasbekar, G. S., Pattathil, S., & Shetty, P. Y. (2017). Opportunistic scheduling as restless bandits. *IEEE Transactions on Control of Network Systems*, 5(4), 1952–1961.

[51] Branco, P., Torgo, L., & Ribeiro, R. P. (2016). A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys (CSUR)*, 49(2), 1–50.

[52] Breton, M., Alj, A., & Haurie, A. (1988). Sequential stackelberg equilibria in two-person games. *Journal of Optimization Theory and Applications*, 59(1), 71–97.

[53] Brochu, E., Cora, V. M., & De Freitas, N. (2010). A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*.

[54] Bubeck, S. & Cesa-Bianchi, N. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *arXiv preprint arXiv:1204.5721*.

[55] Bubeck, S., Cohen, M. B., Lee, J. R., & Lee, Y. T. (2021). Metrical task systems on trees via mirror descent and unfair gluing. *SIAM Journal on Computing*, 50(3), 909–923.

[56] Bubeck, S., Klartag, B., Lee, Y. T., Li, Y., & Sellke, M. (2020). Chasing nested convex bodies nearly optimally. In *SODA* (pp. 1496–1508).: SIAM.

[57] Bubeck, S., Lee, Y. T., Li, Y., & Sellke, M. (2019). Competitively chasing convex bodies. In *STOC* (pp. 861–868).

[58] Bucarey, V., Casorrán, C., Figueroa, Ó., Rosas, K., Navarrete, H., & Ordóñez, F. (2017). Building real stackelberg security games for border patrols. In *International Conference on Decision and Game Theory for Security* (pp. 193–212).: Springer.

[59] Calvete, H. I. & Galé, C. (2007). Linear bilevel multi-follower programming with independent followers. *Journal of Global Optimization*, 39(3), 409–417.

[60] Camacho, E. F. & Alba, C. B. (2013). *Model predictive control*. Springer science & business media.

[61] Campbell, C., Sands, S., Ferraro, C., Tsao, H.-Y. J., & Mavrommatis, A. (2020). From data to action: How marketers can leverage ai. *Business Horizons*, 63(2), 227–243.

[62] Campbell, O. M. & Graham, W. J. (2006). Strategies for reducing maternal mortality: getting on with what works. *The lancet*, 368(9543), 1284–1299.

[63] Challender, D. W. & MacMillan, D. C. (2014). Poaching is more than an enforcement problem. *Conservation Letters*, 7(5), 484–494.

[64] Chen, N., Agarwal, A., Wierman, A., Barman, S., & Andrew, L. L. (2015). Online convex optimization using predictions. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems* (pp. 191–204).

[65] Chen, N., Comden, J., Liu, Z., Gandhi, A., & Wierman, A. (2016). Using predictions in online optimization: Looking forward with an eye on the past. *ACM SIGMETRICS Performance Evaluation Review*, 44(1), 193–206.

[66] Chen, W., Wang, Y., & Yuan, Y. (2013). Combinatorial multi-armed bandit: General framework and applications. In *International Conference on Machine Learning (ICML)* (pp. 151–159).: PMLR.

[67] Collier, A.-r. Y. & Molina, R. L. (2019). Maternal mortality in the united states: updates on trends, causes, and solutions. *Neoreviews*, 20(10), e561–e574.

[68] Collins, L. M., Murphy, S. A., & Strecher, V. (2007). The multiphase optimization strategy (most) and the sequential multiple assignment randomized trial (smart): new methods for more potent ehealth interventions. *American journal of preventive medicine*, 32(5), S112–S118.

[69] Colson, B., Marcotte, P., & Savard, G. (2007). An overview of bilevel optimization. *Annals of operations research*, 153(1), 235–256.

[70] Coniglio, S., Gatti, N., & Marchesi, A. (2020). Computing a pessimistic stackelberg equilibrium with multiple followers: The mixed-pure case. *Algorithmica*, 82(5), 1189–1238.

[71] Conitzer, V. & Sandholm, T. (2006). Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce* (pp. 82–90).: ACM.

[72] Conn, A. R., Gould, N. I., & Toint, P. L. (2000). *Trust region methods*. SIAM.

[73] Contal, E., Perchet, V., & Vayatis, N. (2014). Gaussian process optimization with mutual information. In *International Conference on Machine Learning* (pp. 253–261).

[74] Cooney, R., Roe, D., Dublin, H., Phelps, J., Wilkie, D., Keane, A., Travers, H., Skinner, D., Challender, D. W., Allan, J. R., et al. (2017). From poachers to protectors: engaging local communities in solutions to illegal wildlife trade. *Conservation Letters*, 10(3), 367–374.

[75] Cooney, S., Wang, K., Bondi, E., Nguyen, T., Vayanos, P., Winetrobe, H., Cranford, E. A., Gonzalez, C., Lebiere, C., & Tambe, M. (2019). Learning to signal in the goldilocks zone: Improving adversary compliance in security games. In *ECMLPKDD-19* Würzburg.

[76] Cugurullo, F. (2020). Urban artificial intelligence: From automation to autonomy in the smart city. *Frontiers in Sustainable Cities*, 2, 38.

[77] Dafermos, S. (1988). Sensitivity analysis in variational inequalities. *Mathematics of Operations Research*, 13(3), 421–434.

[78] Dai, W., Gai, Y., Krishnamachari, B., & Zhao, Q. (2011). The non-bayesian restless multi-armed bandit: A case of near-logarithmic regret. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2940–2943).: IEEE.

[79] Del Valle, S. Y., Hyman, J. M., & Chitnis, N. (2013). Mathematical models of contact patterns between age groups for predicting the spread of infectious diseases. *Mathematical biosciences and engineering: MBE*, 10, 1475.

[80] Donti, P., Amos, B., & Kolter, J. Z. (2017). Task-based end-to-end model learning in stochastic optimization. In *NIPS-17* (pp. 5484–5494). Long Beach.

[81] Du, D., Lu, R., & Xu, D. (2012). A primal-dual approximation algorithm for the facility location problem with submodular penalties. *Algorithmica*, 63(1-2), 191–200.

[82] Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., & Abbeel, P. (2016). Rl ^{2}: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*.

[83] Duffy, R. (2014). Waging a war to save biodiversity: the rise of militarized conservation. *International Affairs*, 90(4), 819–834.

[84] Durkota, K., Lisỳ, V., Kiekintveld, C., Horák, K., Bošanskỳ, B., & Pevnỳ, T. (2017). Optimal strategies for detecting data exfiltration by internal and external attackers. In *International Conference on Decision and Game Theory for Security* (pp. 171–192).: Springer.

[85] Dwivedi, R., Murphy, S., & Shah, D. (2022). Counterfactual inference for sequential experimental design. *arXiv preprint arXiv:2202.06891*.

[86] Dwivedi, Y. K., Ismagilova, E., Hughes, D. L., Carlson, J., Filieri, R., Jacobson, J., Jain, V., Karjaluoto, H., Kefi, H., Krishen, A. S., et al. (2021). Setting the future of digital and social media marketing research: Perspectives and research propositions. *International Journal of Information Management*, 59, 102168.

[87] El Balghiti, O., Elmachtoub, A., Grigas, P., & Tewari, A. (2019). Generalization bounds in the predict-then-optimize framework. In *Advances in Neural Information Processing Systems*.

[88] Elkan, C. (2001). The foundations of cost-sensitive learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*.

[89] Elmachtoub, A., Liang, J. C. N., & McNellis, R. (2020). Decision trees for decision-making under the predict-then-optimize framework. In *International Conference on Machine Learning* (pp. 2858–2867).: PMLR.

[90] Elmachtoub, A. N. & Grigas, P. (2017). Smart "predict, then optimize". *arXiv preprint arXiv:1710.08005*.

[91] Elmachtoub, A. N. & Grigas, P. (2021). Smart "predict, then optimize". *Management Science*.

[92] Facchinei, F., Kanzow, C., & Sagratella, S. (2014). Solving quasi-variational inequalities via their KKT conditions. *Mathematical Programming*, 144(1-2), 369–412.

[93] Fang, F., Jiang, A. X., & Tambe, M. (2013). Optimal patrol strategy for protecting moving targets with multiple mobile resources. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems* (pp. 957–964).: International Foundation for Autonomous Agents and Multiagent Systems.

[94] Fang, F., Nguyen, T. H., Pickles, R., Lam, W. Y., Clements, G. R., An, B., Singh, A., Tambe, M., Lemieux, A., et al. (2016). Deploying paws: Field optimization of the protection assistant for wildlife security. In *AAAI*.

[95] Fang, F., Stone, P., & Tambe, M. (2015). When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *IJCAI-15* (pp. 2589–2595). Buenos Aires.

[96] Farrell, J. & Klemperer, P. (2007). Coordination and lock-in: Competition with switching costs and network effects. *Handbook of industrial organization*, 3, 1967–2072.

[97] Ferber, A., Wilder, B., Dilina, B., & Tambe, M. (2020). MIPaaL: Mixed integer program as a layer. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*.

[98] Ferrier, P. et al. (2009). *The economics of agricultural and wildlife smuggling*. Technical report, Springer.

[99] Filteau, M. R. (2012). Deterring defiance:'don't give a poacher a reason to poach'. *International Journal of Rural Criminology*.

[100] Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning* (pp. 1126–1135).: PMLR.

[101] Fischetti, M., Ljubic, I., Monaci, M., & Sinnl, M. (2016). *Interdiction games and monotonicity*. Technical report, Technical Report, DEI, University of Padova.

[102] Fischetti, M., Ljubić, I., Monaci, M., & Sinnl, M. (2019). Interdiction games and monotonicity, with application to knapsack problems. *INFORMS Journal on Computing*.

[103] Flaxman, A. D., Kalai, A. T., & McMahan, H. B. (2004). Online convex optimization in the bandit setting: gradient descent without a gradient. *arXiv preprint cs/0408007*.

[104] for Disease Control, C., (CDC, P., et al. (2004). Spina bifida and anencephaly before and after folic acid mandate–united states, 1995-1996 and 1999-2000. *MMWR. Morbidity and mortality weekly report*, 53(17), 362–365.

[105] Ford, B., Nguyen, T., Tambe, M., Sintov, N., & Delle Fave, F. (2015). Beware the soothsayer: From attack prediction accuracy to predictive reliability in security games. In *GameSec-15* (pp. 35–56).

[106] Forrester, A. I. & Keane, A. J. (2009). Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45(1-3), 50–79.

[107] Foster, D. & Rakhlin, A. (2020). Beyond UCB: Optimal and efficient contextual bandits with regression oracles. In *International Conference on Machine Learning (ICML)* (pp. 3199–3210).: PMLR.

[108] Frank, M. & Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2), 95–110.

[109] Friedman, J. & Linial, N. (1993). On convex body chasing. *Discrete & Computational Geometry*, 9(3), 293–321.

[110] Fu, J., Nazarathy, Y., Moka, S., & Taylor, P. G. (2019). Towards Q-learning the Whittle index for restless bandits. In *2019 Australian & New Zealand Control Conference (ANZCC)* (pp. 249–254).: IEEE.

[111] Futoma, J., Hughes, M. C., & Doshi-Velez, F. (2020). Popcorn: Partially observed prediction constrained reinforcement learning. *arXiv preprint arXiv:2001.04032*.

[112] Gan, J., An, B., & Vorobeychik, Y. (2015a). Security games with protection externalities. In *AAAI-15* (pp. 914–920). Austin.

[113] Gan, J., An, B., & Vorobeychik, Y. (2015b). Security games with protection externalities. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

[114] Gan, J., Elkind, E., Kraus, S., & Wooldridge, M. (2020). Mechanism design for defense coordination in security games. In *AAMAS*.

[115] Gan, J., Elkind, E., & Wooldridge, M. (2018). Stackelberg security games with multiple uncoordinated defenders. *AAMAS*.

[116] Garey, M. R. & Johnson, D. S. (1979). *Computers and intractability*, volume 174. freeman San Francisco.

[117] Garg, N. (2013). *Apache kafka*. Packt Publishing Birmingham.

[118] Garner, M., Hamilton, S., et al. (2011). Principles of epidemiological modelling. *Revue Scientifique et Technique-OIE*, 30(2), 407.

[119] Gemp, I. & Mahadevan, S. (2016). Online monotone optimization. *arXiv preprint arXiv:1608.07888*.

[120] Gerstgrasser, M. & Parkes, D. C. (2022). Oracles & followers: Stackelberg equilibria in deep multi-agent reinforcement learning. *arXiv preprint arXiv:2210.11942*.

[121] Gittins, J., Glazebrook, K., & Weber, R. (2011). *Multi-armed bandit allocation indices*. John Wiley & Sons.

[122] Glazebrook, K. D., Ruiz-Hernandez, D., & Kirkbride, C. (2006a). Some indexable families of restless bandit problems. *Advances in Applied Probability*, 38(3), 643–672.

[123] Glazebrook, K. D., Ruiz-Hernandez, D., & Kirkbride, C. (2006b). Some indexable families of restless bandit problems. *Advances in Applied Probability*, 38(3), 643–672.

[124] Greif, C., Moulding, E., & Orban, D. (2014). Bounds on eigenvalues of matrices arising from interior-point methods. *SIAM Journal on Optimization*, 24(1), 49–83.

[125] Gutfraind, A., Hagberg, A., & Pan, F. (2009). Optimal interdiction of unreactive markovian evaders. In *CPAIOR-09* (pp. 102–116). Pittsburgh.

[126] Gutfraind, A., Hagberg, A. A., Izraelevitz, D., & Pan, F. (2011). Interdiction of a markovian evader. In *Proc. of INFORMS Computing Society* Monterey, CA.

[127] Haarnoja, T., Tang, H., Abbeel, P., & Levine, S. (2017). Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning* (pp. 1352–1361).: PMLR.

[128] Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning* (pp. 1861–1870).: PMLR.

[129] Haghtalab, N., Fang, F., Nguyen, T. H., Sinha, A., Procaccia, A. D., & Tambe, M. (2016). Three strategies to success: Learning adversary models in security games. In *IJCAI-16* (pp. 308–314). New York.

[130] Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *NIPS-17* (pp. 1024–1034). Long Beach.

[131] Harper, F. M. & Konstan, J. A. (2015). The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4), 1–19.

[132] Hazan, E. (2019). Introduction to online convex optimization. *arXiv preprint arXiv:1909.05207*.

[133] Hazan, E., Agarwal, A., & Kale, S. (2007a). Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3), 169–192.

[134] Hazan, E., Rakhlin, A., & Bartlett, P. (2007b). Adaptive online gradient descent. *Advances in Neural Information Processing Systems*, 20.

[135] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T.-S. (2017). Neural collaborative filtering. In *Proceedings of the Twenty-Sixth International Conference on World Wide Web*.

[136] Hochbaum, D. S. & Shmoys, D. B. (1987). Using dual approximation algorithms for scheduling problems theoretical and practical results. *Journal of the ACM (JACM)*, 34(1), 144–162.

[137] Holland, P. W., Laskey, K. B., & Leinhardt, S. (1983). Stochastic blockmodels: First steps. *Social Networks*, 5(2), 109–137.

[138] Houk, V. N., Oakley, G. P., Erickson, J. D., Mulinare, J., & James, L. M. (1992). Recommendations for the use of folic acid to reduce the number of cases of spina bifida and other neural tube defects. *The Morbidity and Mortality Weekly Report (MMWR)*.

[139] Hoyert, D. L. (2022). Maternal mortality rates in the united states, 2020. *Health E-Stats*.

[140] Hsu, Y.-P. (2018). Age of information: Whittle index for scheduling stochastic arrivals. In *2018 IEEE International Symposium on Information Theory (ISIT)* (pp. 2634–2638).: IEEE.

[141] Hu, M. & Fukushima, M. (2011). Variational inequality formulation of a class of multi-leader-follower games. *Journal of optimization theory and applications*, 151(3), 455–473.

[142] Huang, C., Zhai, S., Talbott, W., Martin, M. B., Sun, S.-Y., Guestrin, C., & Susskind, J. (2019). Addressing the loss-metric mismatch with adaptive loss alignment. In *International Conference on Machine Learning* (pp. 2891–2900).: PMLR.

[143] Immorlica, N., Sankararaman, K. A., Schapire, R., & Slivkins, A. (2019). Adversarial bandits with knapsacks. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)* (pp. 202–219).: IEEE.

[144] Isinkaye, F., Folajimi, Y., & Ojokoh, B. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3), 261–273.

[145] Jaggi, M. (2013). Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proceedings of the Thirtieth International Conference on Machine Learning*.

[146] Jain, M., Kardes, E., Kiekintveld, C., Ordónez, F., & Tambe, M. (2010). Security games with arbitrary schedules: A branch and price approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24 (pp. 792–797).

[147] Jain, M., Korzhyk, D., Vaněk, O., Conitzer, V., Pěchouček, M., & Tambe, M. (2011). A double oracle algorithm for zero-sum security games on graphs. In *AAMAS-11* (pp. 327–334). Taipei.

[148] Jaksch, T., Auer, P., & Ortner, R. (2010). Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11, 1563–1600.

[149] Jia, Y., Xu, W., & Liu, X. (2017). An optimization framework for online ride-sharing markets. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)* (pp. 826–835).: IEEE.

[150] Jiang, A. X., Procaccia, A. D., Qian, Y., Shah, N., & Tambe, M. (2013a). Defender (mis) coordination in security games. *IJCAI*.

[151] Jiang, A. X., Yin, Z., Zhang, C., Tambe, M., & Kraus, S. (2013b). Game-theoretic randomization for security patrolling with dynamic execution uncertainty. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems* (pp. 207–214).: International Foundation for Autonomous Agents and Multiagent Systems.

[152] Jiang, S., Song, Z., Weinstein, O., & Zhang, H. (2020). Faster dynamic matrix inverse for faster lps. *arXiv preprint arXiv:2004.07470*.

[153] Jit, M. & Brisson, M. (2011). Modelling the epidemiology of infectious diseases for decision analysis: a primer. *Pharmacoeconomics*, 29, 371–386.

[154] Johnson, B., Böhme, R., & Grossklags, J. (2011). Security games with market insurance. In *GameSec*: Springer.

[155] Johnson, J. M. & Khoshgoftaar, T. M. (2019). Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1), 1–54.

[156] Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4), 455–492.

[157] Joseph, K., Boutin, A., Lisonkova, S., Muraca, G. M., Razaz, N., John, S., Mehrabadi, A., Sabr, Y., Ananth, C. V., & Schisterman, E. (2021). Maternal mortality in the united states: recent trends, current status, and future considerations. *Obstetrics and gynecology*, 137(5), 763.

[158] Jung, Y. H., Abeille, M., & Tewari, A. (2019). Thompson sampling in non-episodic restless bandits. *arXiv preprint arXiv:1910.05654*.

[159] Jung, Y. H. & Tewari, A. (2019). Regret bounds for Thompson sampling in episodic restless bandit problems. *Advances in Neural Information Processing Systems (NeurIPS)*, 32.

[160] Kadota, I., Uysal-Biyikoglu, E., Singh, R., & Modiano, E. (2016). Minimizing the age of information in broadcast wireless networks. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)* (pp. 844–851).: IEEE.

[161] Kandasamy, K., Schneider, J., & Póczos, B. (2015). High dimensional bayesian optimisation and bandits via additive models. In *International Conference on Machine Learning* (pp. 295–304).

[162] Kang, X. & Wu, Y. (2014). Incentive mechanism design for heterogeneous peer-to-peer networks: A stackelberg game approach. *IEEE Transactions on Mobile Computing*, 14(5), 1018–1030.

[163] Kar, D., Fang, F., Delle Fave, F., Sintov, N., & Tambe, M. (2015). A game of thrones: when human behavior models compete in repeated stackelberg security games. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems* (pp. 1381–1390).: International Foundation for Autonomous Agents and Multiagent Systems.

[164] Kar, D., Ford, B., Gholami, S., Fang, F., Plumptre, A., Tambe, M., Driciru, M., Wanyama, F., Rwetsiba, A., Nsubaga, M., et al. (2017). Cloudy with a chance of poaching: Adversary behavior modeling and forecasting with real-world poaching data. In *Proceedings of the Sixteenth International Conference on Autonomous Agents and Multiagent Systems*.

[165] Karkus, P., Ma, X., Hsu, D., Kaelbling, L. P., Lee, W. S., & Lozano-Pérez, T. (2019). Differentiable algorithm networks for composable robot learning. *arXiv preprint arXiv:1905.11602*.

[166] Kaur, J., Kaur, M., Chakrapani, V., Webster, J., Santos, J., & Kumar, R. (2020). Effectiveness of information technology–enabled 'smart eating' health promotion intervention: A cluster randomized controlled trial. *PLOS ONE*, 15, e0225892.

[167] Kidambi, R., Rajeswaran, A., Netrapalli, P., & Joachims, T. (2020). Morel: Model-based offline reinforcement learning. *arXiv preprint arXiv:2005.05951*.

[168] Kiekintveld, C., Lisỳ, V., & Píbil, R. (2015). Game-theoretic foundations for the strategic use of honeypots in network security. In *Cyber Warfare* (pp. 81–101). Springer.

[169] Killian, J. A., Biswas, A., Shah, S., & Tambe, M. (2021). Q-learning Lagrange policies for multi-action restless bandits. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD)* (pp. 871–881).

[170] Killian, J. A., Wilder, B., Sharma, A., Choudhary, V., Dilkina, B., & Tambe, M. (2019). Learning to prescribe interventions for tuberculosis patients using digital adherence data. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 2430–2438).

[171] Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[172] Kipf, T. N. & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *ICLR-17* Toulon.

[173] Klíma, R., Lisỳ, V., & Kiekintveld, C. (2015). Combining online learning and equilibrium computation in security games. In *International Conference on Decision and Game Theory for Security* (pp. 130–149).: Springer.

[174] Koivu, A. & Sairanen, M. (2020). Predicting risk of stillbirth and preterm pregnancies with machine learning. *Health information science and systems*, 8, 1–12.

[175] Koolen, W. M., Warmuth, M. K., Kivinen, J., et al. (2010). Hedging structured concepts. In *COLT* (pp. 93–105).

[176] Korzhyk, D., Conitzer, V., & Parr, R. (2010). Complexity of computing optimal stackelberg strategies in security resource allocation games. In *AAAI-10* Atlanta.

[177] Kraft, D. (1985). On converting optimal control problems into nonlinear programming problems. In *Computational mathematical programming* (pp. 261–280). Springer.

[178] Kraft, D. (1988). A software package for sequential quadratic programming. *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*.

[179] Krawczyk, J. B. & Uryasev, S. (2000). Relaxation algorithms to find Nash equilibria with economic applications. *Environmental Modeling & Assessment*, 5(1), 63–73.

[180] Krishnan, S., Garg, A., Patil, S., Lea, C., Hager, G., Abbeel, P., & Goldberg, K. (2017). Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning. *The International Journal of Robotics Research*, 36(13-14), 1595–1618.

[181] Krishnasamy, S., Akhil, P., Arapostathis, A., Sundaresan, R., & Shakkottai, S. (2018). Augmenting max-weight with explicit learning for wireless scheduling with switching costs. *IEEE/ACM Transactions on Networking*, 26(6), 2501–2514.

[182] Kuhn, H. W. & Tucker, A. W. (2014). Nonlinear programming. In *Traces and emergence of nonlinear programming* (pp. 247–258). Springer.

[183] Kumar, P. R. & Varaiya, P. (2015). *Stochastic systems: Estimation, identification, and adaptive control*. SIAM.

[184] Kushner, H. J. (1964). A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1), 97–106.

[185] Lambert, N., Amos, B., Yadan, O., & Calandra, R. (2020). Objective mismatch in model-based reinforcement learning. *arXiv preprint arXiv:2002.04523*.

[186] Lange, K., Hunter, D. R., & Yang, I. (2000). Optimization transfer using surrogate objective functions. *Journal of Computational and Hraphical Statistics*, 9(1), 1–20.

[187] Lazaric, A. (2012). Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning* (pp. 143–173). Springer.

[188] Lee, H. & Cho, C.-H. (2020). Digital advertising: present and future prospects. *International Journal of Advertising*, 39(3), 332–341.

[189] Leitmann, G. (1978). On generalized stackelberg strategies. *Journal of Optimization Theory and Applications*, 26(4), 637–643.

[190] Letchford, J., Conitzer, V., & Munagala, K. (2009). Learning and approximating the optimal strategy to commit to. In M. Mavronicolas & V. G. Papadopoulou (Eds.), *Algorithmic Game Theory* (pp. 250–262). Berlin, Heidelberg: Springer Berlin Heidelberg.

[191] Leung, J. Y. (1989). Bin packing with restricted piece sizes. *Information Processing Letters*, 31(3), 145–149.

[192] Li, J., Yu, J., Nie, Y., & Wang, Z. (2020a). End-to-end learning and intervention in games. *NeurIPS*.

[193] Li, Y., Du, D., Xiu, N., & Xu, D. (2015). Improved approximation algorithms for the facility location problems with linear/submodular penalties. *Algorithmica*, 73(2), 460–482.

[194] Li, Y. & Li, N. (2020). Leveraging predictions in smoothed online convex optimization via gradient-based algorithms. *arXiv preprint arXiv:2011.12539*.

[195] Li, Y., Qu, G., & Li, N. (2020b). Online optimization with predictions and switching costs: Fast algorithms and the fundamental limit. *IEEE Transactions on Automatic Control*.

[196] Li, Z., Jia, F., Mate, A., Jabbari, S., Chakraborty, M., Tambe, M., & Vorobeychik, Y. (2021). Solving structured hierarchical games using differential backward induction. *CoRR*, abs/2106.04663.

[197] Lin, M., Liu, Z., Wierman, A., & Andrew, L. L. (2012a). Online algorithms for geographical load balancing. In *2012 international green computing conference (IGCC)* (pp. 1–10).: IEEE.

[198] Lin, M., Wierman, A., Andrew, L. L., & Thereska, E. (2012b). Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Transactions on Networking*, 21(5), 1378–1391.

[199] Lina, M. & Jacqueline, M. (1996). Hierarchical systems with weighted reaction set. In *Nonlinear optimization and Applications* (pp. 271–282). Springer.

[200] Ling, C. K., Fang, F., & Kolter, J. Z. (2018). What game are we playing? end-to-end learning in normal and extensive form games. *arXiv preprint arXiv:1805.02777*.

[201] Ling, C. K., Fang, F., & Kolter, J. Z. (2019). Large scale learning of agent rationality in two-player zero-sum games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33 (pp. 6104–6111).

[202] Liu, B. (1998). Stackelberg-Nash equilibrium for multilevel programming with multiple followers using genetic algorithms. *Computers & Mathematics with Applications*, 36(7), 79–89.

[203] Liu, H., Liu, K., & Zhao, Q. (2012). Learning in a changing world: Restless multiarmed bandit with unknown dynamics. *IEEE Transactions on Information Theory*, 59(3), 1902–1916.

[204] Liu, K. & Zhao, Q. (2010). Indexability of restless bandit problems and optimality of whittle index for dynamic multichannel access. *IEEE Transactions on Information Theory*, 56(11), 5547–5567.

[205] Luo, Z.-Q., Pang, J.-S., & Ralph, D. (1996). *Mathematical programs with equilibrium constraints*. Cambridge University Press.

[206] Mandi, J. & Guns, T. (2020). Interior point solving for lp-based prediction+ optimisation. *arXiv preprint arXiv:2010.13943*.

[207] Mandi, J., Stuckey, P. J., Guns, T., et al. (2020). Smart predict-and-optimize for hard combinatorial optimization problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34 (pp. 1603–1610).

[208] Markowitz, H. M. & Todd, G. P. (2000). *Mean-variance Analysis in Portfolio Choice and Capital Markets*, volume 66. John Wiley & Sons.

[209] Mate, A., Biswas, A., Siebenbrunner, C., & Tambe, M. (2022a). Efficient algorithms for finite horizon and streaming restless multi-armed bandit problems. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[210] Mate, A., Killian, J. A., Xu, H., Perrault, A., & Tambe, M. (2020). Collapsing bandits and their application to public health interventions. *arXiv preprint arXiv:2007.04432*.

[211] Mate, A., Madaan, L., Taneja, A., Madhiwalla, N., Verma, S., Singh, G., Hegde, A., Varakantham, P., & Tambe, M. (2022b). Field study in deploying restless multi-armed bandits: Assisting non-profits in improving maternal and child health. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[212] Mattingley, J., Wang, Y., & Boyd, S. (2011). Receding horizon control. *IEEE Control Systems Magazine*, 31(3), 52–65.

[213] Mc Carthy, S. M., Tambe, M., Kiekintveld, C., Gore, M. L., & Killion, A. (2016). Preventing illegal logging: Simultaneous optimization of resource teams and tactics for security. In *AAAI-16* New York.

[214] McKelvey, R. D. & Palfrey, T. R. (1995). Quantal response equilibria for normal form games. *Games and Economic Behavior*, 10(1), 6–38.

[215] Mertikopoulos, P. & Zhou, Z. (2019). Learning in games with continuous action sets and unknown payoff functions. *Mathematical Programming*, 173(1), 465–507.

[216] Meshram, R., Gopalan, A., & Manjunath, D. (2016). Optimal recommendation to users that react: Online learning for a class of pomdps. In *2016 IEEE 55th Conference on Decision and Control (CDC)* (pp. 7210–7215).: IEEE.

[217] Michaud, R. O. (1989). The Markowitz optimization enigma: Is 'optimized' optimal? *Financial Analysts Journal*, 45(1), 31–42.

[218] Močkus, J. (1975). On bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference* (pp. 400–404).: Springer.

[219] Monahan, G. E. (1982). State of the art—a survey of partially observable markov decision processes: theory, models, and algorithms. *Management science*, 28(1), 1–16.

[220] Moore, J. F., Mulindahabi, F., Masozera, M. K., Nichols, J. D., Hines, J. E., Turikunkiko, E., & Oli, M. K. (2018). Are ranger patrols effective in reducing poaching-related threats within protected areas? *Journal of Applied Ecology*, 55(1), 99–107.

[221] Moreto, W. D. & Gau, J. M. (2017). Deterrence, legitimacy, and wildlife crime in protected areas. *Conservation criminology*, (pp. 45–58).

[222] Morgan, J. & Vardy, F. (2004). An experimental study of commitment and observability in stackelberg games. *Games and Economic Behavior*, 49(2), 401–423.

[223] Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., & Grohe, M. (2019). Weisfeiler and Leman go neural: Higher-order graph neural networks. In *AAAI-19* (pp. 4602–4609). Honolulu.

[224] Mortensen, D. T. & Pissarides, C. A. (2001). Taxes, subsidies and equilibrium labour market outcomes. *Available at SSRN 287319*.

[225] Morton, D. P., Pan, F., & Saeger, K. J. (2007). Models for nuclear smuggling interdiction. *IIE Transactions*, 39(1), 3–14.

[226] Mullikin, M., Tan, L., Jansen, J. P., Van Ranst, M., Farkas, N., & Petri, E. (2015). A novel dynamic model for health economic analysis of influenza vaccination in the elderly. *Infectious diseases and therapy*, 4(4), 459–487.

[227] Murugan, P. & Abebaw, B. (2014). Factors contributing to human trafficking, contexts of vulnerability and patterns of victimization: the case of stranded victims in metema, ethiopia. *Ethiopian Journal of the Social Sciences and Humanities*, 10(2), 75–105.

[228] Naghizadeh, P. & Liu, M. (2014). Voluntary participation in cyber-insurance markets. In *Workshop on the Economics of Information Security (WEIS)*.

[229] Nagy, A. M. & Simon, V. (2018). Survey on traffic prediction in smart cities. *Pervasive and Mobile Computing*, 50, 148–163.

[230] Nakamura, T. (2015). One-leader and multiple-follower stackelberg games with private information. *Economics Letters*, 127, 27–30.

[231] Nakhleh, K., Ganji, S., Hsieh, P.-C., Hou, I., Shakkottai, S., et al. (2021). NeurWIN: Neural whittle index network for restless bandits via deep RL. *Advances in Neural Information Processing Systems*, 34, 828–839.

[232] Nehme, M. V. (2009). *Two-person games for stochastic network interdiction: models, methods, and complexities*. The University of Texas at Austin.

[233] Neu, G. & Bartók, G. (2013). An efficient algorithm for learning with semi-bandit feedback. In *International Conference on Algorithmic Learning Theory (ALT)* (pp. 234–248).: Springer.

[234] Ng, A. Y., Harada, D., & Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99 (pp. 278–287).

[235] Nguyen, T., Wellman, M. P., & Singh, S. (2017). A stackelberg game model for botnet data exfiltration. In *International Conference on Decision and Game Theory for Security* (pp. 151–170).: Springer.

[236] Nguyen, T. H., Yang, R., Azaria, A., Kraus, S., & Tambe, M. (2013). Analyzing the effectiveness of adversary modeling in security games. In *AAAI-13* Bellevue, Washington.

[237] Ning, C. & You, F. (2019). Optimization under uncertainty in the era of big data and deep learning: When machine learning meets mathematical programming. *Computers & Chemical Engineering*, 125, 434–448.

[238] Nocedal, J. (2006). Knitro: an integrated package for nonlinear optimization. In *Large-Scale Nonlinear Optimization* (pp. 35–60). Springer.

[239] Nour, N. M. (2008). An introduction to maternal mortality. *Reviews in obstetrics and gynecology*, 1(2), 77.

[240] Okamoto, S., Hazon, N., & Sycara, K. (2012). Solving non-zero sum multiagent network flow security games with attack costs. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2* (pp. 879–888).: International Foundation for Autonomous Agents and Multiagent Systems.

[241] Oksanen, J. & Koivunen, V. (2015). An order optimal policy for exploiting idle spectrum in cognitive radio networks. *IEEE Transactions on Signal Processing*, 63(5), 1214–1227.

[242] Ortner, R., Ryabko, D., Auer, P., & Munos, R. (2012). Regret bounds for restless Markov bandits. In *International Conference on Algorithmic Learning Theory (ALT)* (pp. 214–228).: Springer.

[243] Pan, F., Charlton, W. S., & Morton, D. P. (2003). A stochastic program for interdicting smuggled nuclear material. In *Network interdiction and stochastic integer programming* (pp. 1–19). Springer.

[244] Papadimitriou, C. H. & Tsitsiklis, J. N. (1994). The complexity of optimal queueing network control. In *Proceedings of IEEE 9th Annual Conference on Structure in Complexity Theory* (pp. 318–322).: IEEE.

[245] Parise, F. & Ozdaglar, A. (2019). A variational inequality framework for network games: Existence, uniqueness, convergence and sensitivity analysis. *Games and Economic Behavior*, 114, 47–82.

[246] Patrascu, A. & Necoara, I. (2015). Efficient random coordinate descent algorithms for large-scale structured nonconvex optimization. *Journal of Global Optimization*, 61(1), 19–46.

[247] Perrault, A., Wilder, B., Ewing, E., Mate, A., Dilkina, B., & Tambe, M. (2020a). End-to-end game-focused learning of adversary behavior in security games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34 (pp. 1378–1386).

[248] Perrault, A., Wilder, B., Ewing, E., Mate, A., Dilkina, B., & Tambe, M. (2020b). End-to-end game-focused learning of adversary behavior in security games. In *AAAI-2020* New York.

[249] Perrault, A., Wilder, B., Ewing, E., Mate, A., Dilkina, B., & Tambe, M. (2020c). End-to-end game-focused learning of adversary behavior in security games. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*.

[250] Pfammatter, A., Spring, B., Saligram, N., Davé, R., Gowda, A., Blais, L., Arora, M., Ranjani, H., Ganda, O., Hedeker, D., Reddy, S., & Ramalingam, S. (2016). mhealth intervention to improve diabetes risk behaviors in india: A prospective, parallel group cohort study. *Journal of Medical Internet Research*, 18, e207.

[251] Pirnay, H., López-Negrete, R., & Biegler, L. T. (2012). Optimal sensitivity based on ipopt. *Mathematical Programming Computation*, 4(4), 307–331.

[252] Pita, J., Jain, M., Ordóñez, F., Portway, C., Tambe, M., Western, C., Paruchuri, P., & Kraus, S. (2009). Using game theory for los angeles airport security. *AI Magazine*, 30(1), 43–57.

[253] Plumptre, A. J., Fuller, R. A., Rwetsiba, A., Wanyama, F., Kujirakwinja, D., Driciru, M., Nangendo, G., Watson, J. E., & Possingham, H. P. (2014). Efficiently targeting resources to deter illegal activities in protected areas. *Journal of Applied Ecology*, 51(3), 714–725.

[254] Popescu, I. (2007). Robust mean-covariance solutions for stochastic optimization. *Operations Research*, 55(1), 98–112.

[255] Prabhu, Y. & Varma, M. (2014). Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 263–272).

[256] Prechelt, L. (1998). Early stopping-but when? In *Neural Networks: Tricks of the trade* (pp. 55–69). Springer.

[257] Precup, D. (2000). Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, (pp.80).

[258] Qian, Y., Zhang, C., Krishnamachari, B., & Tambe, M. (2016). Restless poachers: Handling exploration-exploitation tradeoffs in security domains. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems* (pp. 123–131).

[259] Quandl (2020). WIKI various end-of-day data.

[260] Queipo, N. V., Haftka, R. T., Shyy, W., Goel, T., Vaidyanathan, R., & Tucker, P. K. (2005). Surrogate-based analysis and optimization. *Progress in Aerospace Sciences*, 41(1), 1–28.

[261] Quijano-Sánchez, L., Cantador, I., Cortés-Cediel, M. E., & Gil, O. (2020). Recommender systems for smart cities. *Information systems*, 92, 101545.

[262] Raffin, A., Hill, A., Ernestus, M., Gleave, A., Kanervisto, A., & Dormann, N. (2019). Stable baselines3. https://github.com/DLR-RM/stable-baselines3.

[263] Ranchod, P., Rosman, B., & Konidaris, G. (2015). Nonparametric bayesian reward segmentation for skill discovery using inverse reinforcement learning. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 471–477).: IEEE.

[264] Rasmussen, C. E. (2004). Gaussian processes in machine learning. In *Advanced lectures on machine learning* (pp. 63–71). Springer.

[265] Robinson, T., Eldred, M., Willcox, K., & Haimes, R. (2008). Surrogate-based optimization using multifidelity models with variable parameterization and corrected space mapping. *AIAA Journal*, 46(11), 2814–2822.

[266] Ronsmans, C. & Graham, W. J. (2006). Maternal mortality: who, when, where, and why. *The lancet*, 368(9542), 1189–1200.

[267] Rosen, G. E. & Smith, K. F. (2010). Summarizing the evidence on the international trade in illegal wildlife. *EcoHealth*, 7(1), 24–32.

[268] Rosenfeld, A. & Kraus, S. (2017). When security games hit traffic: optimal traffic enforcement under one sided uncertainty. In *Proceedings of the 26th International Conference on Artificial Intelligence, IJCAI*.

[269] Rosenfeld, A., Maksimov, O., & Kraus, S. (2017). Optimizing traffic enforcement: From the lab to the roads. In *International Conference on Decision and Game Theory for Security* (pp. 3–20).: Springer.

[270] Rotemberg, M. (2019). Equilibrium effects of firm subsidies. *American Economic Review*, 109(10), 3475–3513.

[271] Roughgarden, T. (2004). Stackelberg scheduling strategies. *SIAM journal on computing*, 33(2), 332–350.

[272] Roy, S., Ellis, C., Shiva, S., Dasgupta, D., Shandilya, V., & Wu, Q. (2010). A survey of game theory as applied to network security. In *2010 43rd Hawaii International Conference on System Sciences* (pp. 1–10).: IEEE.

[273] Sah, P., Medlock, J., Fitzpatrick, M. C., Singer, B. H., & Galvani, A. P. (2018). Optimizing the impact of low-efficacy influenza vaccines. *Proceedings of the National Academy of Sciences*, 115(20), 5151–5156.

[274] Sahinidis, N. V. (2004). Optimization under uncertainty: state-of-the-art and opportunities. *Computers & Chemical Engineering*, 28(6-7), 971–983.

[275] Saura, J. R. (2021). Using data sciences in digital marketing: Framework, methods, and performance metrics. *Journal of Innovation & Knowledge*, 6(2), 92–102.

[276] Schaul, T., Horgan, D., Gregor, K., & Silver, D. (2015). Universal value function approximators. In *International conference on machine learning* (pp. 1312–1320).: PMLR.

[277] Scholl, T. O., Hediger, M. L., Fischer, R. L., & Shearer, J. W. (1992). Anemia vs iron deficiency: increased risk of preterm delivery in a prospective study. *The American journal of clinical nutrition*, 55(5), 985–988.

[278] Sellke, M. (2020). Chasing convex bodies optimally. In *SODA* (pp. 1509–1518).: SIAM.

[279] Shalev-Shwartz, S. et al. (2011). Online learning and online convex optimization. *Foundations and trends in Machine Learning*, 4(2), 107–194.

[280] Shi, C., Zhang, G., & Lu, J. (2005). The kth-best approach for linear bilevel multi-follower programming. *Journal of Global Optimization*, 33(4), 563–578.

[281] Shieh, E., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., Maule, B., & Meyer, G. (2012). PROTECT: a deployed game theoretic system to protect the ports of the united states. In *International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, Valencia, Spain, June 4-8, 2012 (3 Volumes)* (pp. 13–20).

[282] Shieh, E., Jain, M., Jiang, A. X., & Tambe, M. (2013). Efficiently solving joint activity based security games. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence* (pp. 346–352).: AAAI Press.

[283] Shiva, S., Roy, S., & Dasgupta, D. (2010). Game theory for cyber security. In *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research* (pp.34).: ACM.

[284] Sinha, A., Fang, F., An, B., Kiekintveld, C., & Tambe, M. (2018). Stackelberg security games: Looking beyond a decade of success. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*.

[285] Sinha, A., Kar, D., & Tambe, M. (2016). Learning adversary behavior in security games: A PAC model perspective. In *AAMAS-16* (pp. 214–222). Singapore.

[286] Sinha, A., Malo, P., Frantsev, A., & Deb, K. (2014). Finding optimal strategies in a multi-period multi-leader–follower stackelberg game using an evolutionary algorithm. *Computers & Operations Research*, 41, 374–385.

[287] Sinha, A., Soun, T., & Deb, K. (2019). Using karush-kuhn-tucker proximity measure for solving bilevel optimization problems. *Swarm and evolutionary computation*, 44, 496–510.

[288] Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems* (pp. 2951–2959).

[289] Solis, C. U., Clempner, J. B., & Poznyak, A. S. (2016). Modeling multileader–follower non-cooperative stackelberg games. *Cybernetics and Systems*, 47(8), 650–673.

[290] Spielman, D. A. (2007). Spectral graph theory and its applications. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)* (pp. 29–38).: IEEE.

[291] Srebro, N., Sridharan, K., & Tewari, A. (2011). On the universality of online mirror descent. In *NIPS* (pp. 2645–2653).

[292] Srinivas, N., Krause, A., Kakade, S. M., & Seeger, M. (2009). Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*.

[293] Staiger, H., Laschewski, G., & Grätz, A. (2012). The perceived temperature–a versatile index for the assessment of the human thermal environment. part a: scientific basics. *International journal of biometeorology*, 56(1), 165–176.

[294] Sun, Y., Feng, G., Qin, S., & Sun, S. (2018). Cell association with user behavior awareness in heterogeneous cellular networks. *IEEE Transactions on Vehicular Technology*, 67(5), 4589–4601.

[295] Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990* (pp. 216–224). Elsevier.

[296] Sutton, R. S., Barto, A. G., et al. (1998). *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

[297] Tambe, M. (2011). *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press.

[298] Tan, M., Xia, T., Guo, L., & Wang, S. (2013). Direct optimization of ranking measures for learning to rank models. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 856–864).

[299] Taylor, M. E. & Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7).

[300] Tekin, C. & Liu, M. (2012). Online learning of rested and restless bandits. *IEEE Transactions on Information Theory*, 58(8), 5588–5611.

[301] Teodorescu, I. (2009). Maximum likelihood estimation for markov chains. *arXiv preprint arXiv:0905.4131*.

[302] Thein, K. M. M. (2014). Apache kafka: Next generation distributed messaging system. *International Journal of Scientific Engineering and Technology Research*, 3(47), 9478–9483.

[303] Thomas, P. & Brunskill, E. (2016). Data-efficient off-policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning* (pp. 2139–2148).: PMLR.

[304] Thomas, P. S. (2015). *Safe reinforcement learning*. PhD thesis, University of Massachusetts Libraries.

[305] Tseng, P. (2001). Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3), 475–494.

[306] Ui, T. (2016). Bayesian nash equilibrium and variational inequalities. *Journal of Mathematical Economics*, 63, 139–146.

[307] Uryasev, S. & Rubinstein, R. Y. (1994). On relaxation algorithms in computation of noncooperative equilibria. *IEEE Transactions on Automatic Control*, 39(6), 1263–1267.

[308] Vadlamudi, S. G., Sengupta, S., Taguinod, M., Zhao, Z., Doupé, A., Ahn, G.-J., & Kambhampati, S. (2016). Moving target defense for web applications using bayesian stackelberg games. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems* (pp. 1377–1378).: International Foundation for Autonomous Agents and Multiagent Systems.

[309] Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. In *AAAI*.

[310] Varakantham, P., Lau, H. C., & Yuan, Z. (2013). Scalable randomized patrolling for securing rapid transit networks. In *IAAI*.

[311] Verma, S., Mate, A., Wang, K., Taneja, A., & Tambe, M. (2022). Case study: Applying decision focused learning in the real world. *NeurIPS Workshop on Trustworthy and Socially Responsible Machine Learning (TSRML)*.

[312] Villar, S. S., Bowden, J., & Wason, J. (2015). 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.

[313] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3), 261–272.

[314] Von Essen, E., Hansen, H. P., Nordström Källström, H., Peterson, M. N., & Peterson, T. R. (2014). Deconstructing the poaching phenomenon: A review of typologies for understanding illegal hunting. *British Journal of Criminology*, 54(4), 632–651.

[315] Von Stengel, B. & Zamir, S. (2004). *Leadership with commitment to mixed strategies*. Technical report, Citeseer.

[316] Vynnycky, E., Pitman, R., Siddiqui, R., Gay, N., & Edmunds, W. J. (2008). Estimating the impact of childhood influenza vaccination programmes in england and wales. *Vaccine*, 26(41), 5321–5330.

[317] Wan, J., Liu, J., Shao, Z., Vasilakos, A. V., Imran, M., & Zhou, K. (2016). Mobile crowd sensing for traffic prediction in internet of vehicles. *Sensors*, 16(1), 88.

[318] Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., & Botvinick, M. (2016a). Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*.

[319] Wang, K., Guo, Q., Vayanos, P., Tambe, M., & An, B. (2018). Equilibrium refinement in security games with arbitrary scheduling constraints. In *International Conference on Autonomous Agents and Multiagent Systems, AAMAS*.

[320] Wang, K., Perrault, A., Mate, A., & Tambe, M. (2020a). Scalable game-focused learning of adversary models: Data-to-decisions in network security games. In *Proceedings of the Nineteenth International Conference on Autonomous Agents and Multiagent Systems*.

[321] Wang, K., Shah, S., Chen, H., Perrault, A., Doshi-Velez, F., & Tambe, M. (2021). Learning mdps from features: Predict-then-optimize for sequential decision making by reinforcement learning. *Advances in Neural Information Processing Systems*, 34.

[322] Wang, K., Song, Z., Theocharous, G., & Mahadevan, S. (2023a). Smoothed online combinatorial optimization using imperfect predictions. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[323] Wang, K., Verma, S., Mate, A., Shah, S., Taneja, A., Madhiwalla, N., Hegde, A., & Tambe, M. (2022a). Decision-focused learning in restless multi-armed bandits with application to maternal and child care domain. *arXiv preprint arXiv:2202.00916*.

[324] Wang, K., Wilder, B., Perrault, A., & Tambe, M. (2020b). Automatically learning compact quality-aware surrogates for optimization problems. *arXiv preprint arXiv:2006.10815*.

[325] Wang, K., Wilder, B., Suen, S.-c., Dilkina, B., & Tambe, M. (2019a). Improving gp-ucb algorithm by harnessing decomposed feedback. In *Machine Learning and Knowledge Discovery in Databases* (pp. 555–569). Springer International Publishing.

[326] Wang, K., Xu, L., Perrault, A., Reiter, M. K., & Tambe, M. (2022b). Coordinating followers to reach better equilibria: End-to-end gradient descent for stackelberg games. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[327] Wang, K., Xu, L., Taneja, A., & Tambe, M. (2023b). Optimistic whittle index policy: Online learning for restless bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[328] Wang, K., Yu, J., Chen, L., Zhou, P., Ge, X., & Win, M. Z. (2019b). Opportunistic scheduling revisited using restless bandits: Indexability and index policy. *IEEE Transactions on Wireless Communications*, 18(10), 4997–5010.

[329] Wang, S., Huang, L., & Lui, J. (2020c). Restless-UCB, an efficient and low-complexity algorithm for online restless bandits. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 11878–11889.

[330] Wang, Y., Ge, Y., Li, L., Chen, R., & Xu, T. (2020d). Offline meta-level model-based reinforcement learning approach for cold-start recommendation. *arXiv preprint arXiv:2012.02476*.

[331] Wang, Z., Zhou, B., & Jegelka, S. (2016b). Optimization as estimation with gaussian processes in bandit settings. In *Artificial Intelligence and Statistics* (pp. 1022–1031).

[332] Washburn, A. & Wood, K. (1995). Two-person zero-sum games for network interdiction. *Operations Research*, 43(2), 243–251.

[333] Waxman, B. M. (1988). Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9), 1617–1622.

[334] Weber, R. R. & Weiss, G. (1990). On an index policy for restless bandits. *Journal of applied probability*, 27(3), 637–648.

[335] Weissman, T., Ordentlich, E., Seroussi, G., Verdu, S., & Weinberger, M. J. (2003). Inequalities for the L1 deviation of the empirical distribution. *Hewlett-Packard Labs, Tech. Rep*.

[336] Wellsmith, M. (2011). Wildlife crime: the problems of enforcement. *European Journal on Criminal Policy and Research*, 17, 125–148.

[337] Whittle, P. (1988). Restless bandits: Activity allocation in a changing world. *Journal of applied probability*, 25(A), 287–298.

[338] Wilder, B., Dilkina, B., & Tambe, M. (2019). Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33 (pp. 1658–1665).

[339] Wilder, B., Suen, S.-C., & Tambe, M. (2018). Preventing infectious disease in dynamic populations under uncertainty. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

[340] Woodbury, M. A. (1950). *Inverting modified matrices*. Statistical Research Group.

[341] Woolthuis, R. G., Wallinga, J., & van Boven, M. (2017). Variation in loss of immunity shapes influenza epidemics and the impact of vaccination. *BMC infectious diseases*, 17(1), 632.

[342] World Bank, . (2020). *Poverty and shared prosperity 2020: Reversals of fortune*. The World Bank.

[343] Wyler, L. S. & Sheikh, P. A. (2008). International illegal trade in wildlife: threats and us policy. In *Congressional Research Service Report for Congress*, volume 1: Library of Congress Washington DC Congressional Research Service.

[344] Xiang, X., Li, Q., Khan, S., & Khalaf, O. I. (2021). Urban water resource management for sustainable environment planning using artificial intelligence techniques. *Environmental Impact Assessment Review*, 86, 106515.

[345] Xiao, G., Xiao, M., Gao, G., Zhang, S., Zhao, H., & Zou, X. (2020). Incentive mechanism design for federated learning: A two-stage stackelberg game approach. In *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)* (pp. 148–155).: IEEE.

[346] Xie, Y., Dai, H., Chen, M., Dai, B., Zhao, T., Zha, H., Wei, W., & Pfister, T. (2020). Differentiable top-k operator with optimal transport. *arXiv preprint arXiv:2002.06504*.

[347] Xiong, G., Li, J., & Singh, R. (2022). Reinforcement learning augmented asymptotically optimal index policy for finite-horizon restless bandits. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI)*.

[348] Xu, L., Bondi, E., Fang, F., Perrault, A., Wang, K., & Tambe, M. (2021). Dual-mandate patrols: Multi-armed bandits for green security. *AAAI*.

[349] Xu, L., Gholami, S., Mc Carthy, S., Dilkina, B., Plumptre, A., Tambe, M., Singh, R., Nsubuga, M., Mabonga, J., Driciru, M., et al. (2020). Stay ahead of poachers: Illegal wildlife poaching prediction and patrol planning under uncertainty with field test evaluations (short version). In *2020 IEEE 36th International Conference on Data Engineering (ICDE)* (pp. 1898–1901).: IEEE.

[350] Yang, R., Fang, F., Jiang, A. X., Rajagopal, K., Tambe, M., & Maheswaran, R. (2012). Designing better strategies against human adversaries in network security games. In *AAMAS-12* Valencia.

[351] Yang, R., Kiekintveld, C., OrdóñEz, F., Tambe, M., & John, R. (2013). Improving resource allocation strategies against human adversaries in security games: An extended study. *Artificial Intelligence*, 195, 440–469.

[352] Yigitcanlar, T., Desouza, K. C., Butler, L., & Roozkhosh, F. (2020). Contributions and risks of artificial intelligence (ai) in building smarter cities: Insights from a systematic review of the literature. *Energies*, 13(6), 1473.

[353] Yin, Z., Jain, M., Tambe, M., & Ordónez, F. (2011). Risk-averse strategies for security games with execution and observational uncertainty. In *AAAI*.

[354] Yin, Z., Korzhyk, D., Kiekintveld, C., Conitzer, V., & Tambe, M. (2010). Stackelberg vs. nash in security games: Interchangeability, equivalence, and uniqueness. In *AAMAS-10* (pp. 1139–1146). Toronto.

[355] Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., & Finn, C. (2021). Combo: Conservative offline model-based policy optimization. *arXiv preprint arXiv:2102.08363*.

[356] Yue, Y., Finley, T., Radlinski, F., & Joachims, T. (2007). A support vector method for optimizing average precision. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 271–278).

[357] Zhang, C., Sinha, A., & Tambe, M. (2015). Keeping pace with criminals: Designing patrol allocation against adaptive opportunistic criminals. In *AAMAS-15* (pp. 1351–1359). Istanbul.

[358] Zhang, H., Xiao, Y., Cai, L. X., Niyato, D., Song, L., & Han, Z. (2016). A multi-leader multi-follower stackelberg game for resource management in lte unlicensed. *IEEE Transactions on Wireless Communications*, 16(1), 348–361.

[359] Zheng, Z., Song, L., & Han, Z. (2017). Bridge the gap between admm and stackelberg game: Incentive mechanism design for big data networks. *IEEE Signal Processing Letters*, 24(2), 191–195.

[360] Zhou, Z., Zhang, J., Liu, P., Li, Z., Georgiadis, M. C., & Pistikopoulos, E. N. (2013). A two-stage stochastic programming model for the optimal design of distributed energy systems. *Applied Energy*, 103, 135–144.

[361] Zimmerman, M. E. (2003). The black market for wildlife: Combating transnational organized crime in the illegal wildlife trade. *Vand. J. Transnat'l L.*, 36, 1657.

[362] Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *ICML* (pp. 928–936).

[363] Zintgraf, L., Shiarlis, K., Igl, M., Schulze, S., Gal, Y., Hofmann, K., & Whiteson, S. (2019). Varibad: A very good method for bayes-adaptive deep rl via meta-learning. In *International Conference on Learning Representations*.

[364] 't Sas-Rolfes, M., Challender, D. W., Hinsley, A., Veríssimo, D., & Milner-Gulland, E. (2019). Illegal wildlife trade: Scale, processes, and governance. *Annual Review of Environment and Resources*, 44, 201–228.

Appendix

A

Appendix to Chapter 2

A.1 Missing Proofs in Chapter 2

Theorem 1 (Policy gradient-based unbiased derivative estimate)**.** *We follow the notation of Definition 1 and define* $\Phi_\theta(\tau, \pi) = \sum_{i=1}^{h} \sum_{j=i}^{h} \gamma^j R_\theta(s_j, a_j) \log \pi(a_i|s_i)$*. We have:*

$$\nabla_\pi J_\theta(\pi) = \mathop{\mathbb{E}}_{\tau\sim\pi,\theta} \left[\nabla_\pi \Phi_\theta(\tau, \pi)\right] \implies \begin{aligned} \nabla^2_\pi J_\theta(\pi) &= \mathop{\mathbb{E}}_{\tau\sim\pi,\theta} \left[\nabla_\pi \Phi_\theta \cdot \nabla_\pi \log p_\theta^\top + \nabla^2_\pi \Phi_\theta\right] \\ \nabla^2_{\theta\pi} J_\theta(\pi) &= \mathop{\mathbb{E}}_{\tau\sim\pi,\theta} \left[\nabla_\pi \Phi_\theta \cdot \nabla_\theta \log p_\theta^\top + \nabla^2_{\theta\pi} \Phi_\theta\right] \end{aligned} \tag{2.9}$$

First part of the proof (policy gradient theorem). The first part of the proof follows the policy gradient theorem. We begin with definitions.

Let $\tau = \{s_1, a_1, s_2, a_2, \cdots, s_h, a_h\}$ be a trajectory sampled according to policy π and MDP parameter θ. Define $\tau_j = \{s_1, a_1, \cdots, s_j, a_j\}$ to be a partial trajectory up to time step j for any $j \in [h]$. Define $G_\theta(\tau) = \sum_{j=1}^{h} \gamma^j R_\theta(s_j, a_j)$ to be the discounted value of trajectory τ. Let $p_\theta(\tau, \pi)$ be the probability of seeing trajectory τ under parameter θ and policy π. Given MDP parameter θ, we can compute the expected cumulative reward of policy π by:

$$J_\theta(\pi) = \mathop{\mathbf{E}}_{\tau\sim\pi,\theta} G_\theta(\tau)$$

$$
\begin{aligned}
&= \mathop{\mathbf{E}}_{\tau\sim\pi,\theta} \sum_{j=1}^{h} \gamma^j R_\theta(s_j, a_j) \\
&= \sum_{j=1}^{h} \mathop{\mathbf{E}}_{\tau\sim p_\theta(\tau,\pi)} \gamma^j R_\theta(s_j, a_j) && \text{(A.1)} \\
&= \sum_{j=1}^{h} \mathop{\mathbf{E}}_{\tau_j\sim p_\theta(\tau_j,\pi)} \gamma^j R_\theta(s_j, a_j) && \text{(A.2)} \\
&= \sum_{j=1}^{h} \int_{\tau_j} \gamma^j R_\theta(s_j, a_j) p_\theta(\tau_j, \pi) d\tau_j
\end{aligned}
$$

Equation A.1 to Equation A.2 uses the fact that we only need to sample up to time step j in order to compute $\gamma^j R_\theta(s_j, a_j)$. Everything beyond time step j does not affect the expectation up to time step j. We can compute the policy gradient by:

$$
\begin{aligned}
\nabla_\pi J_\theta(\pi) &= \nabla_\pi \sum_{j=1}^{h} \int_{\tau_j} \gamma^j R_\theta(s_j, a_j) p_\theta(\tau_j, \pi) d\tau_j \\
&= \sum_{j=1}^{h} \int_{\tau_j} \gamma^j R_\theta(s_j, a_j) \nabla_\pi p_\theta(\tau_j, \pi) d\tau_j && \text{(A.3)} \\
&= \sum_{j=1}^{h} \int_{\tau_j} \gamma^j R_\theta(s_j, a_j) p_\theta(\tau_j, \pi) \nabla_\pi \log p_\theta(\tau_j, \pi) d\tau_j && \text{(A.4)}
\end{aligned}
$$

where Equation A.3 is because only the probability term is dependent on policy π, and Equation A.4 is by $\nabla_\pi p_\theta = p_\theta \nabla_\pi \log p_\theta$.

We can now merge the integral back to an expectation over trajectory τ_j by merging the probability term p_θ and the integral:

$$
\begin{aligned}
\nabla_\pi J_\theta(\pi) &= \sum_{j=1}^{h} \mathop{\mathbf{E}}_{\tau_j\sim p_\theta(\tau_j,\pi)} \left[\gamma^j R_\theta(s_j, a_j) \nabla_\pi \log p_\theta(\tau_j, \pi)\right] \\
&= \sum_{j=1}^{h} \mathop{\mathbf{E}}_{\tau\sim p_\theta(\tau,\pi)} \left[\gamma^j R_\theta(s_j, a_j) \nabla_\pi \log p_\theta(\tau_j, \pi)\right] \\
&= \mathop{\mathbf{E}}_{\tau\sim p_\theta(\tau,\pi)} \left[\sum_{j=1}^{h} \gamma^j R_\theta(s_j, a_j) \nabla_\pi \log p_\theta(\tau_j, \pi)\right]
\end{aligned}
$$

$$= \mathop{\mathbf{E}}_{\tau \sim p_\theta(\tau,\pi)} \left[\sum_{j=1}^{h} \gamma^j R_\theta(s_j, a_j) \nabla_\pi \left(\sum_{i=1}^{j} \log \pi(a_i \mid s_i) + \sum_{i=1}^{j} \log p_\theta(s_i, a_i, s_{i+1}) \right) \right] \quad \text{(A.5)}$$

$$= \mathop{\mathbf{E}}_{\tau \sim p_\theta(\tau,\pi)} \left[\sum_{j=1}^{h} \gamma^j R_\theta(s_j, a_j) \sum_{i=1}^{j} \nabla_\pi \log \pi(a_i \mid s_i) \right]$$

$$= \mathop{\mathbf{E}}_{\tau \sim p_\theta(\tau,\pi)} \left[\sum_{j=1}^{h} \sum_{i=1}^{j} \gamma^j R_\theta(s_j, a_j) \nabla_\pi \log \pi(a_i \mid s_i) \right]$$

$$= \mathop{\mathbf{E}}_{\tau \sim p_\theta(\tau,\pi)} \left[\sum_{i=1}^{h} \sum_{j=i}^{h} \gamma^j R_\theta(s_j, a_j) \nabla_\pi \log \pi(a_i \mid s_i) \right]$$

$$= \mathop{\mathbf{E}}_{\tau \sim p_\theta(\tau,\pi)} \left[\nabla_\pi \Phi_\theta(\tau, \pi) \right] \quad \text{(A.6)}$$

where Equation A.5 is by expanding the probability of seeing trajectory τ_j when parameter θ and policy π are used, where the probability decomposes into the first term action probability $\pi(a_i \mid s_i)$, and the second term transition probability $p_\theta(s_i, a_i, s_{i+1})$, which is independent of policy π and thus disappears. The last equation in Equation A.6 connects back to the definition of Φ as defined in the statement of Theorem 1. Φ is easy to compute and easy to differentiate through. We can therefore sample a set of trajectories $\{\tau\}$ to compute the corresponding Φ and its derivative to get the unbiased policy gradient estimate. □

Second part of the proof (second-order derivatives). Given the policy gradient theorem as we recall in the above derivation, we have:

$$\nabla_\pi J_\theta(\pi) = \mathop{\mathbf{E}}_{\tau \sim p_\theta(\tau,\pi)} \left[\nabla_\pi \Phi_\theta(\tau, \pi) \right] \quad \text{(A.7)}$$

We can compute the derivative of Equation A.7 by:

$$\begin{aligned}
\nabla_\pi^2 J_\theta(\pi) &= \nabla_\pi \nabla_\pi J_\theta(\pi) \\
&= \nabla_\pi \mathop{\mathbf{E}}_{\tau \sim p_\theta(\tau,\pi)} \left[\nabla_\pi \Phi_\theta(\tau, \pi) \right] \\
&= \nabla_\pi \int_\tau \nabla_\pi \Phi_\theta(\tau, \pi) p_\theta(\tau, \pi) d\tau \\
&= \int_\tau \left[\nabla_\pi \Phi_\theta(\tau, \pi) \nabla_\pi p_\theta(\tau, \pi)^\top + \nabla_\pi^2 \Phi_\theta(\tau, \pi) p_\theta(\tau, \pi) \right] d\tau \quad \text{(A.8)} \\
&= \int_\tau \left[\nabla_\pi \Phi_\theta(\tau, \pi) \nabla_\pi \log p_\theta(\tau, \pi)^\top + \nabla_\pi^2 \Phi_\theta(\tau, \pi) \right] p_\theta(\tau, \pi) d\tau \\
&= \mathop{\mathbf{E}}_{\tau \sim p_\theta(\tau,\pi)} \left[\nabla_\pi \Phi_\theta(\tau, \pi) \nabla_\pi \log p_\theta(\tau, \pi)^\top + \nabla_\pi^2 \Phi_\theta(\tau, \pi) \right] \quad \text{(A.9)}
\end{aligned}$$

where Equation A.8 passes gradient inside the integral and applies chain rule. Equation A.9 provides an unbiased estimate of the second-order derivative $\nabla^2_\pi J_\theta(\pi)$.

Similarly, we can compute:

$$\begin{aligned}
\nabla^2_{\theta\pi} J_\theta(\pi) &= \nabla_\theta \nabla_\pi J_\theta(\pi) \\
&= \nabla_\theta \mathop{\mathbf{E}}_{\tau\sim p_\theta(\tau,\pi)} \left[\nabla_\pi \Phi_\theta(\tau,\pi)\right] \\
&= \nabla_\theta \int_\tau \nabla_\pi \Phi_\theta(\tau,\pi) p_\theta(\tau,\pi) d\tau \\
&= \int_\tau \left[\nabla_\pi \Phi_\theta(\tau,\pi) \nabla_\theta p_\theta(\tau,\pi)^\top + \nabla^2_{\theta\pi}\Phi_\theta(\tau,\pi) p_\theta(\tau,\pi)\right] d\tau \\
&= \int_\tau \left[\nabla_\pi \Phi_\theta(\tau,\pi) \nabla_\theta \log p_\theta(\tau,\pi)^\top + \nabla^2_{\theta\pi}\Phi_\theta(\tau,\pi)\right] p_\theta(\tau,\pi) d\tau \\
&= \mathop{\mathbf{E}}_{\tau\sim p_\theta(\tau,\pi)} \left[\nabla_\pi \Phi_\theta(\tau,\pi) \nabla_\theta \log p_\theta(\tau,\pi)^\top + \nabla^2_{\theta\pi}\Phi_\theta(\tau,\pi)\right] \qquad \text{(A.10)}
\end{aligned}$$

Equation A.9 and Equation A.10 both serve as unbiased estimates of the corresponding second-order derivatives. We can sample a set of trajectories to compute both of them and get an unbiased estimate of the second-order derivatives. This concludes the proof of Theorem 1. □

Theorem 2 (Bellman-based unbiased derivative estimate)**.** *We follow the notation in Definition 2 to define* $J_\theta(\pi) = \frac{1}{2} \mathop{\mathbb{E}}_{\tau\sim\pi,\theta} \left[\delta^2_\theta(\tau,\pi)\right]$. *We have:*

$$\begin{aligned}
\nabla_\pi J_\theta(\pi) &= \mathop{\mathbb{E}}_{\tau\sim\pi,\theta} \left[\delta \nabla_\pi \delta + \frac{1}{2}\delta^2 \nabla_\pi \log p_\theta\right] \implies \nabla^2_\pi J_\theta(\pi) = \mathop{\mathbb{E}}_{\tau\sim\pi,\theta} \left[\nabla_\pi \delta \nabla_\pi \delta^\top + O(\delta)\right] \\
\nabla^2_{\theta\pi} J_\theta(\pi) &= \mathop{\mathbb{E}}_{\tau\sim\pi,\theta} \left[\nabla_\pi \delta \nabla_\theta \delta^\top + \left(\nabla_\pi \delta \nabla_\theta \log p_\theta^\top + \nabla_\pi \log p_\theta \nabla_\theta \delta^\top + \nabla^2_{\theta\pi}\delta\right)\delta + O(\delta^2)\right] \qquad (2.10)
\end{aligned}$$

First part of the proof (first-order derivative). By the definition of $J_\theta(\pi) = \frac{1}{2} \mathop{\mathbf{E}}_{\tau\sim\pi,\theta} \left[\delta^2(\tau,\pi)\right]$, we can compute its first-order derivative by:

$$\begin{aligned}
\nabla_\pi J_\theta(\pi) &= \nabla_\pi \frac{1}{2} \mathop{\mathbf{E}}_{\tau\sim\pi,\theta} \left[\delta^2_\theta(\tau,\pi)\right] \\
&= \nabla_\pi \frac{1}{2} \int_\tau \delta^2_\theta(\tau,\pi) p_\theta(\tau,\pi) d\tau \\
&= \int_\tau \left[p_\theta(\tau,\pi)\delta_\theta(\tau,\pi)\nabla_\pi \delta_\theta(\tau,\pi) + \frac{1}{2}\delta^2_\theta(\tau,\pi)\nabla_\pi p_\theta(\tau,\pi)\right] d\tau \\
&= \int_\tau \left[\delta_\theta(\tau,\pi)\nabla_\pi \delta_\theta(\tau,\pi) + \frac{1}{2}\delta^2_\theta(\tau,\pi)\nabla_\pi \log p_\theta(\tau,\pi)\right] p_\theta(\tau,\pi) d\tau
\end{aligned}$$

$$= \mathop{\mathbf{E}}_{\tau\sim\pi,\theta}\left[\delta_\theta(\tau,\pi)\nabla_\pi\delta_\theta(\tau,\pi) + \frac{1}{2}\delta_\theta^2(\tau,\pi)\nabla_\pi \log p_\theta(\tau,\pi)\right] \tag{A.11}$$

□

Second part of the proof (second-order derivative). Given Equation A.11, we can further compute the second-order derivatives by:

$$\begin{aligned}
\nabla_\pi^2 J_\theta(\pi) &= \nabla_\pi \nabla_\pi J_\theta(\pi) \\
&= \nabla_\pi \mathop{\mathbf{E}}_{\tau\sim\pi,\theta}\left[\delta_\theta(\tau,\pi)\nabla_\pi\delta_\theta(\tau,\pi) + \frac{1}{2}\delta_\theta^2(\tau,\pi)\nabla_\pi \log p_\theta(\tau,\pi)\right] \\
&= \nabla_\pi \int_\tau \left[\delta_\theta(\tau,\pi)\nabla_\pi\delta_\theta(\tau,\pi) + \frac{1}{2}\delta_\theta^2(\tau,\pi)\nabla_\pi \log p_\theta(\tau,\pi)\right] p_\theta(\tau,\pi) d\tau \\
&= \int_\tau \left(\nabla_\pi\delta\nabla_\pi\delta^\top + \delta\nabla_\pi^2\delta + \delta\nabla\log p_\theta\nabla_\pi\delta^\top + \frac{1}{2}\delta^2\nabla^2\log p_\theta\right) p_\theta \\
&\qquad + \left(\delta\nabla_\pi\delta(\tau,\pi) + \frac{1}{2}\delta^2\nabla_\pi\log p_\theta\right) p_\theta \nabla\log p_\theta^\top \quad d\tau \\
&= \mathop{\mathbf{E}}_{\tau\sim\pi,\theta}\left[\nabla_\pi\delta\nabla_\pi\delta^\top + \delta\nabla_\pi^2\delta + \delta\nabla\log p_\theta\nabla_\pi\delta^\top + \delta\nabla_\pi\delta(\tau,\pi)\nabla\log p_\theta^\top + O(\delta^2)\right] \\
&= \mathop{\mathbf{E}}_{\tau\sim\pi,\theta}\left[\nabla_\pi\delta\nabla_\pi\delta^\top + O(\delta)\right]
\end{aligned}$$

Similarly, we have:

$$\begin{aligned}
\nabla_{\theta\pi}^2 J_\theta(\pi) &= \nabla_\theta \nabla_\pi J_\theta(\pi) \\
&= \nabla_\theta \mathop{\mathbf{E}}_{\tau\sim\pi,\theta}\left[\delta_\theta(\tau,\pi)\nabla_\pi\delta_\theta(\tau,\pi) + \frac{1}{2}\delta_\theta^2(\tau,\pi)\nabla_\pi \log p_\theta(\tau,\pi)\right] \\
&= \nabla_\theta \int_\tau \left[\delta_\theta\nabla_\pi\delta_\theta + \frac{1}{2}\delta_\theta^2\nabla_\pi\log p_\theta\right] p_\theta d\tau \\
&= \int_\tau \left(\nabla_\pi\delta\nabla_\theta\delta^\top + \delta\nabla_{\theta\pi}^2\delta + \delta\nabla_\pi\log p_\theta\nabla_\theta\delta^\top + \frac{1}{2}\delta^2\nabla_{\theta\pi}^2\log p_\theta\right) p_\theta \\
&\qquad + \left(\delta\nabla_\pi\delta + \frac{1}{2}\delta^2\nabla_\pi\log p_\theta\right) p_\theta \nabla_\theta\log p_\theta^\top \quad d\tau \\
&= \mathop{\mathbf{E}}_{\tau\sim\pi,\theta}\left[\nabla_\pi\delta\nabla_\theta\delta^\top + \delta\nabla_{\theta\pi}^2\delta + \delta\nabla_\pi\log p_\theta\nabla_\theta\delta^\top + \delta\nabla_\pi\delta\nabla_\theta\log p_\theta^\top + O(\delta^2)\right] \\
&= \mathop{\mathbf{E}}_{\tau\sim\pi,\theta}\left[\nabla_\pi\delta\nabla_\theta\delta^\top + \left(\nabla_{\theta\pi}^2\delta + \nabla_\pi\log p_\theta\nabla_\theta\delta^\top + \nabla_\pi\delta\nabla_\theta\log p_\theta^\top\right)\delta + O(\delta^2)\right]
\end{aligned} \tag{A.12}$$

which concludes the proof. □

A.2 Additional Discussions of Decision-focused Learning

In this section, we provide additional discussions of applying decision-focused learning to MDPs problems.

A.2.1 Smoothness of the Optimal Policy Derived From Reinforcement Learning Solver

In Equation 2.8, we compute the gradient of the final evaluation metric with respect to the predictive model by applying chain rule. This implicitly requires each individual component in the chain rule to be well-defined. Specifically, the mapping from the MDP parameters to the optimal policy needs to be smooth so that we can compute a meaningful derivative of the policy with respect to the MDP parameters. However, this smoothness requirement is only required in the training time to make the gradient computation available. Once the training is finished, there is no restriction on the policy and the corresponding solver. This smoothness requirement does not restrict the kind of problems that we can solve. We just need to find a solver that can give a smooth policy to ensure the differentiability at training time, e.g., soft actor critic and soft Q learning.

Specifically, the assumption on smooth policy is similar to the idea of soft Q-learning [127] and soft actor-critic [128] proposed by Haarnoja et al. Soft Q-learning relaxes the Bellman equation to a soft Bellman equation to make the policy smoother, while soft actor-critic adds an entropy term as regularization to make the optimal policy smoother. These relaxed policy not only can make the training smoother as stated in the above papers, but also can allow back-propagation through the optimal policy to the input MDP parameters in our paper. These benefits are all due to the smoothness of the optimal policy. Similar issues arise in decision-focused learning in discrete optimization, with Wilder et al. [338] proposing to relaxing the optimal solution by adding a regularization term, which serves as the same purpose as we relax our optimal policy in the sequential decision problem setting.

A.2.2 Unbiased Second-order Derivative Estimates

As we discuss in Section 2.7, correctly approximating the second-order derivatives is the crux of our algorithm. Incorrect approximation may lead to incorrect gradient direction, which can further lead to divergence. Since the second-order derivative formulation as stated in Theorem 1 and Theorem 2 are both unbiased derivative estimate. However, their accuracy depends on how many samples we use to approximate the derivatives. In our experiments, we use 100 sampled trajectories to approximate the second-order derivatives across three domains. The number of samples required to get a sufficiently accurate derivative estimate may depend on the problem size. Larger problems may require more samples to get a good derivative estimate, but more samples also implies more computation cost required to run the back-propagation.

In practice, we find that normalization effect of the Hessian term as discussed in Section 2.7 is very important to reduce the variance caused by the incorrect derivative estimate. Additionally, we also notice that adding a small additional predictive loss term to run back-propagation can stabilize

the training process because the predictive loss does not suffer from sampling variance. This is why we add a weighted predictive loss to the back-propagation in Algorithm 1.

A.2.3 Impact of Optimality in the Forward Pass

In order to differentiate through the KKT conditions, we need the policy π^* return by the reinforcement learning solver to be optimal in Figure 2.1. However, sub-optimal solution is often reached by the reinforcement learning solver and the optimality can impact the gradient computed from differentiating through the KKT conditions.

In this section, we analyze the impact of a sub-optimal policy produced by the reinforcement learning solver. When the problem is smooth, or more precisely when the function $J_\theta(\pi)$ is smooth around the optimal policy π^*, we can bound the gradients $\nabla^2_\pi J_\theta(\pi')$ and $\nabla_\pi J_\theta(\pi')$ computed in Equation 2.8 using a sub-optimal policy π' by the gradients computed using the optimal policy π^*. Specifically, if the Hessian $\nabla^2_\pi J_\theta(\pi^*)$ is sufficiently far from singular, the difference between two gradients computed from sub-optimal and optimal policy using Equation 2.8 can be written as:

$$\left| \frac{d\,\mathrm{Eval}(\pi')}{d\pi} (\nabla^2_\pi J_\theta(\pi'))^{-1} \nabla^2_{\theta\pi} J_\theta(\pi') \frac{d\theta}{dw} - \frac{d\,\mathrm{Eval}(\pi^*)}{d\pi} (\nabla^2_\pi J_\theta(\pi^*))^{-1} \nabla^2_{\theta\pi} J_\theta(\pi^*) \frac{d\theta}{dw} \right|$$

which can be further bounded by applying telescoping sum to decompose the difference into linear combination of the difference in each individual gradient term. This suggests that when the smoothness condition of the above derivatives is met, we can bound the error incurred by sub-optimal policy.

A.3 Experimental Setup

In this section, we describe how we randomly generate the MDP problems and the corresponding missing parameters.

Feature generation Across all three domains, once the missing parameters are generated, we feed each MDP parameter into a randomly initialized neural network with two intermediate layers each with 64 neurons, and an output dimension size 16 to generate a feature vector of size 16 for the corresponding MDP parameter. For example, in the gridworld example, each grid cell comes with a missing reward. So the feature corresponding to this grid cell and the missing reward is generated by feeding the missing reward into a randomly initialized neural network to generate a feature vector of size 16 for this particular grid cell. We repeat the same process for all the parameters in the MDP problem, e.g., all the grid cells in the gridworld problem. The randomly initiated neural network uses ReLU layers as nonlinearity followed by a linear layer in the end. The generated features are normalized to mean 0 and variance 1, and we add Gaussian noise $\mathcal{N}(0, 1)$ to the features, with a signal noise ratio is $1 : 3$, to model that the original missing parameters may not be perfectly recovered

from the noisy features. The predictive model we use to map from generated noisy features to the missing parameters is a single layer neural network with 16 neurons.

Training parameters Across all three examples, we consider the discounted setting where the discount factor is $\gamma = 0.95$. The learning rate is set to be $\alpha = 0.01$. The number of demonstrated trajectories is set to be 100 in both the random and near-optimal settings.

Reinforcement learning solvers In order to train the optimal policy, in the gridworld example, we use tabular value-iteration algorithm to learn the Q value of each state action pair. In the snare finding and the TB problems, since the state space is continuous, we apply DDQN[262,309] to train the Q function and the corresponding policy, where we use a neural network with two intermediate layers each with 64 neurons to represent the function approximators of the Q values. There is one exception in the runtime plot of the snare finding problem in Figure 2.3(c), where the full Hessian computation is infeasible when a two layer neural network is used. Thus we use an one layer neural network with 64 neurons only to test the runtime of different Hessian approximations.

A.3.1 Gridworld Example With Missing Rewards

Problem setup We consider a 5×5 Gridworld environment with unknown rewards as our MDP problems with unknown parameters. The bottom left corner is the starting point and the top right corner is a safe state with a high reward drawn from a normal distribution $\mathcal{N}(5, 1)$. The agent can walk between grid cells by going north, south, east, west, or deciding to stay in the current grid cells. So the number of available actions is 5, while the number of available states is $5 \times 5 = 25$.

The agent collects reward when the agent steps on each individual grid cell. There is 20% chance that each intermediate grid cell is a cliff that gives a high penalty drawn from another normal distribution $\mathcal{N}(-10, 1)$. All the other 80% of grid cells give rewards drawn from $\mathcal{N}(0, 1)$. The goal of the agent is to collect as much reward as possible. We consider a fixed time horizon case with 20 steps, which is sufficient for the agent to go from bottom left to the top right corner.

Training details Within each individual training step for each MDP problem with missing parameters, we first predict the rewards using the predictive model, and then solve the resulting problem using tabular value-iteration. We run in total 10000 iterations to learn the Q values, which are later used to construct the optimal policy. To relax the optimal policy given by the RL solver, we relax the Bellman equation used to run value-iteration by relaxing all the argmax and max operators in the Bellman equation to softmax with temperature 0.1, i.e., we use $\text{SOFTMAX}(0.1 \cdot \text{Q-values})$ to replace all the argmax over Q values. The choice of the tempreratue 0.1 is to make sure that the optimal policy is smooth enough but the relaxation does not impact the optimal policy too much as well.

Random and near-optimal trajectories generation To generate the random trajectories, we have the agent randomly select actions between all actions. To generate the near-optimal trajectories, we replace the softmax with temperature 0.1 by softmax with temperature 1 and train an RL agent using ground truth reward values by 50000 value-iterations to get a near-optimal policy. We then use the trained near-optimal policy to generate 100 independent trajectories as our near-optimal demonstrated trajectories.

A.3.2 Snare Finding Problem With Missing Transition Probability

Problem setup In the snare finding problem, we consider a set of 20 sites that are vulnerable to poaching activity. We randomly select 20% of the sites as high-risk locations where the probability of having a poacher coming and placing a snare is randomly drawn from a normal distribution $\mathcal{N}(0.8, 0.1)$, while the remaining 80% of low-risk sites with probability $\mathcal{N}(0.1, 0.05)$ having a poacher coming to place a snare. These transition probabilities are not known to the ranger, and the ranger has to rely on features of each individual site to predict the corresponding missing transition probability.

We assume the maximum number of snare is 1 per location, meaning that if there is a snare and it has not been removed by the ranger, then the poacher will not place an additional snare there until the snare is removed. The ranger only observes a snare when it is removed. Thus the MDP problem with given parameters is partially observable, where the state maintained by the ranger is the belief of whether a site contains a snare or not, which is a fractional value between 0 and 1 for each site.

The available actions for the ranger are to select a site from 20 sites to visit. If there is a snare in the location, the ranger successfully removes the snare and gets reward 1 with probability 0.9, and otherwise the snare remains there with a reward -1. If there is no snare in the visited site, the ranger gets reward -1. Thus the number of actions to the ranger is 20, while the state space is continuous since the ranger uses continuous belief as the state.

Training details To solve the optimal policy from the predicted parameters, we run DDQN with 1000 iterations to collect random experience and 10000 iterations to train the model. We use a replay buffer to store all the past experience that the agent executed before. To soften the optimal policy, we also use a relaxed Bellman equation as stated in Section A.3.2. Because the cumulative reward and the corresponding Q values in this domain is relatively smaller than the Gridworld domain, we replace all the argmax and max operators by softmax with a larger temperature 1 to reflect the relatively smaller reward values.

Random and near-optimal trajectories generation Similar to Section A.3.1, we generate the random trajectories by having the agent choose action from all available actions uniformly at random. To generate the near-optimal trajectories, we replace all the softmax with temperature 1 by softmax with temperature 5, and we use the ground truth transition probabilities to train the

RL agent by DDQN using 50000 iterations to generate a near-optimal policy. The near-optimal trajectories are then generated by running the trained near-optimal policy for 100 independent runs.

A.3.3 Tuberculosis With Missing Transition Probability

Problem setup There are a total of 5 patients who need to take their medication at each time-step for 30 time-steps. For each patient, there are 2 states – non-adhering (0), and adhering (1). The patients are assumed to start from a non-adhering state. Then, in subsequent time-steps, the patients' states evolve based on their current state and whether they were intervened on by a healthcare worker according to a transition matrix.

The raw transition probabilities for different patients are taken from [170].* However, these raw probabilities do not record a patient's responsiveness to an intervention. To incorporate the effect of intervening, we sample numbers from $U(0, 0.4)$, and (a) add them to the probability of adhering when intervened on, and (b) subtract them from the probability of adhering when not. Finally, we clip the probabilities to the range of $[0.05, 0.95]$ and re-normalize. We use the raw transition probabilities and the randomly generated intervention effect to model the behavior of our synthetic patients and generate all the training trajectories accordingly. The entire transition matrix for each patient is then fed as an input to the feature generation network to generate features for that patient. In this example, we assume the transition matrices to be missing parameters, and try to learn a predictive model to recover the transition matrices from the generated features using either two-stage or various decision-focused learning methods as discussed in the main paper.

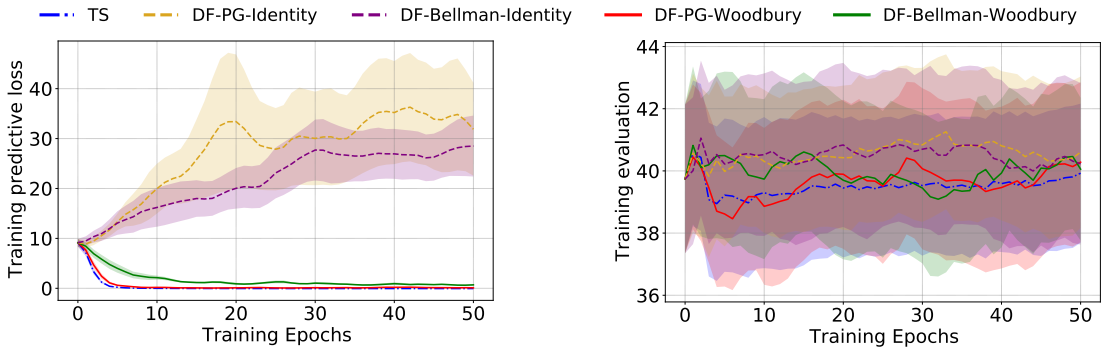
Given the synthetic patients, we consider a healthcare worker who has to choose one patient at every time-step to intervene on. However, the healthcare worker can only observe the 'true state' of a patient when she intervenes on them. At every other time, she has a 'belief' of the patient's state that is constructed from the most recent observation and the predicted transition probabilities. Therefore, the healthcare worker has to learn a policy that maps from these belief states to the action of whom to intervene on, such that the sum of adherences of all patients is maximised over time. The healthcare worker gets a reward of 1 for an adhering patient and 0 for a non-adhering one. Like Problem A.3.2, this problem has a continuous state space (because of the belief states) and discrete action space.

Training details Same as Section A.3.2.

Random and near-optimal trajectories generation Similar to Section A.3.2, we generate the random trajectories by having the agent choose action from all available actions uniformly at random. To generate the near-optimal trajectories, we replace all the softmax with temperature 5 by softmax with temperature 20,† and we use the ground truth transition probabilities to train the

*The raw transition probabilities taken from [170] are only used to generate synthetic patients.

†The reason that we use a relatively larger temperature is because the range of the cumulative reward in TB domain is smaller than the previous two domains. In TB domain, the patients could change from



(a) Predictive loss on training MDPs

(b) Performance on training MDPs

Figure A.1: Learning curves of for the TB problem with random trajectories.

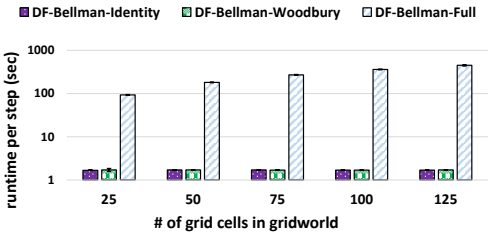
RL agent by DDQN using 100, 000 iterations to generate a near-optimal policy. The near-optimal trajectories are then generated by running the trained near-optimal policy for 100 independent runs.

A.4 Additional Experiment Results

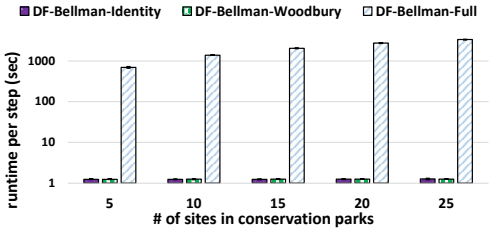
Tuberculosis Adherence The results for this problem can be found in Table 2.1, and the training curves can be found in Figure A.1. While the standard errors associated with the results *seem* very large, this is in large part because of the way in which we report them. To keep it consistent with other problems, we average the absolute OPE scores for each method across multiple problem instances. However, in the TB case, each problem instance can be very different because the patients in each of these instances are sampled from the transition probabilities previously studied in [170] that have diverse patient behaviour. As a result, the baseline OPE values vary widely across different problem instances, causing a larger variation in Figure A.1(b) and contributing as the major source of standard deviation.

Computation cost of Bellman equation-based decision-focused methods We additionally compare the runtime of the operation of backpropagation per gradient step of Bellman equation-based decision-focused learning using different Hessian approximations. This is the runtime required to compute the gradient in the backward pass. We can see that the runtime of methods using identity and Woodbury methods are much smaller than the runtime of full Hes-

non-adhering back to adhering even if there is no intervention, while in contrast, a snare placed in a certain location will not be removed until the ranger visits the place. In other words, the improvement that intervention can introduce is relatively limited compared to the snare finding domain. Thus even though the cumulative reward in TB domain is larger than the previous two domains, the range is smaller and thus we need a larger temperature.



(a) Backpropagation runtime per gradient step of Bellman equation-based decision-focused learning using different Hessian approximations in the gridworld problem.



(b) Backpropagation runtime per gradient step of Bellman equation-based decision-focused learning using different Hessian approximations in the snare finding problem.

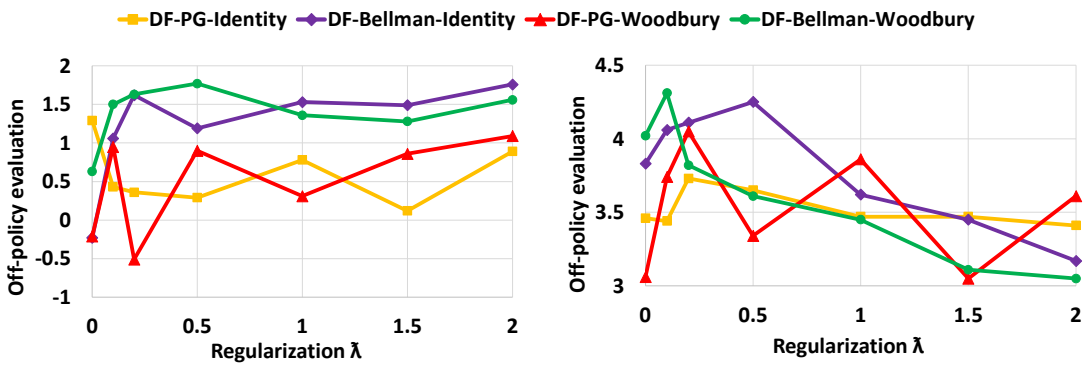
Figure A.2: We compare the backpropagation runtime of decision-focused learning methods using bellman optimality with different Hessian approximations. We can see that the runtime of both identity and Woodbury methods largely outperform the runtime of full Hessian computation, demonstrating the importance of the Hessian approximation. Additionally, the runtime of Woodbury method using low-rank approximation is similar to the runtime of identity method. Woodbury method provides a more accurate approximation with a similar runtime.

sian approximation. This matches to our analysis in Section 2.7 and the experimental results in Figure 2.2(c) and Figure 2.3(c).

Choice of regularizer λ in Algorithm 1 and ablation study We ran an ablation study by varying the regularization constant λ in Algorithm 1 using the snare-finding problem. The experimental result is shown in Figure A.3. The role of regularization in Algorithm 1 is to help resolve the non-convexity issue of the off-policy evaluation (OPE) objective. Decision-focused learning methods can easily get trapped by various local minima due to the non-convexity of the OPE metric. Adding a small two-stage loss can improve the convexity of the optimizing objective and thus help improve the performance as well. We can see that adding small amount of regularization can usually help improve the overall performance in both cases with random and near-optimal trajectories. However, adding too much regularization in Algorithm 1 can push decision-focused learning toward two-stage approach, which can degrade the performance sometimes. The right amount of regularization is critical to balance between the issue of convexity and the optimizing objective.

A.5 Computing Infrastructure

All experiments except were run on a computing cluster, where each node is configured with 2 Intel Xeon Cascade Lake CPUs, 184 GB of RAM, and 70 GB of local scratch space. Within each experiment, we did not implement parallelization nor use GPU, so each experiment was purely run on a single CPU core.



(a) The off-policy evaluation performance of different regularization λ in snare-finding problem with random trajectories.

(b) The off-policy evaluation performance of different regularization λ in snare-finding problem with near-optimal trajectories.

Figure A.3: Ablation study of different regularization λ in Algorithm 1 on the snare-finding problem using different decision-focused learning methods.

B
Appendix to Chapter 3

B.1 Hyperparameter Setting and Computation Infrastructure

We run both Decision Focused Learning and Two-Stage Learning for 50 epochs in 2-state and 5-state synthetic domain problems, 30 epochs in ARMMAN domain and 18 epochs in 2-state partially observable setting. The learning rate r is kept at 0.01 and $\gamma = 0.59$ is used in all experiments. All the experiments are performed on an Intel Xeon CPU with 64 cores and 128 GB memory.

Neural Network Structure

The predictive model m_w we use to predict the transition probability is a neural network with an intermediate layer of size 64 with ReLU activation function, and an output layer of size of the transition probability followed by a softmax layer to match probability distribution. Dropout layers are added to avoid overfitting. The same neural network structure is applied to all domains and all training methods.

In the synthetic datasets, given the generated transition probabilities, we feed the transition probability of each arm into a randomly initialized neural network with two intermediate layers each with 64 neurons, and an output dimension size 16 to generate a feature vector of size 16. The randomly initiated neural network uses ReLU layers as nonlinearity followed by a linear layer in the end.

B.2 Real ARMMAN Dataset

The large-scale quality improvement study conducted by ARMMAN [20] contains 7668 beneficiries in the Round Robin Group. Over a duration of 7 weeks, 20% of the beneficiaries receive at least one active action (LIVE service call). We randomly split the 7668 beneficiaries into 12 groups while preserving the proportion of beneficiaries who received at least one active action. There are 43 features available for every beneficiary which describe characteristics such as age, income, education level, call slot preference, language preference, phone ownership etc.

B.2.1 Protected and Sensitive Features

ARMMAN's mobile voice call program has long been working with socially disadvantaged populations. ARMMAN does not collect or include constitutionally protected and particularly sensitive categories such as caste and religion. Despite such categories not being available, in pursuit of ensuring fairness, we worked with public health and field experts to ensure indicators such as education, and income levels that signify markers of socio-economic marginalization were measured and evaluated for fairness testing.

B.2.2 Feature List

We provide the full list of 43 features used for predicting transition probability:

- Enroll gestation age, age (split into 5 categories), income (8 categories), education level (7 categories), language (5 categories), phone ownership (3 categories), call slot preference (5 categories), enrollment channel (3 categories), stage of pregnancy, days since first call, gravidity, parity, stillbirths, live births

B.2.3 Feature Evaluation

Feature	Two-stage	Decision-focused learning	p-value
age (year)	25.57	24.9	0.06
gestation age (week)	24.28	17.21	0.00

Table B.1: Feature analysis of continuous features. This table summarizes the average feature values of the beneficiaries selected to schedule service calls by different learning methods. The p-value of the continuous features is analyzed using t-test for difference in mean.

In our simulation, we further analyze the demographic features of participants who are selected to schedule service calls by either two-stage learning method and decision-focused learning method. The following tables show the average value of each individual feature over the selected participants with scheduled service calls under the two-stage or decision-focused learning method. The p-value

Feature	Two-stage	DFL	p-value
income (rupee, averaged over multiple categories)	10560.0	11190.0	0.20
education (categorical)	3.32	3.16	0.21
stage of pregnancy	0.13	0.03	**0.00**
language			
language (hindi)	0.53	0.6	**0.04**
language (marathi)	0.45	0.4	0.08
phone ownership			
phone ownership (women)	0.86	0.82	**0.04**
phone ownership (husband)	0.12	0.16	**0.03**
phone ownership (family)	0.02	0.02	1.00
enrollment channel			
channel type (community)	0.7	0.47	**0.00**
channel type (hospital)	0.3	0.53	**0.00**

Table B.2: Feature analysis of categorical features. This table summarizes the average feature values of the beneficiaries selected to schedule service calls by different learning methods. The p-value of the categorical values is analyzed using chi-square test for different proportions.

of the continuous features is analyzed using t-test for difference in mean; the p-value of the categorical values is analyzed using chi-square test for different proportions.

In Table B.1 and Table B.2, we can see that there is no statistical significance (p-value > 0.05) between the average feature values of income and education, meaning that there is no obvious difference in these feature values between the population selected by two different methods. We see statistical significance in some other features, e.g., gestation age, stage of maternal event, language, phone ownership, and channel type, which may be further analyzed to understand the benefit of decision-focused learning, but they do not appear to directly bear upon socio-economic marginalization; these features are more related to the health status of the beneficiaries.

B.3 Consent for Data Collection and Analysis

In this section, we provide information about consent related to data collection, analyzing data, data usage and sharing.

B.3.1 Secondary Analysis and Data Usage

This study falls into the category of secondary analysis of the aforementioned dataset. We use the previously collected engagement trajectories of different beneficiaries participating in the service call program to train the predictive model and evaluate the performance. The evaluation of the proposed algorithm is evaluated via different off-policy policy evaluations, including an importance

sampling-based method and a simulation-based method discussed in Section 3.6. This paper does not involve deployment of the proposed algorithm or any other baselines to the service call program.As noted earlier, the experiments are secondary analysis using different evaluation metrics with approval from the ARMMAN ethics board.

B.3.2 Consent for Data Collection and Sharing

The consent for collecting data is obtained from each of the participants of the service call program. The data collection process is carefully explained to the participants to seek their consent before collecting the data. The data is anonymized before sharing with us to ensure anonymity. Data exchange and use was regulated through clearly defined exchange protocols including anonymization, read-access only to researchers, restricted use of the data for research purposes only, and approval by ARMMAN's ethics review committee.

B.3.3 Universal Accessibility of Health Information

To allay further concerns: this simulation study focuses on improving quality of service calls. Even in the intended future application, all participants will receive the same weekly health information by automated message regardless of whether they are scheduled to receive service calls or not. The service call program does not withhold any information from the participants nor conduct any experimentation on the health information. The health information is always available to all participants, and participants can always request service calls via a free missed call service. In the intended future application our algorithm may only help schedule *additional* service calls to help beneficiaries who are likely to drop out of the program.

B.4 Societal Impacts and Limitations

B.4.1 Societal Impacts

The improvement shown in the real dataset directly reflects the number of engagements improved by our algorithm under different evaluation metrics. On the other hand, because of the use of demographic features to predict the engagement behavior, we must carefully compare the models learned by standard two-stage approach and our decision-focused learning to further examine whether there is any bias or discrimination concern.

Specifically, the data is collected by ARMMAN, an India non-government organization, to help mothers during their pregnancy. The ARMMAN dataset we use in the paper does not contain information related to race, religion, caste or other sensitive features; this information is not available to the machine learning algorithm. Furthermore, examination by ARMMAN staff of the mothers selected for service calls by our algorithm did not reveal any specific bias related to these features. In particular, the program run by ARMMAN targets mothers in economically disadvantaged communities; the majority of the participants (94%) are below the international poverty line determined

by The World Bank[342]. To compare the models learned by two-stage and DF-Whittle approach, we further examine the difference between those mothers who are selected for service call in two-stage and DF-Whittle, respectively. We observe that there are some interesting differences. For example, DF-Whittle chooses to do service calls to expectant mothers earlier in gestational age (22% vs 37%), and to a lower proportion of those who have already given birth (2.8% vs 13%) compared to two-stage, but in terms of the income level, 94% of the mothers selected by both methods are below the poverty line. This suggests that our approach is not biased based on income level, especially when the entire population is coming from economically disadvantaged communities. Our model can identify other features of mothers who are actually in need of service calls.

B.4.2 Limitations

Impact of limited data and the strength of decision-focused learning As shown in Section 3.8 and Figure 3.4, we notice a smaller improvement between decision-focused learning and two-stage approach when there is sufficient data available in the training set. This is because the data is sufficient enough to train a predictive model with small predictive loss, which implies that the predicted transition probabilities and the true transition probabilities are also close enough with similar Whittle indices and Whittle index policy. In this case with sufficient data, there is less discrepancy between predictive loss and the evaluation metrics, which suggests less improvement led by fixing the discrepancy using decision-focused learning. Compared to two-stage approach, decision-focused learning is still more expensive to run. Therefore, when data is sufficient, two-stage may be sufficient to achieve comparable performance while maintaining a low training cost.

On the other hand, we notice a larger improvement between decision-focused learning and two-stage approach when data is limited. When data is limited, predictive loss is less representative with a larger mismatch compared to the evaluation metrics. Therefore, fixing the objective mismatch issue using decision-focused learning becomes more prominent. Therefore, decision-focused learning may be adopted in the limited data case to significantly improve the performance.

Computation cost As we have shown in Section 3.5.5, our approach improves the computation cost of decision-focused learning from $O(M^{\omega N})$ to $O(NM^{\omega+1})$, where N is the number of arms and M is the number of states. This computation cost is linear in the number of arms N, allowing us to scale up to large real-world deployment of RMAB applications with larger number of arms involved in the problem. Nonetheless, the extension in terms of the number of states M is not cheap. The computation cost still grows between cubic and biquadratic as shown in Figure B.1. This is particularly significant when working on partially observable RMAB problems, where the partially observable problems are reduced to fully observable problems with larger number of states. There is room for improving the computation cost in terms of the number of states to make decision-focused learning more scalable to real-world applications.

B.5 Computation Cost Analysis of Decision-focused Learning

We have shown the computation cost of backpropagating through Whittle indices in Section 3.5.5. This section covers the remaining computation cost associated to other components, including the computation cost of Whittle indices in the forward pass, and the computation cost of constructing soft Whittle index policy using soft-top-k operator.

B.5.1 Solving Whittle Index (Forward Pass)

In this section, we discuss the cost of computing Whittle index in the forward pass. In the work by Qian et al. [258], they propose to use value iteration and binary search to solve the Bellman equation with M states. Therefore, every value iteration requires updating the current value functions of M states by considering all the possible M^2 transitions between states, which results in a computation cost of $O(M^2)$ per value iteration. The value iteration is run for a constant number of iterations, and the binary search is run for $O(\log \frac{1}{\varepsilon})$ iterations to get a precision of order ε. In total, the computation cost is of order $O(M^2 \log \frac{1}{\varepsilon}) = O(M^2)$ where we simply use a fixed precision to ignore the dependency on ε.

On the other hand, there is a faster way to compute the value function by solving linear program with M variables directly. The Bellman equation can be expressed as a linear program where all the M variables are the value functions. The best known complexity of solving a linear program with M variables is $O(M^{2+\frac{1}{18}})$ by Jiang et al. [152]. Notice that this complexity is slightly larger than the one in value iteration because (i) value iteration does not guarantee convergence in a constant iterations (ii) the constant associated to the number of value iterations is large.

In total, we need to compute the Whittle index of N arms and for M possible states in $\mathcal{S}$. The total complexity of value iteration and linear program are $O(NM^3)$ with a large constant and $O(NM^{3+\frac{1}{18}})$, respectively. In any cases, the cost of computing all Whittle indices in the forward pass is still smaller than $O(NM^{1+\omega})$, the cost of backpropagating through all the Whittle indices in the backward pass. Therefore, the backward pass is the bottleneck of the entire process.

B.5.2 Soft-top-k Operators

In Section B.5.1 and Section 3.5.5, we analyze the cost of computing and backpropagating through Whittle indices of all states and all arms. In this section, we discuss the cost of computing the soft Whittle index policy from the given Whittle indices using soft-top-k operators.

Soft-top-k operators Xie et al. [346] reduces top-k selection problem to an optimal transport problem that transports a uniform distribution across all input elements with size N to a distribution where the elements with the highest-k values are assigned probability 1 and all the others are assigned 0.

This optimal transport problem with N elements can be efficiently solved by using Bregman projections [41] with complexity $O(LN)$, where L is the number of iterations used to run Bregman

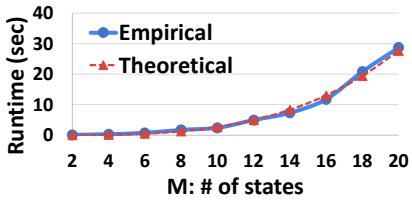


Figure B.1: Computation cost comparison of decision-focused learning in restless multi-armed bandits to the theoretical guarantee with varying number of states M.

projections. In the backward pass, Xie et al. [346] shows that the technique of differentiating through the fixed point equation [32,11] also applies, but the naive implementation requires computation cost $O(N^2)$. Therefore, Xie et al. [346] provides a faster computation approach by leveraging the associate rule in matrix multiplication to lower the backward complexity to $O(N)$.

In summary, a single soft-top-k operator requires $O(LN)$ to compute the result in the forward pass, and $O(N)$ to compute the derivative in the backward pass. In our case, we need to apply one soft-top-k operator for every time step in T and for every trajectory in $\mathcal{T}$. Therefore, the total computation cost of computing a soft Whittle index policy and the associated importance sampling-based evaluation metric is bounded by $O(LNT|\mathcal{T}|)$, which is linear in the number of arms N, but still significantly smaller than $O(NM^{\omega+1})$, the cost of backpropagating through all Whittle indices as shown in Section 3.5.5. Therefore, we just need to concern the computation cost of Whittle indices in decision-focused learning.

B.5.3 Computation Cost Dependency on the Number of States

Figure B.1 compares the computation cost of our algorithm, DF-Whittle, and the theoretical computation cost $O(NM^{\omega+1}$. We vary the number of states M in Figure B.1 and we can see that the computation cost of our algorithm matches the theoretical guarantee on the computation cost. In contrast to the prior work with computation cost $O(M^{\omega N})$, our algorithm significantly improves the computation cost of running decision-focused learning on RMAB problems.

B.6 Importance Sampling-based Evaluations for ARMMAN Dataset with Single Trajectory

Unlike the synthetic datasets that we can produce multiple trajectories of an RMAB problem, in the real problem of service call scheduling problem operated by ARMMAN, there is only one trajectory available to us for every RMAB problem. Due to the specialty of the maternal and child health domain, it is unlikely to have the exactly same set of the pregnant mothers participating in the service call scheduling program at different times and under the same engagement behavior.

Given this restriction, we must evaluate the performance of a newly proposed policy using the only available trajectory. Unfortunately, the standard CWPDIS in Equation 3.12 does not work

because the CWPDIS estimator is canceled out when there is only one trajectory:

$$\text{Eval}_{\text{IS}}(\pi, \mathcal{T}) = \sum_{t\in[T], i\in[N]} \gamma^{t-1} \frac{\mathbb{E}_{\tau\sim\mathcal{T}}\left[r_{t,i}\rho_{ti}(\tau)\right]}{\mathbb{E}_{\tau\sim\mathcal{T}}\left[\rho_{ti}(\tau)\right]} = \sum_{t\in[T], i\in[N]} \gamma^{t-1} \frac{r_{t,i}\rho_{ti}(\tau)}{\rho_{ti}(\tau)} = \sum_{t\in[T], i\in[N]} \gamma^{t-1} r_{t,i} \tag{B.1}$$

which is fixed regardless what target policy π is used and the associated importance sampling weights $\frac{\pi(a_{t,i}|s_t)}{\pi_{\text{beh}}(a_{t,i}|s_t)}$ and $\rho_{ti} = \prod_{t'=1}^{t} \frac{\pi(a_{t',i}|s_{t'})}{\pi_{\text{beh}}(a_{t',i}|s_{t'})}$. This implies that we cannot use CWPDIS to evaluate the target policy when there is only one trajectory.

Accordingly, we use the following variant to evaluate the performance:

$$\text{Eval}_{\text{IS}}(\pi, \mathcal{T}) = \sum_{i\in[N], t\in[T]} \gamma^{t-1} \frac{r_{t,i}\rho'_{ti}(\tau)}{\mathbb{E}_{t'\in[T]}\left[\rho'_{t'i}(\tau)\right]} \tag{B.2}$$

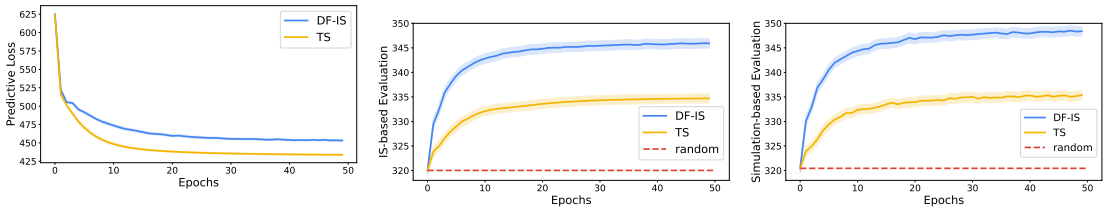
where the new importance sampling weights are defined by $\rho'_{t,i}(\tau) = \frac{\pi(a_{t,i}|s_t)}{\pi_{\text{beh}}(a_{t,i}|s_t)}$, which is not multiplicative compared to the original ones.

The main motivation of this new evaluation metric is to segment the given trajectory into a set of length-1 trajectories. We can apply CWPDIS to the newly generated length-1 trajectories to compute a meaningful estimate because we have more than one trajectory now. The OPE formulation with segmentation is under the assumption that we can decompose the total reward into the contribution of multiple segments using the idea of trajectory segmentation [180,263]. This assumption holds when all segments start with the same state distribution. In our ARMMAN dataset, the data is composed of trajectories of the participants who have enrolled in the system a few weeks ago, which have (almost) reached a stationary distribution. Therefore, the state distribution under the behavior policy, which is a uniform random policy, does not change over time. Our assumption of identical distribution is satisfied and we can decompose the trajectories into smaller segments to perform evaluation. Empirically, we noticed that this temporal decomposition helps define a meaningful importance sampling-based evaluation with the consistency benefit brought by CWPDIS.

B.7 Additional Experimental Results

We provide the learning curves of fully observable 2-state RMAB, fully observable 5-state RMAB, partially observable 2-state RMAB, and the real ARMMAN fully observable 2-state RMAB problems in Figure B.2, Figure B.3, Figure B.4, and Figure B.5, respectively. Across all domains, two-stage method consistently converges to a lower predictive loss faster than decision-focused learning in Figure B.2(a), Figure B.3(a), Figure B.4(a), and Figure B.5(a). However, the learned model does not produce a policy with good performance in the importance sampling-based evaluation metric in Figure B.2(b), Figure B.3(b), Figure B.4(b), and Figure B.5(b), and similarly in the simulation-based

evaluation metric in Figure B.2(c), Figure B.3(c), Figure B.4(c), and Figure B.5(c).

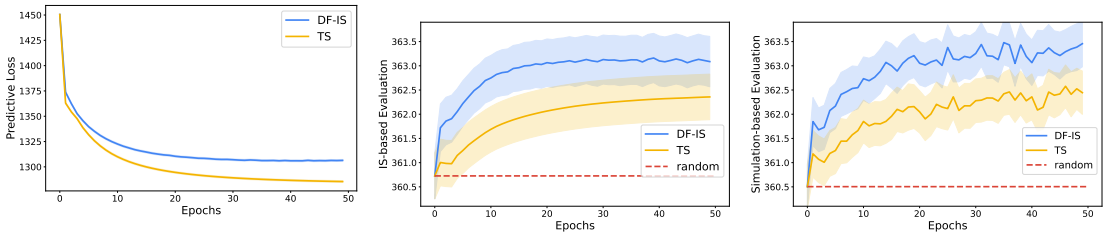


(a) Testing predictive loss v.s. training epoch

(b) Testing IS-based evaluation

(c) Testing simulation-based evaluation

Figure B.2: Comparison between two-stage and decision-focused in the synthetic fully observable 2-state RMAB problems.



(a) Testing predictive loss v.s. training epoch

(b) Testing IS-based evaluation

(c) Testing simulation-based evaluation

Figure B.3: Comparison between two-stage and decision-focused learning for fully observable 5-state RMAB problems.

B.8 Solving for and Differentiating Through the Whittle Index Computation

To solve for the Whittle index for some state $u \in S$, you have to solve the following set of equations:

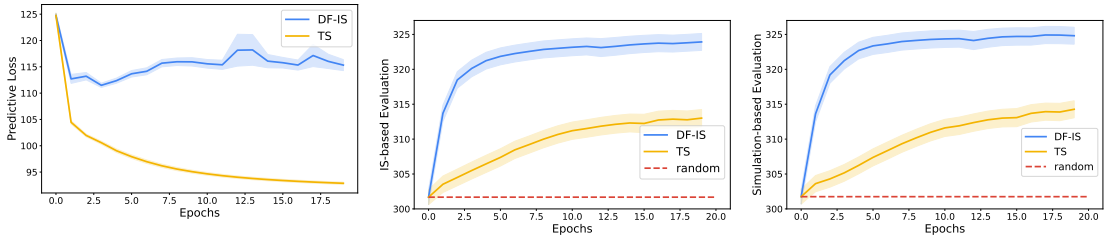
$$V(u) = R(s) + \beta_u + \gamma \sum_{s' \in \mathcal{S}} P(s, 0, s') \cdot V(s')$$

$$V(u) = R(s) + \gamma \sum_{s' \in \mathcal{S}} P(s, 1, s') \cdot V(s') \tag{B.3}$$

$$V(s) = \max_{a \in \{0,1\}} [R(s) + (1-a)\beta_u + \gamma \sum_{s' \in \mathcal{S}} P(s, a, s') \cdot V(s')], \qquad \forall s \in \mathcal{S} - u \tag{B.4}$$

Here:

$\mathcal{S}$ is the set of all states

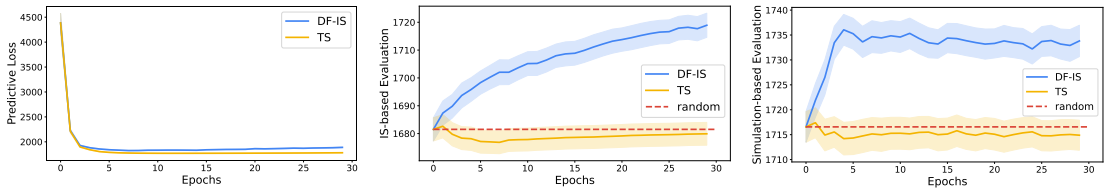


(a) Testing predictive loss v.s. training epoch

(b) Testing IS-based evaluation

(c) Testing simulation-based evaluation

Figure B.4: Comparison between two-stage and decision-focused learning for 2-state partially observable RMAB problems.



(a) Testing predictive loss

(b) Testing IS-based evaluation

(c) Testing simulation-based evaluation

Figure B.5: Comparison between two-stage and decision-focused learning in the real ARMMAN service call scheduling problem. The pulling action in the real dataset is much sparser, leading to a larger mismatch between predictive loss and evaluation metrics. Two-stage overfits to the predictive loss drastically with no improvement in evaluation metrics. In contrast, decision-focused learning can directly optimize the evaluation metric to avoid the objective mismatch issue.

$R(s)$	is the reward for being in state s
$P(s, a, s')$	is the probability of transitioning to state s' when you begin in state s and take action a
$V(s)$	is the expected value of being in state s
β_s	is the whittle index for state s

One way to interpret these equations is to view them as the Bellman Optimality Equations associated with a modified MDP in which the reward function is changed to $R'_u(s, a) = R(s) + (1 - a)\beta_u$, i.e., you are given a 'subsidy' for not acting (Equation B.4). Then, to find the whittle index for state u, you have to find the minimum subsidy for which the value of not acting exceeds the value of acting[337]. At this transition point, the value of not acting is equal to the value of acting in that state (Equation B.3), leading to the set of equations above.

Now, this set of equations is typically hard to solve because of the max terms in Equation B.4. Specifically, knowing whether $\arg\max_a = 0$ or $\arg\max_a = 1$ for some state s is equivalent to knowing what the optimal policy is for this modified MDP; such equations are typically solved

using Value Iteration or variations thereof. However, this problem is slightly more complicated than a standard MDP because one also has to determine the value of β_s. The way that this problem is traditionally solved in the literature is the following:

1. One guesses a value for the subsidy β_s.
2. Given this guess, one solves the Bellman Optimality Equations associated with the modified MDP.
3. Then, one checks the resultant policy. If it is more valuable to act than to not act in state s, the value of the guess for the subsidy is increased. Else, it is decreased.
4. Go to Step 2 and repeat until convergence.

Given the monotonicity and the ability to bound the values of the whittle index, Step 3 above is typically solved using binary search. However, even with Binary Search, this process is quite time-consuming.

In this paper, we provide a much faster solution method for our application of interest. We leverage the small size of our state space to search over the space of policies rather than over the correct value of β_s. Concretely, because $S = \{0, 1\}$ and $A = \{0, 1\}$, the whittle index equations for state $s = 0$ above boil down to:

$$
\begin{aligned}
V(0) &= R(0) + \beta_{s_0} + \gamma \sum_{s' \in \{0,1\}} P(0, 0, s') \cdot V(s') \\
V(0) &= R(0) + \gamma \sum_{s' \in \{0,1\}} P(0, 1, s') \cdot V(s') \\
V(1) &= \max_{a \in \{0,1\}} [R(1) + (1 - a)\beta_{s_0} + \gamma \sum_{s' \in \{0,1\}} P(1, a, s') \cdot V(s')]
\end{aligned} \tag{B.5}
$$

These are 3 equations in 3 unknowns ($V(0), V(1), \beta_{s_0}$). The only hiccup here is that Equation B.5 has a $\max$ term and so this set of equations can not be solved as normal linear equations would be. However, we can 'unroll' Equation B.5 into 2 different equations:

$$V(1) = R(1) + \beta_{s_0} + \gamma \sum_{s' \in \{0,1\}} P(1, 0, s') \cdot V(s'), \qquad \text{or} \tag{B.6}$$

$$V(1) = R(1) + \gamma \sum_{s' \in \{0,1\}} P(1, 1, s') \cdot V(s') \tag{B.7}$$

Each of these corresponds to evaluating the value function associated with the partial policies $s = 1 \rightarrow a = 0$ and $s = 1 \rightarrow a = 1$. Then to get the optimal policy, we can just evaluate both of the policies and choose the better of the two policies, i.e., the policy with the higher expected value $V(1)$. In practice, we pre-compute the Whittle index and value function using the binary search and value iteration approach studied by Qian et al. [258]. Therefore, to determine which equation is satisfied, we just use the pre-computed value functions to evaluate the expected future return of different actions, and use the one with higher value to form a set of linear equations.

This gives us a set of linear equations where Whittle index is a solution. We can therefore derive a closed-form expression of the Whittle index as a function of the transition probabilities, which is differentiable. This completes the differentiability of Whittle index. This technique is equivalent to saying that the policy does not change if we infinitesimally change the input probabilities.

B.8.1 Worked Example

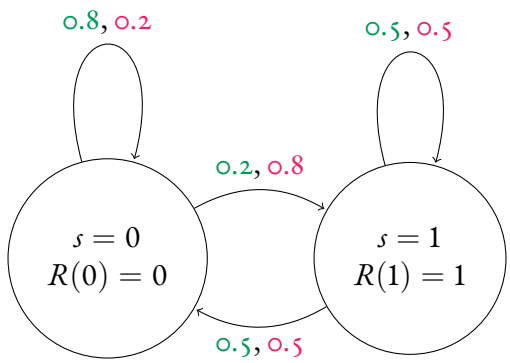


Figure B.6: An MDP with the probabilities associated with the passive action $a = 0$ in green and active action $a = 1$ in red.

Let us consider the concrete example in Figure B.6 with $\gamma = 0.5$. To calculate the whittle index for state $s = 0$, we have to solve the following set of linear equations:

$$
\begin{aligned}
&V(0) = 0 + \beta_{s_0} + 0.5 \cdot [0.8V(0) + 0.2V(1)] && V(0) = 0 + \beta_{s_0} + 0.5 \cdot [0.8V(0) + 0.2V(1)] \\
&V(0) = 0 + 0.5 \cdot [0.2V(0) + 0.8V(1)] && V(0) = 0 + 0.5 \cdot [0.2V(0) + 0.8V(1)] \\
&V(1) = 1 + \beta_{s_0} + 0.5 \cdot [0.5V(0) + 0.5V(1)] && V(1) = 1 + 0.5 \cdot [0.5V(0) + 0.5V(1)] \\
&V(0) \approx 0.65,\ V(1) \approx 1.45, \beta_{s_0} \approx 0.25 && V(0) \approx 0.52,\ V(1) \approx 1.18, \beta_{s_0} \approx 0.20
\end{aligned}
$$

Here the left set of equations corresponds to taking action $a = 0$ in state $s = 1$ and the right corresponds to taking the action $a = 1$. As we can see in the above calculation, given subsidy β_{s_0}, it is better to choose the passive action (a=0) on the left to obtain a higher expected future value $V(1)$. On the other hand, this can also be verified by precomputing the Whittle index and the value function. Therefore, we know that the passive action in Equation B.7 leads to a higher value, where the equality holds. Thus we can express the Whittle index as a solution to the following set of linear

equations:

$$V(0) = R(0) + \beta_{s_0} + \gamma \sum_{s' \in \{0,1\}} P(0, 0, s') \cdot V(s')$$
$$V(0) = R(0) + \gamma \sum_{s' \in \{0,1\}} P(0, 1, s') \cdot V(s')$$
$$V(1) = R(1) + \beta_{s_0} + \gamma \sum_{s' \in \{0,1\}} P(1, 0, s') \cdot V(s')$$

By solving this set of linear equation, we can express the Whittle index β_{s_0} as a function of the transition probabilities. Therefore, we can apply auto-differentiation to compute the derivative $\frac{d\beta_{s_0}}{dP}$.

C
Appendix to Chapter 5

C.1 Computation of Defender Utility

Given coverage z, if we sort the vertices out by intermediate states then absorbing states, then the transition matrix induced by behavior $\theta(z, x)$ can be written as: $P = \begin{bmatrix} Q & R \\ 0 & I \end{bmatrix}$, where I is an identity matrix representing once the attacker reaches any absorbing states, he would never transit to other states. Q, R are both functions of z and x.

The absorbing probability can be computed by $B = (I - Q)^{-1}R \in \mathbb{R}^{|S| \times (|T|+1)}$, where the entry B_{ij} indicates the probability that the attacker initiates from state i and ends up being in absorbing state j. Since we also know the distribution $\pi \in \mathbb{R}^{|S|}$ that the attacker will appear and the defender utility $U^d \in \mathbb{R}^{|T|+1}$ including the reward of catching the attacker, the defender utility can be given by $\pi^\top BU^d$, where the function f is defined by the negative defender utility:

$$f(z, \theta) = -\pi^\top BU^d = -\pi^\top (I - Q(z, \theta))^{-1} R(z, \theta) U^d \tag{C.1}$$

which is still a function of z and θ. We can also compute the derivatives of this function f. But since the computation of f involves matrix inversion, the computation of derivatives will also involve matrix inversions and multiplications, which can be very expensive especially for higher order derivatives.

C.2 The Choices of Loss Function

Given two transition matrices $M, M' \in \mathbb{R}^{|V| \times |V|}$, we can align with the any standard definition of matrix norm:

$$\mathcal{L}(M, M') = \left\| M - M' \right\| \tag{C.2}$$

Another choice of loss function definition is to compute the KL-divergence or cross entropy of the path distribution inferred by these two transition matrices. Since there are absorbing states, the path can eventually terminate when it reaches to any of the absorbing state. However, there could be loop in the graph, which might incur infinitely many possible paths, making the path distribution infinitely dimensional. Another issue is that we do not have a close form of the path distribution, which can prevent us from computing the KL-divergence between two implicit distributions.

In our domain, we usually have samples from the ground truth Markov chain, which can be used as an empirical path distribution. By considering those samples as the empirical distribution Λ, we can compute the KL-divergence between Λ and the predicted Markov chain M:

$$\mathcal{L}(\Lambda, M) = \sum_{\alpha} \text{prob}(\alpha \in \Lambda) \log \frac{\text{prob}(\alpha \in \Lambda)}{\text{prob}(\alpha \in M)} \tag{C.3}$$

which can be efficiently computed since Λ is finite and all the probability can be analytically computed. This serves as an alternative for us to compute the KL-divergence between the ground truth and our prediction.

C.3 Differentiable Quadratic Program

Given a quadratic program:

$$\begin{aligned} \min_{z} \quad & \frac{1}{2} z^T Q z + p^T z \\ \text{s.t.} \quad & Gz \leq h \\ & Az = b \end{aligned} \tag{C.4}$$

According to[80,11], we can compute the derivative of the optimal solution z^* with respect to each parameters in the quadratic program, e.g., Q, p, G, h, A, b. In our case, we only consider the derivative with respect to p, where G, h, A, b are constants and we ignore the derivative of the Hessian Q since its derivative is of third order.

After solving the quadratic program, we can get the solution z^* with dual variables λ^*, ν^* respectively for the inequality constraints and equality constraints. All the variables z^*, λ^*, ν^* are all

functions of prediction θ. In[80,11], they proposed that if we write the KKT conditions:

$$\begin{aligned} Qz^* + p + A^\top \nu^* + G^\top \lambda^* &= 0 \\ Az^* - b &= 0 \\ D(\lambda^*)(Gz^* - h) &= 0 \end{aligned}$$

where $D(\lambda^*)$ is the diagonal matrix with λ^* on the diagonal. We can consider the total derivative of the above equations, which yields:

$$\begin{aligned} dQz^* + Qdz^* + dp + dA^\top \nu^* + A^\top d\nu^* + dG^\top \lambda^* + G^\top d\lambda^* &= 0 \\ dAz^* + Adz^* - db &= 0 \\ D(Gz^* - h)d\lambda^* + D(\lambda^*)(dGz^* + Gdz^* - dh) &= 0 \end{aligned}$$

Since here we assume Q, G, h, A, b are all constants, so we can ignore the derivatives of there terms and get:

$$\begin{aligned} Qdz^* + dp + A^\top d\nu^* + G^\top d\lambda^* &= 0 \\ Adz^* &= 0 \\ D(Gz^* - h)d\lambda^* + D(\lambda^*)Gdz^* &= 0 \end{aligned}$$

which can be further turned into matrix form:

$$\begin{aligned} & \begin{bmatrix} Q & G^\top & A^\top \\ D(\lambda^*)G & D(Gz^* - h) & 0 \\ A & 0 & 0 \end{bmatrix} \begin{bmatrix} dz^* \\ d\lambda^* \\ d\nu^* \end{bmatrix} = \begin{bmatrix} -dp \\ 0 \\ 0 \end{bmatrix} \\ \Leftrightarrow \quad & \begin{bmatrix} Q & G^\top & A^\top \\ D(\lambda^*)G & D(Gz^* - h) & 0 \\ A & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{dz^*}{dp} \\ \frac{d\lambda^*}{dp} \\ \frac{d\nu^*}{dp} \end{bmatrix} = \begin{bmatrix} -\mathbf{I} \\ 0 \\ 0 \end{bmatrix} \\ \Leftrightarrow \quad & \begin{bmatrix} \frac{dz^*}{dp} \\ \frac{d\lambda^*}{dp} \\ \frac{d\nu^*}{dp} \end{bmatrix} = \begin{bmatrix} Q & G^\top & A^\top \\ D(\lambda^*)G & D(Gz^* - h) & 0 \\ A & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} -\mathbf{I} \\ 0 \\ 0 \end{bmatrix} \end{aligned} \tag{C.5}$$

This allows us to compute the gradients $\frac{dz^*}{dp}$ by solving the corresponging linear equation.

We can also combine the chain rule $\frac{df}{dp} = \frac{df}{dz^*}\frac{dz^*}{dp}$ by:

$$\frac{df}{dp} = \begin{bmatrix} \frac{dz^*}{dp} \\ \frac{d\lambda^*}{dp} \\ \frac{d\nu^*}{dp} \end{bmatrix}^\top \begin{bmatrix} \frac{df}{dz^*} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -\mathbf{I} \\ 0 \\ 0 \end{bmatrix}^\top \begin{bmatrix} Q & D(\lambda^*)G^\top & A^\top \\ G & D(Gz^* - h) & 0 \\ A & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \frac{df}{dz^*} \\ 0 \\ 0 \end{bmatrix}$$

Or equivalently, define

$$\begin{bmatrix} dz \\ d\lambda \\ d\nu \end{bmatrix} = \begin{bmatrix} Q & D(\lambda^*)G^\top & A^\top \\ G & D(Gz^* - h) & 0 \\ A & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} -\frac{df}{dz^*} \\ 0 \\ 0 \end{bmatrix} \tag{C.6}$$

then $\frac{df}{dp} = dz$, which is the derivative of the objective function f with respect to the linear coefficient p of the quadratic program.

C.4 Proof of Theorem 3

Theorem 3. *When the intermediate prediction matches the ground truth, i.e., $\theta(\cdot, \cdot; w^*) = \theta^*$, we have $\frac{df(z^*, \theta^*)}{dw}|_{w=w^*} = 0$ for both Algorithm 3 and Algorithm 4 with any block C.*

Proof. We first prove for the Algorithm 3 case. Our goal is to prove that this $\frac{df}{dp}|_{\theta=\theta^*} = d_z$ is exactly 0 at θ^*, meaning there is no gradient at the true optimal prediction θ^*.

To prove this, we directly show that $d_z = 0, d_\lambda = 1_{\{\lambda^* \neq 0\}}, d_\nu = \nu^*$ is a solution. When the KKT matrix in Equation C.5 is non-singular, this implies that $d_z = 0$ is the unique solution. When the KKT matrix is singular, $d_z = 0$ is a subgradient. Furthermore, if we follow the implementation in [80], they remove the dependent rows of the KKT matrix C.5 such that the matrix is non-singular, which again implies that $d_z = 0$ is the unique solution. To verify this, we can compute:

$$\begin{aligned} &\begin{bmatrix} Q & G^\top D(\lambda^*) & A^\top \\ G & D(Gz^* - h) & 0 \\ A & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1_{\{\lambda^* \neq 0\}} \\ \nu^* \end{bmatrix} \\ = &\begin{bmatrix} G^\top D(\lambda^*) 1_{\{\lambda^* \neq 0\}} + A^\top \nu^* \\ D(Gz^* - h) 1_{\{\lambda^* \neq 0\}} \\ 0 \end{bmatrix} = \begin{bmatrix} G^\top \lambda^* + A^\top \nu^* \\ 0 \\ 0 \end{bmatrix} \end{aligned}$$

Notice that the KKT condition of the quadratic program implies:

$$\begin{aligned} & Qz^* + p + A^\top \nu^* + G^\top \lambda^* = 0 \\ \Leftrightarrow \quad & Qz^* + \frac{\partial f}{\partial z}|_{z=z^*, \theta=\theta^*} - Qz^* + A^\top \nu^* + G^\top \lambda^* = 0 \\ \Leftrightarrow \quad & \frac{\partial f}{\partial z}|_{z=z^*, \theta=\theta^*} + A^\top \nu^* + G^\top \lambda^* = 0 \end{aligned}$$

$$
\begin{bmatrix} Q & G^\top D(\lambda^*) & A^\top \\ G & D(Gz^* - h) & 0 \\ A & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1_{\{\lambda^* \neq 0\}} \\ \nu^* \end{bmatrix}
$$
$$
= \begin{bmatrix} G^\top \lambda^* + A^\top \nu^* \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -\frac{df}{dz^*} \\ 0 \\ 0 \end{bmatrix}
$$

This verifies that $d_z = 0, d_\lambda = 1_{\{\lambda^* \neq 0\}}, d_\nu = \nu^*$ is a solution of Equation C.6. This concludes the proof of Algorithm 3.

To prove for Algorithm 4, we consider the following equation:

$$
\begin{bmatrix} dz_C \\ d\lambda \\ d\nu \end{bmatrix} = \begin{bmatrix} Q_C & D(\lambda^*)G_C^\top & A_C^\top \\ G_C & D(G_C z_C^* - h_C) & 0 \\ A_C & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} -\frac{df}{dz_C^*} \\ 0 \\ 0 \end{bmatrix} \tag{C.7}
$$

where $G = \begin{bmatrix} G_C & G_{\overline{C}} \end{bmatrix}, A = \begin{bmatrix} A_C & A_{\overline{C}} \end{bmatrix}$ that G_C, A_C correspond to the coefficients of indices C. $h_C = h - G_{\overline{C}} z_{\overline{C}}^*$ corresponds to the modified linear inequalities without the effect of terms $z_{\overline{C}}$.

We can also verify that $\frac{df}{dp_C}|_{\theta=\theta^*} = dz_C = 0$ is a solution in Equation C.7. By setting $dz_C = 0, d\lambda = 1_{\{\lambda^* \neq 0\}}, d_\nu = \nu^*$, we can find that this also satisfies the Equation C.7.

All of these imply that $\frac{df}{dp_C}|_{\theta=\theta^*} = 0$ (or at least a feasible subderivative). By applying Equation 5.9 of Algorithm 3 or Equation 5.11 of Algorithm 4, we can get $\frac{df(z^*,\theta^*)}{d\theta}|_{\theta=\theta^*} = 0$ where θ^* is the optimal model parameter that gives the correct prediction θ^*. □

C.5 Proof of Theorem 4

Theorem 4. *The quadratic programs in Algorithm 3 and Algorithm 4 share the same primal solutions on the block C. They also share the same dual solution on the non-degenerate constraints containing at least one variable in the block.*

Proof. Since both algorithms are derived from Taylor expansion around a local optimum, the Hessian is always positive definite. Therefore, the solution given by the quadratic program is exactly the same as the local optimum previously computed, which is shared for both algorithms. So both of them share the same primal solutions at indices C.

For the dual solutions, we can write down the quadratic programs C.4 for Algorithm 3 by:

$$
\begin{aligned} \min_z \quad & \frac{1}{2} z^\top Q z + p^\top z \\ \text{s.t.} \quad & Gz \leq h \\ & Az = b \end{aligned} \tag{C.8}
$$

with $Q = \frac{\partial^2 f}{\partial z^2}|_{z=z^*}, p = \frac{\partial f}{\partial z}|_{z=z^*} - Qz^*$. The KKT stationary condition can be given by:

$$\begin{aligned} & Qz^* + p + G^\top \lambda^* + A^\top \nu^* = 0 \\ \Leftrightarrow \quad & Qz^* + \frac{\partial f}{\partial z}|_{z=z^*} - Qz^* + G^\top \lambda^* + A^\top \nu^* = 0 \\ \Leftrightarrow \quad & \frac{\partial f}{\partial z}|_{z=z^*} + G^\top \lambda^* + A^\top \nu^* = 0 \end{aligned} \tag{C.9}$$

Similarly for Algorithm 4 in the case there is no degenerative constraint, we have:

$$\begin{aligned} \min_{z_C} \quad & \frac{1}{2} z_C^\top Q_{CC} z_C + p_C^\top z_C \\ \text{s.t.} \quad & G_C z_C \leq h_C = h - G_{\overline{C}} z_{\overline{C}} \\ & A_C z_C = b_C = b - A_{\overline{C}} z_{\overline{C}} \end{aligned} \tag{C.10}$$

where $Q_{CC} = \frac{\partial^2 f}{\partial z_C^2}|_{z=z^*}, p_C = \frac{\partial f}{\partial z_C}|_{z=z^*} - Q_{CC} z_C^*$, and constraints $G = \begin{bmatrix} G_C & G_{\overline{C}} \end{bmatrix}, A = \begin{bmatrix} A_C & A_{\overline{C}} \end{bmatrix}$. The KKT stationary condition can be given by:

$$\begin{aligned} & Q_{CC} z_C^* + p_C + G_C^\top \lambda^* + A_C^\top \nu^* = 0 \\ \Leftrightarrow \quad & Q_{CC} z_C^* + \frac{\partial f}{\partial z_C}|_{z=z^*} - Q_{CC} z_C^* + G_C^\top \lambda^* + A_C^\top \nu^* = 0 \\ \Leftrightarrow \quad & \frac{\partial f}{\partial z_C}|_{z=z^*} + G_C^\top \lambda^* + A_C^\top \nu^* = 0 \end{aligned} \tag{C.11}$$

Comparing Equation C.9 and Equation C.11, we can find that Equation C.11 is just a subset of Equation C.9, or more specifically the equations at indices C. Similarly, they also share the same primal, dual feasibility conditions, and complementary slackness conditions. Therefore, the dual solutions of the KKT conditions of quadratic program C.8 is also a solution of the KKT conditions of C.10.

When there are degenerative constraints, for example, some rows $\overline{R}$ of the constraints G_C are degenerative and thus be all 0 after truncating by block C, i.e., $G_{\overline{R},C} = 0$. In this case, $G_C^\top \lambda^* = G_{R,C}^\top \lambda_R^* + G_{\overline{R},C}^\top \lambda_{\overline{R}}^* = G_{R,C}^\top \lambda_R^*$, where there is no constraint posted on $\lambda_{\overline{R}}^*$, which can be arbitrary here. Similarly, some rows $\overline{L}$ of equality constraints A_C might also be degenerative, i.e., $A_{\overline{L},C} = 0$. But if we only consider the non-degenerative constraints $G_{R,C}$ and $A_{L,C}$, we can re-write the KKT stationary conditions in Equation C.11 by:

$$\begin{aligned} & \frac{\partial f}{\partial z_C}|_{z=z^*} + G_C^\top \lambda^* + A_C^\top \nu^* = 0 \\ \Leftrightarrow \quad & \frac{\partial f}{\partial z_C}|_{z=z^*} + G_{R,C}^\top \lambda^* + A_{L,C}^\top \nu^* = 0 \end{aligned} \tag{C.12}$$

In this case, the entire KKT condition with non-degenerative dual variables is non-singular, which imposes a unique solution to the dual variables. But we have shown that the dual solution of Equation C.9 is also a solution to Equation C.11, which is again a solution to Equation C.12. By uniqueness, this solution of Equation C.12is also a solution of Equation C.11 on the non-degenerative constraints $G_{R,C}, A_{L,C}$, thus a solution to the Equation C.9, which concludes the proof. □

C.6 Proof of Theorem 5

Theorem 5. *Given the primal solution z^* and the dual solution λ^* of the quadratic program in Algorithm 3 with linear constraints G, h, A, b, the Hessian $Q = \frac{\partial^2 f}{\partial z}$, linear coefficient $p = \frac{\partial f}{\partial z}$, and the sampled indices $C \subset \{1, 2, ..., |E|\}$, the gradient $\frac{dz_C^*}{dp_C} \in \mathbb{R}^{|C|\times|C|}$ computed in Algorithm 4 is an approximation to the block component of the gradient $\frac{dz^*}{dp} \in \mathbb{R}^{|E|\times|E|}$ computed in Algorithm 3. More specifically,*

$$\left\| \left(\frac{dz^*}{dp}\right)_{CC} - \frac{dz_C^*}{dp_C} \right\| \leq \frac{\Delta + \Delta_C}{\mu_{min}(Q)} \max(\lambda^*, 1) \left\| K_{CC}^{-1} \right\| \left\| \left(\frac{dz^*}{dp}\right)_{CC} \right\| \tag{5.12}$$

where $\Delta = \left\| G^\top G + A^\top A \right\|$, $\Delta_C = \left\| Q_{C\bar{C}} Q_{\bar{C}C} \right\|$, and $\mu_{min}(Q)$ is the smallest eigenvalue of positive definite matrix Q. K_{CC} is the KTT matrix given by the quadratic program in Algorithm 4.

Proof. Denote $K = \begin{bmatrix} Q & G^\top & A^\top \\ D(\lambda^*)G & D(Gz^* - h) & 0 \\ A & 0 & 0 \end{bmatrix}$ to be the KKT matrix C.5 of the quadratic program C.8 given by Algorithm 3. We can also denote $K_{CC} = \begin{bmatrix} Q_{CC} & G_C^\top & A_C^\top \\ D(\lambda^*)G_C & D(G_C z_C^* - h_C) & 0 \\ A_C & 0 & 0 \end{bmatrix}$ to be the KKT matrix of the quadratic program C.10 given by Algorithm 4. Notice that K_{CC} is in fact a block of K since they share the same primal and dual solution. According to Equation C.5, we can write down the gradient $\frac{dz^*}{dp}$ and $\frac{dz_C^*}{dp_C}$ respectively in Algorithm 3 and Algorithm 4 by:

$$\frac{dz^*}{dp} = \begin{bmatrix} \mathbf{I} \\ 0 \\ 0 \end{bmatrix}^\top K^{-1} \begin{bmatrix} -\mathbf{I} \\ 0 \\ 0 \end{bmatrix}, \quad \frac{dz_C^*}{dp_C} = \begin{bmatrix} \mathbf{I} \\ 0 \\ 0 \end{bmatrix}^\top K_{CC}^{-1} \begin{bmatrix} -\mathbf{I} \\ 0 \\ 0 \end{bmatrix}$$

If we use block form to represent the KKT matrix K, we can get:

$$K = \begin{bmatrix} K_1 & K_2 \\ K_3 & K_4 \end{bmatrix}$$

where we can apply the block matrix inversion technique and get:

$$K^{-1} = \begin{bmatrix} K_1^{-1} + K_1^{-1}K_2(K_4 - K_3K_1^{-1}K_2)^{-1}K_3K_1^{-1} & -K_1^{-1}K_2(K_4 - K_3K_1^{-1}K_2)^{-1} \\ -(K_4 - K_3K_1^{-1}K_2)^{-1}K_3K_1^{-1} & (K_4 - K_3K_1^{-1}K_2)^{-1} \end{bmatrix} \tag{C.13}$$

where K_1 needs to be invertible here.

Set $K_1 = Q_{\overline{C}\overline{C}}, K_2 = \begin{bmatrix} Q_{C\overline{C}} & G_{\overline{C}}^\top & A_{\overline{C}}^\top \end{bmatrix}, K_3 = \begin{bmatrix} Q_{\overline{C}C} \\ D(\lambda^*)G_{\overline{C}} \\ A_{\overline{C}} \end{bmatrix}, K_4 = K_{CC}$, where $K_1 = Q_{\overline{C}\overline{C}}$ is positive definite therefore also invertible. We can see that $K_1 \in \mathbb{R}^{|\overline{C}|\times|\overline{C}|}$ and the sizes of K_2, K_3, K_4 depend on the size of the block C and the size of the constraints G_C, A_C, which can probably help visualize the size of the block matrix.

If we truncate the gradient $\frac{dz^*}{dp}$ to its C block, it is equivalent to remove the $\overline{C}$ part from K^{-1}, which gives us:

$$\left(\frac{dz^*}{dp}\right)_{CC} = \begin{bmatrix} \mathbf{I} \\ 0 \\ 0 \end{bmatrix}^\top \left(K^{-1}\right)_{CC} \begin{bmatrix} -\mathbf{I} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{I} \\ 0 \\ 0 \end{bmatrix}^\top (K_4 - K_3K_1^{-1}K_2)^{-1} \begin{bmatrix} -\mathbf{I} \\ 0 \\ 0 \end{bmatrix}$$

Therefore, the difference between $(\frac{dz^*}{dp})_{CC}$ and $\frac{dz_C^*}{dp_C}$ can be bounded by:

$$\begin{aligned}
\left(\frac{dz^*}{dp}\right)_{CC} - \frac{dz_C^*}{dp_C} &= \begin{bmatrix} \mathbf{I} \\ 0 \\ 0 \end{bmatrix}^\top (K_4 - K_3K_1^{-1}K_2)^{-1} \begin{bmatrix} -\mathbf{I} \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} \mathbf{I} \\ 0 \\ 0 \end{bmatrix}^\top K_{CC}^{-1} \begin{bmatrix} -\mathbf{I} \\ 0 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{I} \\ 0 \\ 0 \end{bmatrix}^\top ((K_4 - K_3K_1^{-1}K_2)^{-1} - K_{CC}^{-1}) \begin{bmatrix} -\mathbf{I} \\ 0 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{I} \\ 0 \\ 0 \end{bmatrix}^\top (K_4 - K_3K_1^{-1}K_2)^{-1}(I - (K_4 - K_3K_1^{-1}K_2)K_{CC}^{-1}) \begin{bmatrix} -\mathbf{I} \\ 0 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{I} \\ 0 \\ 0 \end{bmatrix}^\top (K_4 - K_3K_1^{-1}K_2)^{-1}(K_3K_1^{-1}K_2K_{CC}^{-1}) \begin{bmatrix} -\mathbf{I} \\ 0 \\ 0 \end{bmatrix}
\end{aligned} \tag{C.14}$$

where the last equality comes from the choice $K_4 = K_{CC}$, thus the identity matrix is canceled out. We can then bound the matrix norm of $K_3K_1^{-1}K_2K_{CC}^{-1}$ by:

$$\left\|K_3K_1^{-1}K_2K_{CC}^{-1}\right\| \leq \|K_3K_2\| \left\|Q_{\overline{C}\overline{C}}^{-1}\right\| \left\|K_{CC}^{-1}\right\|$$

$$
\begin{aligned}
&\leq \frac{\max(\lambda^*, 1) \left\| K_2^\top K_2 \right\|}{\mu_{\min}(Q_{\overline{CC}})} \left\| K_{CC}^{-1} \right\| \\
&\leq \frac{\Delta + \Delta_C}{\mu_{\min}(Q_{\overline{CC}})} \max(\lambda^*, 1) \left\| K_{CC}^{-1} \right\| \\
&\leq \frac{\Delta + \Delta_C}{\mu_{\min}(Q)} \max(\lambda^*, 1) \left\| K_{CC}^{-1} \right\| \qquad \text{(C.15)}
\end{aligned}
$$

where the second inequality is from the fact that K_3 is a matrix multiplication of $K_2^\top$ and a diagonal matrix with 1 and λ^* on the diagonal. The matrix norm can be bounded by the matrix norm of the diagonal matrix, thus $\max(\lambda^*, 1)$, and the remaining matrix multiplication $K_2^\top K_2$. The third inequality is due to the singular value $\left\| K_2^\top K_2 \right\| = \left\| K_2 K_2^\top \right\| = \left\| Q_{C\overline{C}} Q_{\overline{C}C} + G_{\overline{C}}^\top G_{\overline{C}} + A_{\overline{C}}^\top A_{\overline{C}} \right\| \leq \left\| Q_{C\overline{C}} Q_{\overline{C}C} + G^\top G + A^\top A \right\| \leq \Delta + \Delta_C$, where the middle inequality is due to the fact that all these individual terms are positive semi-definite, so adding new postive semi-definite terms such that they become $G^\top G, A^\top A$ only increases the norm value.

Taking matrix norm to Inequality C.14 and using Inequality C.15 to substitute $\left\| K_3 K_1^{-1} K_2 K_{CC}^{-1} \right\|$, we can get:

$$
\begin{aligned}
\left\| \left(\frac{dz^*}{dp} \right)_{CC} - \frac{dz_C^*}{dp_C} \right\| &\leq \frac{\Delta + \Delta_C}{\mu_{\min}(Q)} \left\| K_{CC}^{-1} \right\| \left\| \begin{bmatrix} \mathbf{I} \\ 0 \\ 0 \end{bmatrix} (K_4 - K_3 K_1^{-1} K_2)^{-1} \begin{bmatrix} -\mathbf{I} \\ 0 \\ 0 \end{bmatrix} \right\| \\
&= \frac{\Delta + \Delta_C}{\mu_{\min}(Q)} \left\| K_{CC}^{-1} \right\| \left\| \left(\frac{dz^*}{dp} \right)_{CC} \right\| \qquad \text{(C.16)}
\end{aligned}
$$

which concludes the proof. □

C.6.1 Discussion of Singularity of KKT Matrix

One biggest concern is whether the KKT matrices K and K_{CC} are singular. If the chosen K_{CC} is singular, then the bound provided in Theorem 5 becomes useless.

As discussed in the appendix of[80], in Equation C.5, due to KKT condition, at least one of λ_i^* and $(Gz^* - h)_i$ is 0. Also as they discussed, when both of them are 0, the whole i-th row in $D(\lambda^*)G$ and $Gz^* - h$ is all 0. We can either impose new constraint or just remove the row to make the matrix non-singular.

If $\lambda_i^* = 0$ with $(Gz^* - h)_i > 0$, then in the i-th row, there is only the term $(Gz^* - h)_i$ being nonzero. Thus we can solve the equation in the i-th row by setting $(\frac{d\lambda^*}{dp})_i = 0$ and remove the row and the variable $(\frac{d\lambda^*}{dp})_i$ from the linear equation. Therefore, the linear equation and the matrix can

be simplified by:

$$\begin{bmatrix} Q & G_I^\top & A^\top \\ D(\lambda_I^*)G_I & D(Gz^* - h)_I & 0 \\ A & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{dz^*}{dp} \\ \frac{d\lambda_I^*}{dp} \\ \frac{d\nu^*}{dp} \end{bmatrix} = \begin{bmatrix} -\mathbf{I} \\ 0 \\ 0 \end{bmatrix}$$

where $I = \{i : \lambda_i^* \neq 0\}$. But notice that $(Gz^* - h)_i = 0$ due to the KKT conditions and the assumption of I. So we can simply write:

$$\begin{bmatrix} Q & G_I^\top & A^\top \\ D(\lambda_I^*)G_I & 0 & 0 \\ A & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{dz^*}{dp} \\ \frac{d\lambda_I^*}{dp} \\ \frac{d\nu^*}{dp} \end{bmatrix} = \begin{bmatrix} -\mathbf{I} \\ 0 \\ 0 \end{bmatrix} \tag{C.17}$$

Notice that we can assume $\begin{bmatrix} D(\lambda^*)G_I \\ A_I \end{bmatrix}$ to have a full row rank now. Equivalently, we can also assume $\begin{bmatrix} G_I^\top & A_I^\top \end{bmatrix}$ to have a full column rank.

C.6.2 Singularity of Block KKT Matrix

Given a simplified version of the non-singular full KKT matrix in Equation C.17, we can write down the block KKT matrix as:

$$\begin{bmatrix} Q_{CC} & G_{I,C}^\top & A_C^\top \\ D(\lambda_I^*)G_{I,C} & 0 & 0 \\ A_C & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{dz_C^*}{dp_C} \\ \frac{d\lambda_I^*}{dp_C} \\ \frac{d\nu^*}{dp_C} \end{bmatrix} = \begin{bmatrix} -\mathbf{I} \\ 0 \\ 0 \end{bmatrix} \tag{C.18}$$

where $G_I = \begin{bmatrix} G_{I,C} & G_{I,\overline{C}} \end{bmatrix}, A_I = \begin{bmatrix} A_{I,C} & A_{I,\overline{C}} \end{bmatrix}$. In order to make the block KKT matrix non-singular, we need to select C such that $\begin{bmatrix} G_C \\ A_C \end{bmatrix}$ remains full row rank. In this case, the block KKT matrix will remain non-singular and thus invertible.

In practice, we cannot access to the dual varialbe λ^* before solving the QP and choosing the block C. But we can compute the slack variables $Gz^{\text{opt}} - h$ since z^{opt} is given. We need to make sure to make $G_{I,C}$ nonzero for $I = \{i : \lambda_i^* \neq 0\} \subset \{i : (Gz^{\text{opt}} - h)_i = 0\}$, or equivalently the indices of tight constraints.

Some choices of block C might make Equation C.18 singular but still solvable. That is due to some dependent rows in $\begin{bmatrix} G_{I,C} \\ A_C \end{bmatrix}$, which admit to each others since the right hand side is all 0. This allows us to remove the redundant constraints and re-solve the linear equation by applying matrix inversion. But in this case, the block KKT matrix will not contain all the constraints, which leaves some constraints out of the block. Algorithm 4 still works but the K_1 in the proof of Theorem 5 is

not just $Q_{\overline{CC}}$ but contains some additional terms from constraints excluded by the block quadratic program. The bound will also vary since we need to estimate the eigenvalue of K_1, which depends on the added constraints and does not have a simple form here.

D
Appendix to Chapter 6

D.1 Preservation of Convexity and Submodularity

Proposition 1. *If f is convex, then $g_P(y, \theta) := f(Py, \theta)$ is convex.*

Proof. The convexity can be simply verified by computing the second-order derivative:

$$\frac{d^2 g}{dy^2} = \frac{d^2 f(Py, \theta)}{dy^2} = P^\top \frac{d^2 f}{dz^2} P \succeq 0$$

where the last inequality comes from the convexity of f, i.e., $\frac{d^2 f}{dz^2} \succeq 0$. □

Proposition 2. *If f is DR-submodular and $P \geq 0$, then $g_P(y, \theta) := f(Py, \theta)$ is DR-submodular.*

Proof. Assume f has the property of diminishing return submodularity (DR-submodular)[46]. According to definition of continuous DR-submodularity, we have:

$$\nabla^2_{z_i, z_j} f(z, \theta) \leq 0 \; \forall i, j \in [n], y$$

After applying the reparameterization, we can write:

$$g_P(y, \theta) = f(z, \theta)$$

and the second-order derivative:

$$\nabla_y^2 g_P(y,\theta) = P^\top \nabla_z^2 f_P(z,\theta) P \leq 0$$

Since all the entries of P are non-negative and all the entries of $\nabla_z^2 f_P(z,\theta)$ are non-positive by DR-submodularity, the product $\nabla_y^2 g_P(y,\theta)$ also has all the entries being non-positive, which satisfies the definition of DR-submodularity. □

D.2 Quasiconvexity in Reparameterization Matrix

Proposition 3. *$OPT(\theta, P) := \min_{y \text{ feasible}} g_P(y,\theta)$ is not globally quasiconvex in P.*

Proof. Without loss of generality, let us ignore the effect of θ and write $g_P(y) = f(Pz)$. In this proof, we will construct a strongly convex function f where the induced optimal value function $\mathrm{OPT}(P) := \min_y g_P(y)$ is not quasiconvex.

Consider $z = [z_1, z_2, z_3]^\top \in \mathbb{R}^3$. Define $f(z) = \left\| z - \begin{pmatrix}1\\1\\1\end{pmatrix} \right\|^2 \geq 0$ for all $z \in \mathbb{R}^3$. Define $P = \begin{pmatrix}1 & 0\\1 & 0\\0 & 2\end{pmatrix}$ and $P' = \begin{pmatrix}0 & 1\\0 & 1\\2 & 0\end{pmatrix}$. Apparently, $z^* = \begin{pmatrix}1\\1\\1\end{pmatrix} = P\begin{pmatrix}1\\0.5\end{pmatrix}$ and $z^* = \begin{pmatrix}1\\1\\1\end{pmatrix} = P'\begin{pmatrix}0.5\\1\end{pmatrix}$ are both achievable. So the optimal values $\mathrm{OPT}(P) = \mathrm{OPT}(P') = 0$. But the combination $P'' = \frac{1}{2}P + \frac{1}{2}P' = \begin{pmatrix}0.5 & 0.5\\0.5 & 0.5\\1 & 1\end{pmatrix}$ cannot, which results in an optimal value $OPT(P'') = \min_y g_{P''}(y) =>$ 0 since $\begin{pmatrix}1\\1\\1\end{pmatrix} \notin \mathrm{span}(P'')$. This implies $\mathrm{OPT}(\frac{1}{2}P + \frac{1}{2}P') = \mathrm{OPT}(P'') > 0 = \frac{1}{2}\mathrm{OPT}(P) + \frac{1}{2}\mathrm{OPT}(P')$. Thus $\mathrm{OPT}(P)$ is not globally convex in the feasible domain. □

Theorem 6. *If $f(\cdot,\theta)$ is quasiconvex, then $OPT(\theta, P) := \min_{y \text{ feasible}} g_P(y,\theta)$ is quasiconvex in P_i, the i-th column of matrix P, for any $1 \leq i \leq m$, where $P = [P_1, P_2, \ldots, P_m]$.*

Proof. Let us assume $P = [p_1, p_2, \ldots, p_m]$ and $P' = [p'_1, p'_2, \ldots, p'_m]$, where $p_i = p'_i \, \forall i \neq 1$ with only the first column different. In the optimization problem parameterized by P, there is an optimal solution $z = \sum_{i=1}^m p_i y_i, y_i \geq 0 \, \forall i$. Similarly, there is an optimal solution $z' = \sum_{i=1}^m p'_i y'_i, y'_i \geq 0 \, \forall i$ for the optimization problem parameterized by P'. Denote $h(P) := \mathrm{OPT}(\theta, P)$. We know that $f(z) = h(P), f(z') = h(P')$. Denote $P'' = cP + (1-c)P' = [p''_1, p''_2, \ldots, p''_m]$ to be a convex combination of P and P'. Clearly, $p''_1 = cp_1 + (1-c)p'_1$ and $p''_i = p_i = p'_i \, \forall i \neq 1$. Then we can

construct a solution

$$
\begin{aligned}
z'' &= \frac{1}{\frac{c}{y_1} + \frac{1-c}{y_1'}}\left(\frac{c}{y_1}z + \frac{1-c}{y_1'}z'\right) \\
&= \frac{1}{\frac{c}{y_1} + \frac{1-c}{y_1'}}\left(\frac{c}{y_1}\sum_{i=1}^{m} p_i y_i + \frac{1-c}{y_1'}\sum_{i=1}^{m} p_i' y_i'\right) \\
&= \frac{1}{\frac{c}{y_1} + \frac{1-c}{y_1'}}(cp_1 + (1-c)p_1') + \frac{1}{\frac{c}{y_1} + \frac{1-c}{y_1'}}\sum_{i=2}^{m} p_i\left(\frac{y_i}{y_1} + \frac{y_i'}{y_1'}\right) \\
&\in \text{Span}(P'')
\end{aligned}
$$

Thus, z'' is a feasible solution in the optimization problem parameterized by P''. By the convexity of f, we also know that

$$
\begin{aligned}
h(cP + (1-c)P') = h(P'') &\leq f(z'') \\
&= f\left(\frac{1}{\frac{c}{y_1} + \frac{1-c}{y_1'}}\left(\frac{c}{y_1}z + \frac{1-c}{y_1'}z'\right)\right) \\
&\leq \max(f(z), f(z')) \\
&= \max(h(P), h(P'))
\end{aligned}
$$

When one of y_1, y_1' is 0, without loss of generality we assume $y_1 = 0$. Then we can construct a solution $z'' = z$ which is still feasible in the optimization problem parameterized by $P'' = cP + (1-c)P'$. Then we have the following:

$$
h(P'') \leq f(z'') = f(z) = h(P) \leq \max(h(P), h(P'))
$$

which concludes the proof. □

D.3 Sample Complexity of Learning Predictive Model in Surrogate Problem

Theorem 7. *Let $\mathcal{H}_{lin}$ be the hypothesis class of all linear function mappings from $x \in \mathcal{X} \subset \mathbb{R}^p$ to $\theta \in \Theta \in \mathbb{R}^n$, and let $P \in \mathbb{R}^{n \times m}$ be a linear reparameterization used to construct the surrogate. The expected Rademacher complexity over t i.i.d. random samples drawn from $\mathcal{D}$ can be bounded by:*

$$
Rad^t(\mathcal{H}_{lin}) \leq 2mC\sqrt{\frac{2p\log(2mt\,\|P^+\|_{\rho_2}(S))}{t}} + O\left(\frac{1}{t}\right) \tag{6.4}
$$

where $C := sup_\theta(max_z f(z,\theta) - min_z f(z,\theta))$ is the gap between the optimal solution quality and the worst solution quality, $\rho_2(S)$ is the diameter of the set S, and P^+ is the pseudoinverse.

The proof of Theorem 7 relies on the results given by Balghiti et al.[87]. Balghiti et al. analyzed

the sample complexity of predict-then-optimize framework when the optimization problem is a constrained linear optimization problem.

The sample complexity depends on the hypothesis class $\mathcal{H}$, mapping from the feature space $\mathcal{X}$ to the parameter space Θ. $z_S^*(\theta) = \operatorname{argmin}_{z \in S} f(z, \theta)$ characterizes the optimal solution with given parameter $\theta \in \Theta$ and feasible region S. This can be obtained by solving any linear program solver with given parameters θ. The optimization gap with given parameter P is defined as $\omega_S(\theta) := \max_{z \in S} f(z, \theta) - \min_{z \in S} f(z, \theta)$, and $\omega_S(\Theta) := \sup_{\theta \in \Theta} \omega_S(\theta)$ is defined as the upper bound on optimization gap of all the possible parameter $\theta \in \Theta$. $z^*(\mathcal{H}) := \{x \to z^*(\Phi(x)) | \Phi \in \mathcal{H}\}$ is the set of all function mappings from features x to the predictive parameters $\theta = \Phi(x)$ and then to the optimal solution $z^*(\theta)$.

Definition 28 (Natarajan dimension)**.** *Suppose that S is a polyhedron and $\mathfrak{S}$ is the set of its extreme points. Let $\mathcal{F} \in \mathfrak{S}^{\mathcal{X}}$ be a hypothesis space of function mappings from $\mathcal{X}$ to $\mathfrak{S}$, and let $A \subset \mathcal{X}$ to be given. We say that $\mathcal{F}$ shatters A if there exists $g_1, g_2 \in \mathcal{F}$ such that*

- *$g_1(x) \neq g_2(x)\ \forall x \in A$.*
- *For all $B \subset A$, there exists $g \in \mathcal{F}$ such that (i) for all $x \in B, g(x) = g_1(x)$ and (ii) for all $x \in A \backslash B, g(x) = g_2(x)$.*

The Natarajan dimension of $\mathcal{F}$, denoted by $d_N(\mathcal{F})$, is the maximum cardinality of a set N-shattered by $\mathcal{F}$.

We first state their results below:

Theorem 29 (Balghiti et al.[87] Theorem 2)**.** *Suppose that S is a polyhedron and $\mathfrak{S}$ is the set of its extreme points. Let $\mathcal{H}$ be a family of functions mapping from features $\mathcal{X}$ to parameters $\Theta \in \mathbb{R}^n$ with decision variable $z \in \mathbb{R}^n$ and objective function $f(z, \theta) = \theta^\top z$. Then we have that*

$$Rad^t(\mathcal{H}) \leq \omega_S^*(\Theta) \sqrt{\frac{2 d_N(z^*(\mathcal{H})) \log(t|\mathfrak{S}|^2)}{t}}. \tag{D.1}$$

where Rad^t denotes the Radamacher complexity averaging over all the possible realization of t i.i.d. samples drawn from distribution $\mathcal{D}$.

The following corollary provided by Balghiti et al.[87] introduces a bound on Natarajan dimension of linear hypothesis class $\mathcal{H}$, mapping from $\mathcal{X} \in \mathbb{R}^p$ to $\Theta \in \mathbb{R}^n$:

Corollary 3 (Balghiti et al.[87] Corollary 1)**.** *Suppose that S is a polyhedron and $\mathfrak{S}$ is the set of its extreme points. Let $\mathcal{H}_{lin}$ be the hypothesis class of all linear functions, i.e., $\mathcal{H}_{lin} = \{x \to Bx | B \in \mathbb{R}^{n \times p}\}$. Then we have*

$$d_N(z^*(\mathcal{H}_{lin})) \leq np \tag{D.2}$$

Also $|\mathfrak{S}|$ can be estimated by constructing an ε-covering of the feasible region by open balls with radius ε. Let $\hat{\mathfrak{S}}_\varepsilon$ be the centers of all these open balls. We can choose $\varepsilon = \frac{1}{t}$ and the number of open balls required to cover S can be estimated by

$$|\hat{\mathfrak{S}}_\varepsilon| \leq \left(2t\rho_2(S)\sqrt{n}\right)^n \tag{D.3}$$

Combining Equation D.1, D.2, and D.3, the Radamacher complexity can be bounded by:

Corollary 4 (Balghiti et al.[87] Corollary 2)**.**

$$Rad^t(\mathcal{H}_{lin}) \leq 2n\omega_S(\Theta)\sqrt{\frac{2p\log(2nt\rho_2(S))}{t}} + O(\frac{1}{t}) \tag{D.4}$$

Now we are ready to prove Theorem 7:

Proof of Theorem 7. Now let us consider our case. We have a linear mapping from features $x \in \mathcal{X} \subset \mathbb{R}^p$ to the parameters $\theta = Bx \in \Theta \in \mathbb{R}^n$ with $B \in \mathbb{R}^{n\times p}$. The objective function is formed by

$$g_P(y,\theta) = f(Py,\theta) = \theta^\top Py = (P^\top\theta)^\top y = (P^\top Bx)^\top y \tag{D.5}$$

This is equivalent to have a linear mapping from $x \in \mathcal{X} \subset \mathbb{R}^p$ to $\theta' = P^\top Bx$ where $P^\top B \in \mathbb{R}^{m\times p}$, and the objective function is just $g_P(y,\theta') = \theta'^\top y$. This yields a similar bound but with a smaller dimension $m \ll n$ as in Equation D.6:

$$\mathrm{Rad}^t(\mathcal{H}_{\mathrm{lin}}) \leq 2m\omega_S(\Theta)\sqrt{\frac{2p\log(2mt\rho_2(S'))}{t}} + O(\frac{1}{t}) \tag{D.6}$$

where $\omega_S(\Theta)$ is unchanged because the optimality gap is not changed by the reparameterization. The only thing changed except for the substitution of m is that the feasible region S' is now defined in a lower-dimensional space under reparameterization P. But since $\forall y \in S'$, we have $Py \in S$ too. So the diameter of the new feasible region can also be bounded by:

$$\begin{aligned}
\rho(S') &= \max_{y,y'\in S'} \left\|y - y'\right\| \\
&= \max_{y,y'\in S'} \left\|P^+P(y - y')\right\| \\
&= \max_{y,y'\in S'} \left\|P^+(Py - Py')\right\| \\
&\leq \max_{z,z'\in S'} \left\|P^+(z - z')\right\| \\
&\leq \left\|P^+\right\| \max_{z,z'\in S'} \left\|z - z'\right\| \\
&= \left\|P^+\right\| \rho(S)
\end{aligned}$$

where $P^+ \in \mathbb{R}^{m\times n}$ is the pseudoinverse of the reparameterization matrix P with $P^+P = I \in \mathbb{R}^{m\times m}$ (assuming the matrix does not collapse). Substituting the term $\rho(S')$ in Equation D.6, we

can get the bound on the Radamacher complexity in Equation 6.4, which concludes the proof of Theorem 7. □

D.4 Non-linear Reparameterization

The main reason that we use a linear reparameterization is to maintain the convexity of the inequality constraints and the linearity of the equality constraints. Instead, if we apply a convex reparameterization $z = P(y)$, e.g., an input convex neural network[13], then the inequality constraints will remain convex but the equality constraints will no longer be affine anymore. So such convex reparameterization can be useful when there is no equality constraint. Lastly, we can still apply non-convex reparameterization but it can create non-convex inequality and equality constraints, which can be challenging to solve. All of these imply that the choice of reparameterization should depend on the type of optimization problem to make sure we do not lose the scalability while solving the surrogate problem.

D.5 Computing Infrastructure

All experiments were run on the computing cluster, where each node configured with 2 Intel Xeon Cascade Lake CPUs, 184 GB of RAM, and 70 GB of local scratch space. Within each experiment, we did not implement parallelization. So each experiment was purely run on a single CPU core. The main bottleneck of the computation is on solving the optimization problem, where we use Scipy[313] blackbox optimization solver using SLSQP method. No GPU was used to train the neural network and throughout the experiments.

E
Appendix to Chapter 7

E.1 Proofs of Proposition 4

and Proposition 5

Proposition 4. *The variance returned by Algorithm 5 is*

$$\sigma^2_{T,entire}(\boldsymbol{x}) = k(\boldsymbol{x},\boldsymbol{x}) - \sum_{i,j} \boldsymbol{z}_i^\top (\sum_l \boldsymbol{D}_l \boldsymbol{K}_{l,T} \boldsymbol{D}_l)^{-1} \boldsymbol{z}_j \tag{7.6}$$

where $\boldsymbol{D}_j = diag([g_j(\boldsymbol{x}_1), ..., g_j(\boldsymbol{x}_T)])$ *and* $\boldsymbol{z}_i = \boldsymbol{D}_i \boldsymbol{k}_{j,T}(\boldsymbol{x}) g_j(\boldsymbol{x}) \in \mathbb{R}^T$.

Proof of Proposition 4. According to Equation (7.4), the posterior covariance $k_{T,\text{entire}}(\boldsymbol{x},\boldsymbol{x}')$ in Algorithm 5 can be written as:

$$k_{T,\text{entire}}(\boldsymbol{x},\boldsymbol{x}') = k(\boldsymbol{x},\boldsymbol{x}') - \boldsymbol{k}_T(\boldsymbol{x})^\top \boldsymbol{K}_T{}^{-1} \boldsymbol{k}_T(\boldsymbol{x}) \tag{E.1}$$

By the decomposition assumption (Equation (7.1)), we have $k(\boldsymbol{x},\boldsymbol{x}') = \sum_{j=1}^{J} g_j(\boldsymbol{x}) k_j(\boldsymbol{x},\boldsymbol{x}') g_j(\boldsymbol{x}')$.

Moreover,

$$
\begin{aligned}
\boldsymbol{k}_T(\boldsymbol{x}) &= [k(\boldsymbol{x}_1, \boldsymbol{x}), \ldots, k(\boldsymbol{x}_T, \boldsymbol{x})]^\top \\
&= \sum_{j=1}^{J} [g_j(\boldsymbol{x}_1) k_j(\boldsymbol{x}_1, \boldsymbol{x}) g_j(\boldsymbol{x}), \ldots, g_j(\boldsymbol{x}_T) k_j(\boldsymbol{x}_T, \boldsymbol{x}) g_j(\boldsymbol{x})]^\top \\
&= \sum_{j=1}^{J} \boldsymbol{D}_j \boldsymbol{k}_{j,T}(\boldsymbol{x}) g_j(\boldsymbol{x}) \qquad \text{(E.2)}
\end{aligned}
$$

where $\boldsymbol{k}_{j,T}(\boldsymbol{x}) = [k_j(\boldsymbol{x}_1, \boldsymbol{x}), \ldots, k_j(\boldsymbol{x}_T, \boldsymbol{x})]^\top$.

The variance function $\sigma_T^2(\boldsymbol{x})$ is just the value of covariance function with $\boldsymbol{x}' = \boldsymbol{x}$. Therefore, combining Equation (E.1) and (E.2), the variance can be written as:

$$
\begin{aligned}
\sigma_{T,\text{entire}}^2(\boldsymbol{x}) &= k_{T,\text{entire}}(\boldsymbol{x}, \boldsymbol{x}) \\
&= k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}_T(\boldsymbol{x})^\top \boldsymbol{K}_T^{-1} \boldsymbol{k}_T(\boldsymbol{x}) \\
&= k(\boldsymbol{x}, \boldsymbol{x}) - \sum_{i,j} g_i(\boldsymbol{x}) \boldsymbol{k}_{i,T}(\boldsymbol{x})^\top \boldsymbol{D}_i^\top \boldsymbol{K}_T^{-1} \boldsymbol{D}_j \boldsymbol{k}_{j,T}(\boldsymbol{x}) g_j(\boldsymbol{x}) \\
&= k(\boldsymbol{x}, \boldsymbol{x}) - \sum_{i,j} \boldsymbol{z}_i^\top \boldsymbol{K}_T^{-1} \boldsymbol{z}_j \qquad \text{(E.3)} \\
&= k(\boldsymbol{x}, \boldsymbol{x}) - \sum_{i,j} \boldsymbol{z}_i^\top (\sum_{l} \boldsymbol{D}_l \boldsymbol{K}_{l,T} \boldsymbol{D}_l)^{-1} \boldsymbol{z}_j \qquad \text{(E.4)}
\end{aligned}
$$

with $\boldsymbol{z}_i = \boldsymbol{D}_i \boldsymbol{k}_{j,T} g_j(\boldsymbol{x}) \in \mathbb{R}^n$ and equation (E.3) to (E.4) is coming from:

$$
\begin{aligned}
\boldsymbol{K}_T &= [k(\boldsymbol{x}, \boldsymbol{x}')]_{\boldsymbol{x}, \boldsymbol{x}' \in A_T} + \text{diag}([\sigma^2(\boldsymbol{x}_t)]_{t \in [T]}) \qquad \text{(E.5)} \\
&= \sum_{j=1}^{J} [g_j(\boldsymbol{x}) k_j(\boldsymbol{x}, \boldsymbol{x}') g_j(\boldsymbol{x}')]_{\boldsymbol{x}, \boldsymbol{x}' \in A_T} \\
&\qquad + \text{diag}([g_j^2(\boldsymbol{x}_t) \sigma_j^2(\boldsymbol{x}_t)]_{t \in [T]}) \qquad \text{(E.6)} \\
&= \sum_{j} \boldsymbol{D}_j ([k_j(\boldsymbol{x}, \boldsymbol{x}')]_{\boldsymbol{x}, \boldsymbol{x}' \in A_T} + \text{diag}([\sigma_j^2(\boldsymbol{x}_t)]_{t \in [T]})) \boldsymbol{D}_j \\
&= \sum_{j} \boldsymbol{D}_j \boldsymbol{K}_{j,T} \boldsymbol{D}_j
\end{aligned}
$$

where the first kernel term from Equation (E.5) to (E.6) is derived by from definition. And in the latter term, from the decomposition assumption (7.1), the noise variance $\sigma^2(\boldsymbol{x})$ of the target function f at point $\boldsymbol{x}$ is the cumulative variance of the noise variance $\sigma_j^2(\boldsymbol{x})$ of each individual function f_j,

i.e.

$$\sigma^2(\boldsymbol{x}) = \sum_{j=1}^{J} g_j^2(\boldsymbol{x})\sigma_j^2(\boldsymbol{x}) \; \forall \boldsymbol{x} \in \mathcal{X}, j \in [J]$$

which explains the derivation from Equation E.5 to E.6. □

Proposition 5. *The variance returned by Algorithm 6 is*

$$\sigma^2_{T,decomp}(\boldsymbol{x}) = k(\boldsymbol{x},\boldsymbol{x}) - \sum_l \boldsymbol{z}_l^\top (\boldsymbol{D}_l \boldsymbol{K}_{l,T} \boldsymbol{D}_l)^{-1} \boldsymbol{z}_l \tag{7.7}$$

Proof of Proposition 5. In Algorithm 6, it runs GP regression to each function $f_j(\boldsymbol{x})$ respectively. We can compute the corresponding posterior covariance function $k_{j,T}$ by:

$$k_{j,T}(\boldsymbol{x},\boldsymbol{x}') = k_j(\boldsymbol{x},\boldsymbol{x}') - \boldsymbol{k}_{j,T}(\boldsymbol{x})^\top \boldsymbol{K}_{j,T}^{-1} \boldsymbol{k}_{j,T}(\boldsymbol{x})$$

By Algorithm 6, the synthetic covariance of the target function $f(\boldsymbol{x})$ is:

$$k_{T,\text{decomp}}(\boldsymbol{x},\boldsymbol{x}') = \sum_{j=1}^{J} g_j(\boldsymbol{x}) k_{j,T}(\boldsymbol{x},\boldsymbol{x}') g_j(\boldsymbol{x}')$$

$$\begin{aligned}
&\sigma^2_{T,\text{decomp}}(\boldsymbol{x}) = k_{T,\text{decomp}}(\boldsymbol{x},\boldsymbol{x}) \\
&= \sum_{j=1}^{J} g_j(\boldsymbol{x}) k_{j,T}(\boldsymbol{x},\boldsymbol{x}) g_j(\boldsymbol{x}) \\
&= \sum_{j=1}^{J} g_j(\boldsymbol{x}) k_j(\boldsymbol{x},\boldsymbol{x}) g_j(\boldsymbol{x}) - \sum_{j=1}^{J} g_j(\boldsymbol{x}) \boldsymbol{k}_{j,T}(\boldsymbol{x})^\top \boldsymbol{K}_{j,T}^{-1} \boldsymbol{k}_{j,T}(\boldsymbol{x}) g_j(\boldsymbol{x}) \\
&= k(\boldsymbol{x},\boldsymbol{x}) - \sum_j \boldsymbol{z}_j^\top \boldsymbol{D}_j^{-1} \boldsymbol{K}_{j,T}^{-1} \boldsymbol{D}_j^{-1} \boldsymbol{z}_j \\
&= k(\boldsymbol{x},\boldsymbol{x}) - \sum_j \boldsymbol{z}_j^\top (\boldsymbol{D}_j \boldsymbol{K}_{j,T} \boldsymbol{D}_j)^{-1} \boldsymbol{z}_j
\end{aligned}$$

□

E.2 Proof of Theorem 8

Theorem 8. *The variance provided by decomposed Gaussian process regression (Algorithm 6) is less than or equal to the variance provided by Gaussian process regression (Algorithm 5), which implies the uncertainty by using decomposed Gaussian process regression is smaller.*

Proof. If we write $\boldsymbol{B}_l = \boldsymbol{D}_l \boldsymbol{K}_{l,T} \boldsymbol{D}_l$, $\boldsymbol{B}$ is positive definite since it is the multiplication of positive definite matrix $\boldsymbol{K}_{l,T}$ and two $\boldsymbol{D}_l$ identical diagonal matrices. Using Proposition 4 and 5, the difference between Equation (7.6) and (7.7) can be written as:

$$
\begin{aligned}
&\sigma^2_{T,\text{entire}}(\boldsymbol{x}) - \sigma^2_{T,\text{decomp}}(\boldsymbol{x}) \\
&= \sum_l \boldsymbol{z}_l^\top (\boldsymbol{D}_l \boldsymbol{K}_{l,T} \boldsymbol{D}_l)^{-1} \boldsymbol{z}_l - \sum_{i,j} \boldsymbol{z}_i^\top (\sum_l \boldsymbol{D}_l \boldsymbol{K}_{l,T} \boldsymbol{D}_l)^{-1} \boldsymbol{z}_j \\
&= \sum_l \boldsymbol{z}_l^\top \boldsymbol{B}_l^{-1} \boldsymbol{z}_l - \sum_{i,j} \boldsymbol{z}_i^\top (\sum_l \boldsymbol{B}_l)^{-1} \boldsymbol{z}_j \\
&= \sum_l \boldsymbol{z}_l^\top \boldsymbol{B}_l^{-1} \boldsymbol{z}_l - J(\frac{\sum_i \boldsymbol{z}_i}{J})^\top (\frac{\sum_l \boldsymbol{B}_l}{J})^{-1} (\frac{\sum_i \boldsymbol{z}_i}{J}) \\
&= \sum_l h(\boldsymbol{B}_l, \boldsymbol{z}_l) - Jh(\bar{\boldsymbol{B}}, \bar{\boldsymbol{z}}) \geq 0
\end{aligned}
$$

where $\bar{\boldsymbol{B}} = \frac{\sum_i \boldsymbol{B}_i}{J}$ and $\bar{\boldsymbol{z}} = \frac{\sum_i \boldsymbol{z}_i}{J}$ are the average value. The last inequality comes from Jensen inequality and Lemma 1, which says the matrix-fractional function h is convex. □

E.3 Proof of Theorem 9

In order to prove this, we follow the similar techniques as GPUCB[292], which is illustrated as follows:

Lemma 3 (Modified version of Lemma 5.1 from Srinivas et al.). *Given $f(\boldsymbol{x}) = \sum_{j=1}^{J} g_j(\boldsymbol{x}) f_j(\boldsymbol{x})$ (Definition 7), deterministic known functions g_j and unknown $f_j \sim GP(0, k_j(\boldsymbol{x}, \boldsymbol{x}'))$, pick $\delta \in (0,1)$ and set $\beta_t = 2\log(|\mathcal{X}|\pi_t/\delta)$, where $\sum_{t\geq 1} \pi_t^{-1} = 1, \pi_t > 0$. Then, the $\mu_{t-1}(\boldsymbol{x}), \sigma_{t-1}(\boldsymbol{x})$ returned by Algorithm 7 satisfy:*

$$
|f(\boldsymbol{x}) - \mu_{t-1}(\boldsymbol{x})| \leq \beta_t^{1/2} \sigma_{t-1}(\boldsymbol{x}) \qquad \forall \boldsymbol{x} \in \mathcal{X}, t \geq 1
$$

with probability $1 - \delta$.

Proof. Fix $t \geq 1$ and $\boldsymbol{x} \in \mathcal{X}$, Conditioned on sampled points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}\}$ and sampled values $\{y_{1,j}, \ldots, y_{t-1,j} \forall j \in [J]\}$, the Bayesian property of decomposed GP regression (Algorithm 6) implies that the function value at point $\boldsymbol{x}$ forms a Gaussian distribution with mean $\mu_{t-1}(\boldsymbol{x})$ and variance $\sigma^2_{t-1}(\boldsymbol{x})$, i.e. $f(\boldsymbol{x}) \sim N(\mu_{t-1}(\boldsymbol{x}), \sigma^2_{t-1}(\boldsymbol{x}))$. Now, if $r \sim N(0,1)$, then

$$
Pr\{r > c\} = e^{-c^2/2}(2\pi)^{-1/2} \int e^{-(r-c)^2/2 - c(r-c)} dr \leq e^{-c^2/2} Pr\{r > 0\} = (1/2)e^{-c^2/2} \tag{E.7}
$$

for $c > 0$, since $e^{-c(r-c)} \leq 1$ for $r \geq c$. Therefore, $Pr\{|f(\boldsymbol{x}) - \mu_{t-1}(\boldsymbol{x})| > \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x})\} \leq e^{-\beta_t^{1/2}}$, using $r = (f(\boldsymbol{x}) - \mu_{t-1}(\boldsymbol{x}))/\sigma_{t-1}(\boldsymbol{x})$ and $c = \beta_t^{1/2}$. Then apply the union bound to all $\boldsymbol{x} \in \mathcal{X}$:

$$|f(\boldsymbol{x}) - \mu_{t-1}(\boldsymbol{x})| \leq \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}) \; \forall \boldsymbol{x} \in \mathcal{X}$$

holds with probability $\geq 1 - |\mathcal{X}|e^{-\beta_t^{1/2}}$. Choosing $|\mathcal{X}|e^{-\beta_t^{1/2}} = \delta/\pi_t$ and using the union bound for $t \in \mathbb{N}$, the statement holds. For example, we can use $\pi_t = \pi^2 t^2/6$. □

The proof is almost the same as Theorem 5.1 in Srinivas et al. [292] except the Bayesian property of decomposed Gaussian process, where the Bayesian property of decomposed Gaussian process can be gotten from the Bayesian property of each individual function f_j and the linear combination of Gaussian distributions is still a Gaussian distribution, which implies the posterior belief after performing decomposed GP regression at a given point $\boldsymbol{x}$ still form a Gaussian distribution with composed mean and variance.

Lemma 4 (Modified version of Lemma 5.2 from Srinivas et al. [292]). *Fix $t \geq 1$. If $|f(\boldsymbol{x}) - \mu_{t-1}(\boldsymbol{x})| \leq \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}$, then the regret r_t is bounded by $2\beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}_t)$, where $\boldsymbol{x}_t$ is the t-th choice of Algorithm 7.*

Proof. By definition of $\boldsymbol{x}_t$: $\mu_{t-1}(\boldsymbol{x}_t) + \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}_t) \geq \mu_{t-1}(\boldsymbol{x}^*) + \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}^*) \geq f(\boldsymbol{x}^*)$. Therefore,

$$r_t = f(\boldsymbol{x}^*) - f(\boldsymbol{x}_t) \leq \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}_t) + \mu_{t-1}(\boldsymbol{x}_t) - f(\boldsymbol{x}_t) \leq 2\beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}_t)$$

□

Lemma 5 (Modified version of Lemma 5.3 from Srinivas et al. [292]). *The information gain for the points selected can be expressed in terms of the predictive variances. If $\boldsymbol{f}_{j,T} = \{f_j(\boldsymbol{x}_t)\}_{t\in[T]} \in \mathbb{R}^T$ and $\boldsymbol{y}_{j,T} = \{y_{j,t}\}_{t\in[T]} \in \mathbb{R}^T$:*

$$I(\boldsymbol{y}_{j,T} : \boldsymbol{f}_{j,T}) = \frac{1}{2}\sum_{t=1}^{T} \log(1 + \sigma^{-2}\sigma_{j,t-1}^2(\boldsymbol{x}_t))$$

where $f_j(\boldsymbol{x}_t), y_{j,t}, \sigma_{j,t-1}^2$ follow the definition and derivation in Algorithm 7.

Proof. Directly follow by replacing all the f, y, σ by f_j, y_j, σ_j in the proof of Theorem 5.3 from Srinivas et al. [292]. □

Theorem 9. *Let $\delta \in (0,1)$ and $\beta_t = 2\log(|\mathcal{X}|t^2\pi^2/6\delta)$. Running decomposed GP-UCB (Algorithm 7) for a composed sample $f(\boldsymbol{x}) = \sum_{j=1} g_j(\boldsymbol{x})f_j(\boldsymbol{x})$ with bounded variance $k_j(\boldsymbol{x},\boldsymbol{x}) \leq 1$ and each $f_j \sim GP(0, k_j(\boldsymbol{x},\boldsymbol{x}'))$, we obtain a regret bound of $\mathcal{O}(\sqrt{T\log|\mathcal{X}|\sum_{j=1}^{J} B_j^2\gamma_{j,T}})$ with high probability,*

where $B_j = \max_{\boldsymbol{x} \in \mathcal{X}} |g_j(\boldsymbol{x})|$. *Precisely,*

$$Pr\Big\{R_T \leq \sqrt{C_1 T \beta_T \sum_{j=1}^{J} B_j^2 \gamma_{j,T}} \, \forall T \geq 1\Big\} \geq 1 - \delta \tag{7.8}$$

where $C_1 = 8/\log(1+\sigma^{-2})$ *with noise variance* σ^2.

Proof. According to Lemma 5, we can take advantage of the individual information gain of each $f_j(\boldsymbol{x})$, which is

$$\begin{aligned} I_j(y_{j,T}; f_j) &= \frac{1}{2} \sum_{t=1}^{T} \log(1 + \sigma^{-2} \sigma_{j,t-1}^2(\boldsymbol{x}_t)) \\ \gamma_{j,T} &= \max I_j(y_{j,T}; f_j) \end{aligned}$$

Besides, we can also bound the total regret by the individual information gains as following:

$$\begin{aligned} \sum_{j=1}^{J} B_j^2 I_j(y_{j,T}; f_j) &= \frac{1}{2} \sum_{j=1}^{J} B_j^2 \sum_{t=1}^{T} \log(1 + \sigma^{-2} \sigma_{j,t-1}^2(\boldsymbol{x}_t)) \\ &\geq \frac{1}{2} \sum_{j=1}^{J} B_j^2 \sum_{t=1}^{T} C_2^{-1} \sigma^{-2} \sigma_{j,t-1}^2(\boldsymbol{x}_t) \\ &\geq \frac{1}{2} C_2^{-1} \sigma^{-2} \sum_{j=1}^{J} \sum_{t=1}^{T} g_j^2(\boldsymbol{x}_t) \sigma_{j,t-1}^2(\boldsymbol{x}_t) \\ &\geq \frac{1}{2} C_2^{-1} \sigma^{-2} \sum_{t=1}^{T} \frac{r_t^2}{4\beta_t} \\ &\geq \frac{C_2^{-1} \sigma^{-2}}{8\beta_T} \sum_{t=1}^{T} r_t^2 \end{aligned}$$

where $C_2 = \sigma^{-2}/\log(1+\sigma^{-2}) \geq 1, s^2 \leq C_2 \log(1+s^2)$ for $s \in [0, \sigma^{-2}]$ and $\sigma^{-2}\sigma_{j,t-1}^2(\boldsymbol{x}_t) \leq \sigma^{-2} k_j(\boldsymbol{x}_t, \boldsymbol{x}_t) \leq \sigma^{-2}$. Let $C_1 = 8\sigma^2 C_2 = 8/\log(1+\sigma^{-2})$. Applying Cauchy inequality gives us:

$$C_1 \beta_T T \sum_{j=1}^{J} B_j^2 I_j(y_{j,T}; f_j) \geq (\sum_{t=1}^{T} r_t)^2 = R_T^2$$

which implies a similar upper bound

$$R_T \leq \sqrt{C_1 T \beta_T \sum_{j=1}^{J} B_j^2 \gamma_{j,T}}$$

□

E.4 Proof of Theorem 10

All the proofs in Theorem 9 apply except Lemma 3. Since the decomposition here is non-linear, therefore the composition of outcomes of Gaussian processes is no longer an outcome of Gaussian process, which prohibits us to have a nice Gaussian process property: function value $f(\boldsymbol{x})$ does not form a Gaussian distribution. Due to the non-linearity, the distribution gets distorted, losing its original form with Gaussian distribution. Fortunately, if the partial derivatives of function $g : \mathbb{R}^J \to \mathbb{R}$ (Definition 8) are bounded, then we can still perform a similar estimation and bound the distribution by a larger Gaussian distribution, which enables us to have a similar result.

Lemma 6 (General Version with Definition 8 and Algorithm 8)**.** *Given $f(\boldsymbol{x}) = g(f_1(\boldsymbol{x}), ..., f_J(\boldsymbol{x}))$ (Definition 8), deterministic known functions g and unknown $f_j \sim GP(0, k_j(\boldsymbol{x}, \boldsymbol{x}'))$, pick $\delta \in (0,1)$ and set $\beta_t = 2\log(|\mathcal{X}|J\pi_t/\delta)$, where $\sum_{t\geq 1} \pi_t^{-1} = 1, \pi_t > 0$. Further assume the function g has bounded partial derivatives $B_j = \max_{\boldsymbol{x}\in\mathcal{X}} |\nabla_j g(\boldsymbol{x})| \; \forall j \in [J]$. Then, the $\mu_{t-1}(\boldsymbol{x}), \sigma_{t-1}(\boldsymbol{x})$ returned by Algorithm 8 satisfy:*

$$|f(\boldsymbol{x}) - \mu_{t-1}(\boldsymbol{x})| \leq \beta_t^{1/2} \sigma_{t-1}(\boldsymbol{x}) \forall \boldsymbol{x} \in \mathcal{X}, t \geq 1$$

with probability $1 - \delta$.

Proof. The main problem here is the posterior distribution of $f(\boldsymbol{x})$ is not a Gaussian distribution. But fortunately, the posterior distribution of each $f_j(\boldsymbol{x})$ is still a Gaussian distribution with mean $\mu_{j,t-1}(\boldsymbol{x})$ and variance $\sigma_{j,t-1}^2(\boldsymbol{x})$ for any given $\boldsymbol{x} \in \mathcal{X}$. Then,

$$\begin{aligned} |f(\boldsymbol{x}) - \mu_{t-1}(\boldsymbol{x})| &= |g(f_1(\boldsymbol{x}), ..., f_J(\boldsymbol{x})) - g(\mu_{1,t-1}(\boldsymbol{x}), ..., \mu_{J,t-1}(\boldsymbol{x}))| \\ &\leq \sum_{j=1}^{J} B_j |f_j(\boldsymbol{x}) - \mu_{j,t-1}(\boldsymbol{x})| \end{aligned} \tag{E.8}$$

Applying the same argument in Lemma 3 to function f_j:

$$Pr\{|f_j(\boldsymbol{x}) - \mu_{j,t-1}(\boldsymbol{x})| > \beta_t^{1/2} \sigma_{j,t-1}(\boldsymbol{x})\} \leq e^{-\beta_t^{1/2}}$$

Then applying the union bound on $j \in [J]$, we get

$$
\begin{aligned}
|f(\boldsymbol{x}) - \mu_{t-1}(\boldsymbol{x})| &\leq \sum_{j=1}^{J} B_j |f_j(\boldsymbol{x}) - \mu_{j,t-1}(\boldsymbol{x})| \\
&\leq \sum_{j=1}^{J} B_j \beta_t^{1/2} \sigma_{j,t-1}(\boldsymbol{x}) \\
&\leq \beta_t^{1/2} \sqrt{J(\sum_{j=1}^{J} B_j^2 \sigma_{j,t-1}^2(\boldsymbol{x}))} \\
&= \beta_t^{1/2} \sigma_{t-1}(\boldsymbol{x})
\end{aligned}
$$

with probability $1 - Je^{-\beta_t^{1/2}}$, where the last inequality is from Cauchy's inequality. Then apply union bound again to all $\boldsymbol{x} \in \mathcal{X}$, the above inequality yields:

$$
|f(\boldsymbol{x}) - \mu_{t-1}(\boldsymbol{x})| \leq \beta_t^{1/2} \sigma_{t-1}(\boldsymbol{x}) \; \forall \boldsymbol{x} \in \mathcal{X}
$$

with probability $1 - |\mathcal{X}| J e^{-\beta_t^{1/2}}$. Choosing $|\mathcal{X}| J e^{-\beta_t^{1/2}} = \delta / \pi_t$ and using the union bound for $t \in \mathbb{N}$, the statement holds, i.e. $\beta_t = 2 \log(|\mathcal{X}| J \pi_t / \delta)$. Specifically, if we choose $\pi_t = \pi^2 t^2 / 6$, then it implies $\beta_t = 2 \log(|\mathcal{X}| J t^2 \pi^2 / 6\delta)$. □

Theorem 10. *By running generalized decomposed GP-UCB with $\beta_t = 2 \log(|\mathcal{X}| J t^2 \pi^2 / 6\delta)$ for a composed sample $f(\boldsymbol{x}) = g(f_1(\boldsymbol{x}), \ldots, f_J(\boldsymbol{x}))$ of GPs with bounded variance $k_j(\boldsymbol{x}, \boldsymbol{x}) \leq 1$ and each $f_j \sim GP(0, k_j(\boldsymbol{x}, \boldsymbol{x}'))$. we obtain a regret bound of $\mathcal{O}(\sqrt{T \log |\mathcal{X}| \sum_{j=1}^{J} B_j^2 \gamma_{j,T}})$ with high probability, where $B_j = \max_{\boldsymbol{x} \in \mathcal{X}} |\nabla_j g(\boldsymbol{x})|$. Precisely,*

$$
Pr\{R_T \leq \sqrt{C_1 T \beta_T \sum_{j=1}^{J} B_j^2 \gamma_{j,T}} \; \forall T \geq 1\} \geq 1 - \delta \tag{7.10}
$$

where $C_1 = 8 / \log(1 + \sigma^{-2})$ with noise variance σ^2.

Proof. Directly follow by the same proofs of Theorem 9 with Lemma 4, Lemma 5, and Lemma 6. □

Remark 1. *In inequality* (E.8)*, if we write $Z_j = |f_j(\boldsymbol{x}) - \mu_{j,t-1}(\boldsymbol{x})|$, where $f_j(\boldsymbol{x}) - \mu_{j,t-1}(\boldsymbol{x})$ is sampled from a normal distribution with* 0 *mean and $\sigma_{j,t-1}^2(\boldsymbol{x})$ (due to Gaussian process property). Then this Z_j is a random variable drawn from a half-normal distribution with parameter $\sigma_j(\boldsymbol{x})$ (no longer the variance here).*

The summation of half-normal distributions can still be computed and bounded by a similar inequality like inequality (E.7). *This can provide a constant ratio of improvement to the β_t exploration parameter, thus the regret bound as well. However it does not change the order of regret and sample complexity. Therefore we are not going to cover this here.*

F
Appendix to Chapter 8

F.1 Notation of Chapter 8

All the notations used in the problem statement, restless multi-armed bandits, and regret analysis are shown in Table F.1 and Table F.2.

Problem instantiation	
Symbol	**Definition**
K	Budget in each timestep
N	Number of arms. Each arm indexed by $i \in [N]$
t	Episode
T	Number of episodes
h	Timestep within a single episode
H	Horizon length for a single episode
γ	Discount factor, with $\gamma \in (0, 1)$

Table F.1: List of common notations in the online restless multi-armed bandit problem

Restless bandit notation	
Symbol	**Definition**
$\boldsymbol{P}$	Set of transition probabilities across all arms, with P_i as transitions for a single arm
$\boldsymbol{P}^\star$	True transition probabilities
$\mathcal{S}$	Set of finitely many possible states with $\lvert\mathcal{S}\rvert = M$ possible states
$\boldsymbol{s}_h$	State of the RMAB instance at timestep h, with $\boldsymbol{s}_h \in \mathcal{S}^N$ and initial state $\boldsymbol{s}_{\text{init}}$
$s_{h,i}$	State of arm $i \in [N]$ at timestep h
$\mathcal{A}$	Set of possible actions. We consider $\{0, 1\}$
$\boldsymbol{a}_h$	Action at time h, with $\boldsymbol{a}_h \in \mathcal{A}^N$
$a_{h,i}$	Action taken on arm i at timestep h
R	Given reward function as a function of the state and action $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.
$\pi^{(t)}$	Learner's policy in episode t, where $\pi^{(t)} : \mathcal{S}^N \to \mathcal{A}^N$
$\pi^\star$	The optimal policy that maximizes the total future reward.
$\mathcal{P}_m$	The optimization problem defined to maximize the optimistic Whittle index value.
$\mathcal{P}_V$	The optimization problem defined to maximize the optimistic future value.
$Q^{m_i}(s, a)$	Q-value in Bellman equation. The Q-value is defined as the future value associated to the current state and action.
$R(s_{h,i}, a_{h,i})$	Reward from arm i at timestep h with action $a_{h,i}$
$U_\pi^{\boldsymbol{P},\lambda}(\boldsymbol{s}_1)$	Lagrangian relaxation of learner's objective, with optimal value $U_\star^{\boldsymbol{P},\lambda}$
$V_{\pi_i}^{P_i,\lambda}(s_{1,i})$	Value for being in state s_i
λ	Global penalty for taking action $a = 1$
β_i	Whittle index penalty for arm i
$W_i(P_i, s_i)$	Whittle index of arm i with transitions P_i and state s_i
$\mathcal{D}$	Dataset of historical transitions

Table F.2: List of common notations in the online restless multi-armed bandit regret analysis

F.2 Societal Impacts

Restless bandits have been increasingly applied to socially impactful problems including healthcare and energy distribution. In these settings, we would likely not know the transition dynamics in advance, particularly if we are working with a new patient population (for healthcare) or new residential community (for energy). Even past work on streaming bandits[209] which allow for new mothers

to enroll over time assume that the transition probabilities are fully known in advance, which is not realistic. Our UCWhittle approach enabling online learning for RMABs has the potential to greatly broaden the applicability of RMABs for social impact, particularly as our theoretical results guarantee limited regret.

F.3 Limitations

One challenge with our UCWhittle approach is that online learning often converges slower than offline learning that reuses all the data to train for many epochs. In order to accommodate new data coming in, online learning approaches often take a single update when each new data arrives. In contrast, offline learning can iterate through the same data for many times, which allows offline learning approaches to fit the data repeatedly. Therefore, online learning approaches often require more data to reach the same performance as offline learning approaches.

However, this slower learning behavior also allows online learning approaches to be less biased to the existing dataset. Online learning approaches are incentivized to explore and update data that is less queried previously, which also encourages exploring underrepresented groups. This property encourages the exploration process and reduce bias to the learned model. This is particularly important when there are features involved in the learning process. Online learning approaches are able to explore unseen features more, while offline learning approaches often rely on extrapolation and are unable to handle unseen features. Our work further extends research in online learning in RMABs, which also helps explore more possibility to accommodate new data and new features that are unseen in the existing dataset.

F.4 Full Proofs

F.4.1 Confidence Bound

Proposition 6. *Given $\delta > 0$ and $t \geq 1$, we have:* $\Pr\left(\boldsymbol{P}^\star \in \boldsymbol{B}^{(t)}\right) \geq 1 - \frac{\delta}{t^4}$.

Proof. Generally, the L1-deviation of the true distribution and the empirical distribution over m distinct events from n samples is bounded according to[335] by:

$$\Pr(\|\hat{p} - p\|_1 \geq \varepsilon) \leq (2^m - 2)\exp^{(-\frac{n\varepsilon^2}{2})} \tag{F.1}$$

This result can be applied to our case to compare $P_i^{(t)}(s, a, \cdot) \in \mathbb{R}^{|S|}$ with $P^\star(s, a, \cdot) \in \mathbb{R}^{|S|}$ for every state s and action a. We have:

$$\Pr\left(\left\|P_i^{(t)}(s, a, \cdot) - P^\star(s, a, \cdot)\right\|_1 \geq \varepsilon\right) \leq \left(2^{|S|} - 2\right)\exp^{\left(-\frac{n\varepsilon^2}{2}\right)} \tag{F.2}$$

By choosing $\varepsilon = \sqrt{\frac{2}{n}\log\left(2^{|S|}|S||A|N\frac{t^4}{\delta}\right)} \leq \sqrt{\frac{2|S|}{n}\log\left(2|S||A|N\frac{t^4}{\delta}\right)}$, we have:

$$\Pr\left(\left\|P_i^{(t)}(s,a,\cdot) - P^\star(s,a,\cdot)\right\|_1 \geq \sqrt{\frac{2|S|}{n}\log\left(2|S||A|N\frac{t^4}{\delta}\right)}\right) \leq 2^{|S|}\exp^{-\log\left(2^{|S|}|S||A|N\frac{t^4}{\delta}\right)}$$
$$= \frac{\delta}{|S||A|Nt^4} \tag{F.3}$$

Set $n = \max\{1, N_i^{(t)}(s,a)\}$ for each pair of (s,a). Taking union bound over all states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, and arms $i \in [N]$ yields:

$$\Pr\left(\boldsymbol{P}^\star \notin \boldsymbol{B}^{(t)}\right) \leq \frac{\delta}{t^4} \quad \Longrightarrow \quad \Pr\left(\boldsymbol{P}^\star \in \boldsymbol{B}^{(t)}\right) \geq 1 - \frac{\delta}{t^4} \tag{F.4}$$

□

F.4.2 Regret Decomposition

Theorem 12 (Per-episode regret decomposition in the fully observable setting)**.** *For an arm i, fix $P_i^{(t)}$, $P_i^\star$, λ, and the initial state $s_{1,i}$. We have:*

$$V^{P_i^{(t)},\lambda}_{\pi_i^{(t)}}(s_{1,i}) - V^{P_i^\star,\lambda}_{\pi_i^{(t)}}(s_{1,i}) = \mathbb{E}_{P_i^\star,\pi_i^{(t)}}\left[\sum_{h=1}^{\infty}\gamma^{h-1}\left(\mathcal{T}^{P_i^{(t)}}_{\pi_i^{(t)}} - \mathcal{T}^{P_i^\star}_{\pi_i^{(t)}}\right)V^{P_i^{(t)},\lambda}_{\pi_i^{(t)}}(s_{h,i})\right]. \tag{8.11}$$

Proof. Since the value function is a fixed point to the corresponding Bellman operator, we have:

$$\begin{aligned} V^{P_i^{(t)}}_{\pi_i^{(t)}}(s_{1,i}) - V^{P_i^\star}_{\pi_i^{(t)}}(s_{1,i}) &= \left(\mathcal{T}^{P_i^{(t)}}_{\pi_i^{(t)}}V^{P_i^{(t)}}_{\pi_i^{(t)}} - \mathcal{T}^{P_i^\star}_{\pi_i^{(t)}}V^{P_i^\star}_{\pi_i^{(t)}}\right)(s_{1,i}) \\ &= \left(\mathcal{T}^{P_i^{(t)}}_{\pi_i^{(t)}} - \mathcal{T}^{P_i^\star}_{\pi_i^{(t)}}\right)V^{P_i^{(t)}}_{\pi_i^{(t)}}(s_{1,i}) + \mathcal{T}^{P_i^\star}_{\pi_i^{(t)}}\left(V^{P_i^{(t)}}_{\pi_i^{(t)}} - V^{P_i^\star}_{\pi_i^{(t)}}\right)(s_{1,i}) \end{aligned} \tag{F.5}$$

where the second term in Equation (F.5) can be further expanded by the Bellman operator:

$$\begin{aligned} \mathcal{T}^{P_i^\star}_{\pi_i^{(t)}}(V^{P_i^{(t)}}_{\pi_i^{(t)}} - V^{P_i^\star}_{\pi_i^{(t)}})(s_{1,i}) &= \mathbb{E}_{a\sim\pi_i^{(t)}}\left[\gamma\sum_{s'\in\mathcal{S}}P_i^\star(s_{1,i},a,s')(V^{P_i^{(t)}}_{\pi_i^{(t)}}(s') - V^{P_i^\star}_{\pi_i^{(t)}}(s'))\right] \\ &= \gamma\mathbb{E}_{s_{2,i}\sim P_i^\star,\pi_i^{(t)}}\left[V^{P_i^{(t)}}_{\pi_i^{(t)}}(s_{2,i}) - V^{P_i^\star}_{\pi_i^{(t)}}(s_{2,i})\right] \end{aligned} \tag{F.6}$$

We can repeatedly apply the decomposition process in Equation (F.5) to the value function difference in Equation (F.6) to get Equation (8.11), which concludes the proof. □

F.4.3 Regret Bound with Given Penalty

Theorem 13. *Assume the penalty term $\lambda^{(t)} = \lambda$ is given and the RMAB instance is ε-ergodicity after H timesteps. Then with probability $1-\delta$, the cumulative regret in T episodes is:*

$$Reg_{\lambda}(t) \leq O\left(\frac{1}{\varepsilon}|S||A|^{\frac{1}{2}}NH\sqrt{T\log T}\right) . \tag{8.12}$$

Proof. We can write

$$\begin{aligned}\mathrm{Reg}(T) &= \sum_{t=1}^{T}\mathrm{Reg}^{(t)} = \sum_{t=1}^{T}\left(\mathrm{Reg}^{(t)}1_{\boldsymbol{P}^{\star}\notin\boldsymbol{B}^{(t)}} + \mathrm{Reg}^{(t)}1_{\boldsymbol{P}^{\star}\in\boldsymbol{B}^{(t)}}\right)\\ &= \sum_{t=1}^{T}\mathrm{Reg}^{(t)}1_{\boldsymbol{P}^{\star}\notin\boldsymbol{B}^{(t)}} + \sum_{t=1}^{T}\mathrm{Reg}^{(t)}1_{\boldsymbol{P}^{\star}\in\boldsymbol{B}^{(t)}}\end{aligned} \tag{F.7}$$

We will analyze both terms separately and combine them together in the end.

Regret when the confidence bounds do not hold

$$\begin{aligned}\sum_{t=1}^{T}\mathrm{Reg}^{(t)}1_{\boldsymbol{P}^{\star}\notin\boldsymbol{B}^{(t)}} &= \sum_{t=1}^{\sqrt{T}}\mathrm{Reg}^{(t)}1_{\boldsymbol{P}^{\star}\notin\boldsymbol{B}^{(t)}} + \sum_{t=\sqrt{T}+1}^{T}\mathrm{Reg}^{(t)}1_{\boldsymbol{P}^{\star}\notin\boldsymbol{B}^{(t)}}\\ &\leq \frac{NR_{\max}}{1-\gamma}\sqrt{T} + \sum_{t=\sqrt{T}+1}^{T}\mathrm{Reg}^{(t)}1_{\boldsymbol{P}^{\star}\notin\boldsymbol{B}^{(t)}}\end{aligned} \tag{F.8}$$

where we use the trivial upper bound of the individual regret $\mathrm{Reg}^{(t)} \leq \frac{NR_{\max}}{1-\gamma}$ for all t, where $R_{\max}$ is the maximal reward per time step.

Notice that the second term vanishes with probability:

$$\begin{aligned}\Pr\left(\left\{\boldsymbol{P}^{\star}\in\boldsymbol{B}^{(t)}\ \forall\sqrt{T}\leq t\leq T\right\}\right) &\geq 1 - \sum_{\sqrt{T}\leq t\leq T}\Pr\left(\left\{\boldsymbol{P}^{\star}\in\boldsymbol{B}^{(t)}\right\}\right)\\ &\geq 1 - \sum_{\sqrt{T}\leq t\leq T}\frac{\delta}{t^4}\\ &\geq 1 - \sum_{\sqrt{T}\leq t\leq T}\frac{3\delta}{t^4}\\ &\geq 1 - \int_{\sqrt{T}}^{\infty}\frac{3\delta}{t^4}dt\end{aligned}$$

$$= 1 - \frac{\delta}{T^{3/2}} \tag{F.9}$$

Therefore, the regret outside of confidence bounds is upper bounded by $O(\sqrt{T})$ with probability at least $1 - \frac{\delta}{T^{3/2}}$. We can apply union bound to all possible $T \in \mathbb{N}$, which holds with high probability:

$$1 - \sum_{T=1}^{\infty} \frac{\delta}{T^{3/2}} = 1 - O(\delta)\,. \tag{F.10}$$

Regret when the confidence bounds hold Notice that

$$\begin{aligned}
&\left(\mathcal{T}^{P_i^{(t)}}_{\pi_i^{(t)}} - \mathcal{T}^{P_i^\star}_{\pi_i^{(t)}}\right) V(s) \\
&= \mathop{\mathbb{E}}_{a\sim\pi_i^{(t)}}\left[\left(R(s,a) + \sum_{s'\in\mathcal{S}} P_i^{(t)}(s,a,s')V(s')\right) - \left(R(s,a) + \sum_{s'\in\mathcal{S}} P_i^\star(s,a,s')\right)V(s')\right] \\
&= \mathop{\mathbb{E}}_{a\sim\pi_i^{(t)}}\left[\sum_{s'\in\mathcal{S}}(P_i^{(t)}(s,a,s') - P_i^\star(s,a,s'))V(s')\right]
\end{aligned}$$

When the confidence bound holds $\boldsymbol{P}^\star \in \boldsymbol{B}^{(t)}$, we can bound the regret at round l by:

$$\begin{aligned}
\mathrm{Reg}^{(t)} &= U^{\boldsymbol{P}^{(t)}}_{\pi^{(t)}}(\boldsymbol{s}_1) - U^{\boldsymbol{P}^\star}_{\pi^{(t)}}(\boldsymbol{s}_1) \\
&= \sum_{i=1}^{N} V^{P_i^{(t)}}_{\pi_i^{(t)}}(s_{1,i}) - V^{P_i^\star}_{\pi_i^{(t)}}(s_{1,i}) \\
&= \sum_{i=1}^{N} \mathbb{E}_{P_i^\star,\pi_i^{(t)}}\left[\sum_{h=1}^{\infty}\gamma^{h-1}(\mathcal{T}^{P_i^{(t)}}_{\pi_i^{(t)}} - \mathcal{T}^{P_i^\star}_{\pi_i^{(t)}})V^{P_i^{(t)}}_{\pi_i^{(t)}}(s_{h,i})\right] \\
&= \sum_{i=1}^{N} \mathbb{E}_{P_i^\star,\pi_i^{(t)}} \sum_{h=1}^{\infty}\sum_{s'\in\mathcal{S}}\gamma^{h-1}(P_i^{(t)}(s_{h,i},a_{h,i},s') - P_i^\star(s_{h,i},a_{h,i},s'))V^{P_i^{(t)}}_{\pi_i^{(t)}}(s') \\
&\le \sum_{i=1}^{N} \mathbb{E}_{P_i^\star,\pi_i^{(t)}} \sum_{h=1}^{\infty}\gamma^{h-1}\left\|P_i^{(t)}(s_{h,i},a_{h,i},\cdot) - P_i^\star(s_{h,i},a_{h,i},\cdot)\right\|_1 V_{\max} \\
&\le 2\sum_{i=1}^{N} \mathbb{E}_{P_i^\star,\pi^{(t)}} \sum_{h=1}^{\infty}\gamma^{h-1} d_i^{(t)}(s_{h,i},a_{h,i})V_{\max}
\end{aligned} \tag{F.11}$$

Next, we split the term into regret within H horizon and the regret outside of H horizon. By applying Theorem 30 with the assumption (Assumption 4) of the H-step ergodicity ε of MDP

associated to arm i, we can bound the regret outside of H horizon by the regret at H time step:

$$
\begin{aligned}
&\mathbb{E}_{P_i^\star,\pi^{(t)}} \sum_{h=H+1}^{\infty} \gamma^{h-1} d_i^{(t)}(s_{h,i}, a_{h,i}) V_{\max} \\
&= \sum_{h=H+1}^{\infty} \gamma^{h-1} \mathbb{E}_{s_{h,i},a_{h,i}\sim P_i^\star,\pi_i^{(t)}} \left[d_i^{(t)}(s_{h,i}, a_{h,i}) V_{\max} \right] \\
&\leq \sum_{h=H+1}^{\infty} \gamma^{h-1} \frac{1}{\varepsilon} \mathbb{E}_{s_{H,i},a_{H,i}\sim P_i^\star,\pi_i^{(t)}} \left[d_i^{(t)}(s_{h,i}, a_{h,i}) V_{\max} \right] \\
&= \frac{\gamma^H}{\varepsilon(1-\gamma)} \mathbb{E}_{s_{H,i},a_{H,i}\sim P_i^\star,\pi_i^{(t)}} \left[d_i^{(t)}(s_{h,i}, a_{h,i}) V_{\max} \right] \qquad \text{(F.12)}
\end{aligned}
$$

Now, we can further bound the contribution of arm i in Equation F.11 by substituting the regret after H steps by Equation F.12 to get:

$$
\begin{aligned}
&\mathbb{E}_{P_i^\star,\pi_i^{(t)}} \sum_{h=1}^{\infty} \gamma^{h-1} d_i^{(t)}(s_{h,i}, a_{h,i}) V_{\max} \\
&\leq \mathbb{E}_{P_i^\star,\pi_i^{(t)}} \left(\sum_{h=1}^{H} \gamma^{h-1} d_i^{(t)}(s_{h,i}, a_{h,i}) + \frac{\gamma^H}{\delta(1-\gamma)} d_i^{(t)}(s_{H,i}, a_{H,i}) \right) V_{\max} \\
&\leq \left(1 + \frac{\gamma^H}{\varepsilon(1-\gamma)}\right) \mathbb{E}_{P_i^\star,\pi_i^{(t)}} \left(\sum_{h=1}^{H} d_i^{(t)}(s_{h,i}, a_{h,i}) V_{\max} \right) \\
&= \left(1 + \frac{\gamma^H}{\varepsilon(1-\gamma)}\right) \sqrt{2|S| \log(2|A|Nt)} V_{\max} \mathbb{E}_{P_i^\star,\pi_i^{(t)}} \left(\sum_{h=1}^{H} \frac{1}{\sqrt{\max\{1, N_i^{(t)}(s,a)\}}} \right) \\
&\leq \left(1 + \frac{\gamma^H}{\varepsilon(1-\gamma)}\right) \sqrt{2|S| \log(2|A|NT)} V_{\max} \mathbb{E}_{P_i^\star,\pi_i^{(t)}} \left(\sum_{s\in\mathcal{S},a\in\mathcal{A}} \frac{v_i^{(t)}(s,a)}{\sqrt{\max\{1, N_i^{(t)}(s,a)\}}} \right) \qquad \text{(F.13)}
\end{aligned}
$$

where $v_i^{(t)}(s,a)$ is a random variable denoting the number of visitations to the pair (s,a) at arm i that the policy $\pi_i^{(t)}$ visits within H steps under the transition probability $P_i^\star$.

Recall that $\sum_{j=1}^{l-1} v_i^{(j)}(s,a) = N_i^{(t)}(s,a)$. We also know that $0 \leq v_i^{(j)}(s,a) \leq H$. Applying Lemma 7, we have:

$$
\sum_{t=1}^{T} \frac{v_i^{(t)}(s,a)}{\sqrt{\max\{1, N_i^{(t)}(s,a)\}}} \leq \left(\sqrt{H+1} + 1\right) \sqrt{N_i^{(t)}(s,a)} \qquad \text{(F.14)}
$$

Taking summation over all the (s, a) pairs and applying Jensen inequality give us:

$$\begin{aligned}
&\left(\sqrt{H+1}+1\right) \sum_{s\in\mathcal{S},a\in\mathcal{A}} \sqrt{N_i^{(t)}(s,a)} \\
&\leq \left(\sqrt{H+1}+1\right) |S||A| \sqrt{\frac{\sum_{s\in\mathcal{S},a\in\mathcal{A}} N_i^{(t)}(s,a)}{|S||A|}} \\
&= \left(\sqrt{H+1}+1\right) \sqrt{|S||A|TH} \qquad \text{(F.15)}
\end{aligned}$$

where $\sum_{s\in\mathcal{S},a\in\mathcal{A}} N_i^{(t)}(s,a) = TH$ is the total number of state-action pairs visited in T rounds.

Lastly, using the trivial upper bound $V_{\max} \leq \frac{R_{\max}}{1-\gamma}$, we can take summation over the regret from all T rounds. This give us:

$$\begin{aligned}
&\sum_{t=1}^{T} \mathrm{Reg}^{(t)} 1_{\boldsymbol{P}^\star \in \boldsymbol{B}^{(t)}} \qquad \text{(F.16)} \\
&\leq \sum_{i=1}^{N} 2\left(1+\frac{\gamma^H}{\varepsilon(1-\gamma)}\right) \sqrt{2|S|\log(2|A|NT)} V_{\max} \left(\sqrt{H+1}+1\right) \sqrt{|S||A|TH} \\
&\leq O\left(\frac{1}{\varepsilon}|S||A|^{\frac{1}{2}} NH\sqrt{T\log T}\right) \qquad \text{(F.17)}
\end{aligned}$$

Combining everything together In the first part, we show that $\sum_{t=1}^{T} \mathrm{Reg}^{(t)} 1_{\boldsymbol{P}^\star \notin \boldsymbol{B}^{(t)}}$ is upper bounded by $O(\sqrt{T})$ for all $T \in \mathbb{N}$ with probability $1 - O(\delta)$. In the second part, we show that $\sum_{t=1}^{T} \mathrm{Reg}^{(t)} 1_{\boldsymbol{P}^\star \in \boldsymbol{B}^{(t)}} = O(|S||A|^{\frac{1}{2}} N\sqrt{T\log T})$. Therefore, we can conclude that the total regret $\mathrm{Reg}(T)$ is upper bounded by $O(|S||A|^{\frac{1}{2}} N\sqrt{T\log T})$ for all $T \in \mathbb{N}$ with probability $1 - O(\delta)$. □

F.4.4 Supplementary Lemma and Theorem

Assumption 4 (Ergodic Markov chain). *We denote $u_h^{P_i^\star,\pi_i}$ to be the state distribution of Markov chain induced by the MDP with transition probability $P_i^\star$ and policy π_i after h time steps. We assume $u_h^{P_i^\star,\pi_i}(s) > \varepsilon > 0$ for all entry $s \in \mathcal{S}$, all arm $i \in [N]$, $h \geq H$, and all policy π_i. In other words, the state distribution after H steps is universally lower-bounded by $\varepsilon > 0$, which we say that the MDP is H-step ε-ergodic.*

Assumption 4 can be achieved when both the MDP is ergodic and the horizon H is large enough.

Theorem 30 (Regret outside of H steps)**.** *When the Markov chain induced by transition $P_i^\star$ and policy π is H-step ε ergodic, we have:*

$$\mathbb{E}_{s_{h,i},a_{h,i}\sim P_i^\star,\pi} f(s_{h,i}, a_{h,i}) \leq \frac{1}{\varepsilon}\mathbb{E}_{s_{H,i},a_{H,i}\sim P_i^\star,\pi} f(s_{H,i}, a_{H,i}) \tag{F.18}$$

for all non-negative function f and $h \geq H$.

Proof.

$$\begin{aligned}
\mathbb{E}_{s_{h,i},a_{h,i}\sim P_i^\star,\pi} f(s_{h,i}, a_{h,i}) &= \sum_{s\sim S, a\sim A} \Pr(\pi_i(s) = a) u_h(s) f(s,a) \\
&\leq \sum_{s\sim S, a\sim A} \Pr(\pi_i(s) = a) f(s,a) \\
&\leq \sum_{s\sim S, a\sim A} \Pr(\pi_i(s) = a) \frac{u_H(s)}{\varepsilon} f(s,a) \\
&= \frac{1}{\varepsilon} \sum_{s\sim S, a\sim A} \Pr(\pi_i(s) = a) u_H(s) f(s,a) \\
&= \frac{1}{\varepsilon} E_{s_{H,i},a_{H,i}\sim P_i^\star,\pi} f(s_{H,i}, a_{H,i})
\end{aligned} \tag{F.19}$$

□

Lemma 7. *For any sequence of numbers $z_1, \cdots, z_T$ with $0 \leq z_j \leq H$ and $Z_t = \max\{1, \sum_{j=1}^{t} z_j\}$, we have:*

$$\sum_{t=1}^{T} \frac{z_t}{\sqrt{Z_{t-1}}} \leq \left(\sqrt{H+1}+1\right)\sqrt{Z_T} \tag{F.20}$$

Proof. Proof by induction. Assume that Equation F.20 holds for $T-1$. We have:

$$\sum_{t=1}^{T-1} \frac{z_t}{\sqrt{Z_{t-1}}} \leq \left(\sqrt{H+1}+1\right)\sqrt{Z_{T-1}} \tag{F.21}$$

Adding an additional term $\frac{z_T}{\sqrt{Z_{T-1}}}$, we get:

$$\sum_{t=1}^{T-1} \frac{z_t}{\sqrt{Z_{t-1}}} + \frac{z_T}{\sqrt{Z_{T-1}}} \leq \left(\sqrt{H+1}+1\right)\sqrt{Z_{T-1}} + \frac{z_T}{\sqrt{Z_{T-1}}}$$

$$
\begin{aligned}
&= \sqrt{\left(\sqrt{H+1}+1\right)^2 Z_{T-1} + 2\left(\sqrt{H+1}+1\right) z_T + \frac{z_T^2}{Z_{T-1}}} \\
&\leq \sqrt{\left(\sqrt{H+1}+1\right)^2 Z_{T-1} + 2\left(\sqrt{H+1}+1\right) z_T + H z_T} \\
&\leq \sqrt{\left(\sqrt{H+1}+1\right)^2 Z_{T-1} + \left(\sqrt{H+1}+1\right)^2 z_T} \\
&\leq \left(\sqrt{H+1}+1\right) \sqrt{Z_{T-1} + z_T} \\
&= \left(\sqrt{H+1}+1\right) \sqrt{Z_T} \qquad \text{(F.22)}
\end{aligned}
$$

which implies the Equation F.20 also holds for T.

The initial case with $T = 1$ holds trivially. Therefore, by induction, we conclude the proof. □

F.4.5 Regret Bound with Unknown Optimal Penalty

Theorem 14 (Regret bound with optimal penalty)**.** *Assume the penalty $\lambda^{(t)}$ in Algorithm 9 is updated by a saddle point $(\lambda^{(t)}, \boldsymbol{P}^{(t)}, \pi^{(t)}) = \arg\min_\lambda \max_{\boldsymbol{P},\pi} U_\pi^{\boldsymbol{P},\lambda}(\boldsymbol{s}_1)$ subject to constraints in Equation ($\mathcal{P}_V$). The cumulative regret of the optimal Lagrangian objective is bounded with probability $1-\delta$:*

$$
Reg_{\lambda^\star}(t) \leq O\left(\frac{1}{\varepsilon}|S||A|^{\frac{1}{2}} NH\sqrt{T\log T}\right) . \qquad (8.14)
$$

Proof. The main challenge of an unknown penalty term $\lambda^\star$ is that the optimality of the chosen transition $\boldsymbol{P}^{(t)}$ and policy $\pi^{(t)}$ does not hold in Theorem 11 due to the misalignment of the penalty $\lambda^{(t)}$ used in solving the optimization in Equation ($\mathcal{P}_V$) and the penalty $\lambda^\star$ used in computing the regret.

The optimality of $\lambda^{(t)}$ (minimizing $U_\pi^{\boldsymbol{P},\lambda}$) and the optimality of $\boldsymbol{P}^{(t)}$, $\pi^{(t)}$ (maximizing $U_\pi^{\boldsymbol{P},\lambda}$) are given by:

$$
\lambda^{(t)}, \boldsymbol{P}^{(t)}, \pi^{(t)} = \arg\min_\lambda \max_{\boldsymbol{P},\pi} U_\pi^{\boldsymbol{P},\lambda}
$$

which give us, respectively:

$$
U_{\pi^{(t)}}^{\boldsymbol{P}^{(t)},\lambda^{(t)}} \leq U_{\pi^{(t)}}^{\boldsymbol{P}^{(t)},\lambda^\star}, \quad U_{\pi^\star}^{\boldsymbol{P}^\star,\lambda^{(t)}} \leq U_{\pi^{(t)}}^{\boldsymbol{P}^{(t)},\lambda^{(t)}} \qquad \text{(F.23)}
$$

Similarly, the optimality of $\lambda^\star$ can be written as:

$$
\lambda^\star = \arg\min_\lambda U_{\pi^\star}^{\boldsymbol{P}^\star,\lambda}
$$

which gives us

$$U_{\pi^\star}^{\boldsymbol{P}^\star,\lambda^\star} \leq U_{\pi^\star}^{\boldsymbol{P}^\star,\lambda^{(t)}} \tag{F.24}$$

Combining Inequality F.23 and Inequality F.24, we can bound:

$$U_{\pi^\star}^{\boldsymbol{P}^\star,\lambda^\star} \leq U_{\pi^\star}^{\boldsymbol{P}^\star,\lambda^{(t)}} \leq U_{\pi^{(t)}}^{\boldsymbol{P}^{(t)},\lambda^{(t)}} \leq U_{\pi^{(t)}}^{\boldsymbol{P}^{(t)},\lambda^\star}$$

This implies that:

$$\mathrm{Reg}_{\lambda^\star}^{(t)} = U_{\pi^\star}^{\boldsymbol{P}^\star,\lambda^\star} - U_{\pi^{(t)}}^{\boldsymbol{P}^\star,\lambda^\star} \leq U_{\pi^{(t)}}^{\boldsymbol{P}^{(t)},\lambda^\star} - U_{\pi^{(t)}}^{\boldsymbol{P}^\star,\lambda^\star} \tag{F.25}$$

which is exactly the same result as shown in Equation 8.10. The rest of the proof follows the same argument of Theorem 12 and Theorem 13, which concludes the proof. □

F.4.6 Choice of Horizon and Ergodicity Constant ε

For a given Markov chain, we need H to be sufficiently large to ensure the probability of visiting any state after H steps is at least a positive constant $\varepsilon > 0$. The choice of H depends on the MDP; we elaborate below how to select H and ε.

We follow a similar analysis of Markov chain convergence from Chapter 10 in [290] by defining:

$$\omega_2 = \max_{\pi \in \Pi} \sigma_2(P_\pi)$$

where $\sigma_2(P)$ is the magnitude of the second largest eigenvalue of the random walk matrix P_π induced by the policy π. In practice, ω_2 can be upper bounded by 1 if the MDP satisfies some properties, e.g., laziness of the Markov chain induced from the MDP (Chapter 10.2 in [290]).

Let v be the corresponding stationary distribution of the random walk matrix P_π with the policy π that maximizes the second largest eigenvalue. We know that v is strictly positive by ergodicity. When $\sigma_2 < 1$, we can write $r = \min_i v_i > 0$ and choose $\varepsilon = \frac{1}{2}r > 0$.

Let w be an arbitrary initial distribution. By applying Theorem 10.4.1 from [290] (the directed graph version), for every $t > H = \log_{\omega_2}(\frac{1}{2}r^{3/2}) = \log_{\omega_2}(\sqrt{2}\varepsilon^{3/2})$, we have:

$$|v - P_\pi^t w|_1 \leq \sqrt{\frac{1}{\min_i v_i}}\omega_2^t \leq \frac{r}{2}$$

which implies that the minimum value of $P_\pi^t w$ and the minimum value of v, i.e., r, differ by at most $\frac{r}{2}$. This implies that the minimum value of $P_\pi^t w$ is at least $\frac{r}{2} = \varepsilon$ for any initial distribution w. This choice of ε and H satisfies our requirement mentioned in Appendix F.4.4.

F.5 Experiment Details

F.5.1 Whittle Index Implementation Speedups

We introduce a number of implementation-level improvements to speed up the computation of Whittle indices. To our knowledge these approaches are novel for Whittle index computation.

Early termination The key insight is that the Whittle index threshold policy will pull the arms with the K largest Whittle indices. As we compute Whittle indices for each of the N arms, after we have computed the first K Whittle indices, any future arm selected would have to have Whittle index at least as high as the K-th largest seen so far in order to be pulled. Let us notate the K-th largest value seen so far as `top-k`.

Whittle indices are computed using a binary search procedure[258], which at each iteration tracks the upper bound $\overline{\lambda}$ and lower bound $\underline{\lambda}$ of the index. Once the upper bound falls below that of the minimum value of the K largest indices so far $\overline{\lambda} <$ `top-k`, then we can terminate the binary search procedure as we are guaranteed that we would not act on that arm anyways. We implement the tracking of the K largest indices so far with a priority queue.

Similarly, we implement early termination to solve the bilinear programs ($\mathcal{P}_V$) and ($\mathcal{P}_m$) as callbacks in the Gurobi solver, in which we check the value of the current objective bound.

Memoization We memoize every Whittle index result computed throughout execution to track the index resulting from each pair of probabilities P_i and current state s_i as we perform calculations for each arm i. We implement this memoizer as a dictionary where the key is a tuple (P_i, s_i) with P_i recorded to four decimal places.

To implement the bilinear programs ($\mathcal{P}_V$) and ($\mathcal{P}_m$), we similarly memoize using the lower confidence bound (LCB) and upper confidence bound (UCB) that comprise the space $\boldsymbol{B}_i^{(t)}$.

F.5.2 Synthetic Data

The synthetic datasets are created by generating transition probabilities $P^i_{s,a,s'}$ sampled uniformly at random from the interval $[0, 1]$ for each arm i, starting state s, action a, and next state s'. Specifically we select transition probabilities for the probability of transitioning to a good state $P^i_{s,a,s'=1}$, then set $P^i_{s,a,s'=0} = 1 - P^i_{s,a,s'=1}$.

To ensure the validity constraints that acting is always helpful and starting in the good state is always helpful, we apply the following: for all arms $i \in [N]$:

- *Acting is always helpful:* If this requirement is violated with $P^i_{s,a=1,1} < P^i_{s,a=0,1}$, then $P^i_{s,a=0,1} = P^i_{s,a=1,1} \times \eta$ where η is uniform noise sampled between $[0, 1]$.
- *Starting from good state is always helpful:* If this requirement is violated with $P^i_{s=1,a,1} < P^i_{s=0,a,1}$, then $P^i_{s=0,a,1} = P^i_{s=1,a,1} \times \eta$ where η is uniform noise sampled between $[0, 1]$.

The *thin margin* dataset is created by mirroring the procedure described above but then constraining the probability of transitioning to a good state $P^i_{s,a,s'=1}$ to the interval $[0.2, 0.4]$. Thus the probabilities of transitioning to the bad state $P^i_{s,a,s'=0}$ are all between $[0.6, 0.8]$.

F.5.3 Acting in Low-Budget Settings

The potential impact of effectively allocating one resource is greater in low-budget settings. As one example, the ARMMAN setting from our experiments helps distribute a small number of health-care workers across a group of pregnant women for preventative health care. We study real data from ARMMAN to show that the performance gap between approaches is wider in low-budget settings.

Using one actual instance from ARMMAN, we consider distributing healthcare workers across mothers (arms). Using the true transition probabilities, we calculate the (sorted) Whittle indices of an optimal policy as: 0.42, 0.39, 0.28, 0.23, 0.19, 0.11, 0.07, 0.

In the table below, we first show the expected reward of the optimal action and a random action (baseline) as we increase budget in the ARMMAN problem. We then calculate the difference in reward between the optimal action and random action for each budget level, normalized per worker. It is clear that the potential impact over the baseline of effectively allocating one worker is greater in low budget settings.

	Reward		Reward gap per worker
K	Optimal	Random	$(\text{Opt} - \text{Random})/K$
1	0.42	0.211	0.209
2	0.81	0.423	0.194
3	1.09	0.634	0.152
4	1.32	0.845	0.119
5	1.51	1.056	0.091
6	1.62	1.268	0.059
7	1.69	1.479	0.030
8	1.69	1.690	0.000

Table F.3: Average reward contribution from each health worker in the online learning restless multi-armed bandit problem analysis.

F.5.4 Computation Infrastructure

All results are averaged over 30 random seeds. Experiments were executed on a cluster running CentOS with Intel(R) Xeon(R) CPU E5-2683 v4 @ 2.1 GHz with 8GB of RAM using Python 3.9.12. The bilinear program solved using Gurobi optimizer 9.5.1.

G
Appendix to Chapter 9

G.1 Computation Infrastructure

All the experiments were run on instances with 8 CPUs using 2nd generation Intel Xeon Platinum 8000 series processor (Skylake-SP or Cascade Lake) with a sustained all core Turbo CPU clock speed of up to 3.6 GHz. All algorithms do not require GPU to run. The implementation will be made available when accepted.

G.2 Societal Impact

The idea of smoothed online combinatorial optimization is not restricted to distributed streaming systems. Anything with a switching cost can be benefited from the study of smoothed online combinatorial optimization, including public policy with a switching cost[96], medication and wireless scheduling problems[181], where both of these can impact the process of policy making and scheduling algorithms. Including the distributed streaming system problem, these are all applications of smoothed online combinatorial optimization that can lead to change of the current algorithm design in our daily life with impact to the society.

G.3 Proofs of Theorem 15 and Theorem 16

Theorem 15. *Under Assumption 1, the regret from time step t to $t+S-1$ in Equation 9.3 is upper bounded by:* $\mathrm{Reg}_t^{t+S-1}(z_{t-1}) \leq 2L \sum_{s=t}^{t+S-1} \varepsilon_s^{(t)}$. *where L is the Lipschitz constant in Assumption 1.*

Proof. For simplicity of the proof, we define function $g(x, y, z)$ as follows:

$$g(x, y, z) := f(x, y) + d(x, z)$$

which includes both the cost from the cost function f and the switching cost d.

Let $\{z'_s\}_{s\in\{t,t+1,\cdots,t+S-1\}}$ be the optimal solutions when the full information of the cost function parameters $\{\theta_s\}_{s\in\{t,t+1,\cdots,t+S-1\}}$ is given. Let $\{z_s\}_{s\in\{t,t+1,\cdots,t+S-1\}}$ be the optimal solutions using the predicted parameters $\{\theta_s^{(t)}\}_{s\in\{t,t+1,\cdots,t+S-1\}}$. Without loss of generality, we let $z'_{t-1} = z_{t-1}$ to be the same initial decision at the time step $t-1$. We have:

$$\begin{aligned}
&\mathrm{Reg}_t^{t+S-1}(z_{t-1}) \\
=&\left(\sum_{s=t}^{t+S-1} g(z_s, \theta_s, z_{s-1}) - g(z'_s, \theta_s, z'_{s-1})\right) \\
=&\left(\sum_{s=t}^{t+S-1} g(z_s, \theta_s, z_{s-1}) - g(z_s, \theta_s^{(t)}, z_{s-1})\right) + \left(\sum_{s=t}^{t+S-1} g(z_s, \theta_s^{(t)}, z_{s-1}) - g(z'_s, \theta_s^{(t)}, z'_{s-1})\right) \\
&\qquad + \left(\sum_{s=t}^{t+S-1} g(z'_s, \theta_s^{(t)}, z'_{s-1}) - g(z'_s, \theta_s, z'_{s-1})\right) \qquad \text{(G.1)} \\
\leq& \sum_{s=t}^{t+S-1} L\left\|\theta_s - \theta_s^{(t)}\right\| + 0 + \sum_{s=t}^{t+S-1} L\left\|\theta_s^{(t)} - \theta_s\right\| \\
=&2L \sum_{s=t}^{t+S-1} \left\|\theta_s - \theta_s^{(t)}\right\| \\
\leq&2L \sum_{s=t}^{t+S-1} \varepsilon_s^{(t)}
\end{aligned}$$

The first term in Equation (G.1) can be bounded by (similar the third term):

$$\begin{aligned}
g(z_s, \theta_s, z_{s-1}) - g(z_s, \theta_s^{(t)}, z_{s-1}) =& f(z_s, \theta_s, z_{s-1}) + d(z_s, z_{s-1}) - f(z_s, \theta_s^{(t)}, z_{s-1}) - d(z_s, z_{s-1}) \\
=& f(z_s, \theta_s, z_{s-1}) - f(z_s, \theta_s^{(t)}, z_{s-1}) \leq L\left\|\theta_s - \theta_s^{(t)}\right\|
\end{aligned}$$

The second term in Equation (G.1) is non-positive because the optimality of the sequence $\{z_s\}_{s\in\{t,t+1,\cdots,t+S-1\}}$

when using the predictions as the parameters, i.e.,

$$\sum_{s=t}^{t+S-1} g(z_s, \theta_s^{(t)}, z_{s-1}) \leq \sum_{s=t}^{t+S-1} g(z_s^*, \theta_s^{(t)}, z_{s-1}^*)$$

□

Theorem 16. *Given Lipschitzness L in Assumption 1 and the maximal switching cost B in Assumption 2, in T time steps, Algorithm 10 achieves cumulative regret upper bounded by 2BI, where I is the total number of planning windows used in Algorithm 10.*

Proof. In the offline setting, given all the traffic up to time T, we can solve the optimization problem in Equation (9.1) to get the optimal solution z^*. We use $\text{cost}(z^*, \theta)$ to denote the optimal offline cost.

On the other hand, we assume that Algorithm 10 runs with I restarts and each restart runs S_i time steps using the predictions to plan ahead for each $i \in [I]$. Let $T_i = \sum_{j=1}^{i-1} S_j + 1$ be the start time of the i-th planning window. We can split the decisions into chunks — $\{z_{T_i+s}\}_{s\in\{0,1,\cdots,S_i-1\}}$ for each $i \in [I]$ that correspond to the decisions obtained in the i-th planning window.

Now we would like to compare the cost of the offline optimal solution $\{z_t^*\}_{t\in[T]}$ with the online solution $\{z_t\}_{t\in[T]}$ within the i-th chunk $\{T_i, T_i+1, \cdots, T_i+S_i-1\}$. Since the initial point $z_{T_i-1}^*$ of the offline optimal solution and the initial point z_{T_i-1} of the online solution are different, we cannot directly apply the result in Theorem 15 to bound the regret.

To resolve the misalignment, we additionally define $\{z_t'\}_{t\in\{T_i,T_i+1,\cdots,T_i+S_i-1\}}$ to be a *new* offline optimal solution starting from T_i till $T_i + S_i - 1$ with $z_{T_i-1}' = z_{T_i-1}$ being the initial point. z_t' serves as an intermediate to link z_t^* and z_t. Compare to this new offline solution with the same initial decision, the corresponding regret becomes:

$$\text{Reg}_{T_i}^{T_i+S_i-1} = \text{Reg}_{T_i}^{T_i+S_i-1}(z_{T_i-1}) := \sum_{t=T_i}^{T_i+S_i-1} \left(f(z_t, \theta_t) + d(z_t, z_{t-1})\right) - \sum_{t=T_i}^{T_i+S_i-1} \left(f(z_t', \theta_t) + d(z_t', z_{t-1}')\right) \tag{G.2}$$

Therefore, we can write:

$$\sum_{t=T_i}^{T_i+S_i-1} \left(f(z_t, \theta_t) + d(z_t, z_{t-1})\right) \tag{G.3}$$

$$= \text{Reg}_{T_i}^{T_i+S_i-1} + \sum_{t=T_i}^{T_i+S_i-1} \left(f(z_t', \theta_t) + d(z_t', z_{t-1}')\right) \tag{G.4}$$

$$\leq \text{Reg}_{T_i}^{T_i+S_i-1} + f(z_{T_i}^*, \theta_{T_i}) + d(z_{T_i}^*, z_{T_i-1}) + \sum_{t=T_i+1}^{T_i+S_i-1} \left(f(z_t^*, \theta_t) + d(z_t^*, z_{t-1}^*)\right) \tag{G.5}$$

$$\leq \text{Reg}_{T_i}^{T_i+S_i-1} + B + f(z^*_{T_i}, \theta_{T_i}) + d(z^*_{T_i}, z^*_{T_i-1}) + \sum_{t=T_i+1}^{T_i+S_i-1} \left(f(z^*_t, \theta_t) + d(z^*_t, z^*_{t-1})\right) \tag{G.6}$$

$$= \text{Reg}_{T_i}^{T_i+S_i-1} + B + \sum_{t=T_i}^{T_i+S_i-1} \left(f(z^*_t, \theta_t) + d(z^*_t, z^*_{t-1})\right) \tag{G.7}$$

First, from Equation (G.3) to Equation (G.4) is by the definition of $\text{Reg}_{T_i}^{T_i+S_i-1}$ in Equation (G.2). Second, Equation (G.4) to Equation (G.5) is due to the optimality of z'_t:

$$\{z'_t\}_{t\in\{T_i,T_i+1,\cdots,T_i+S_i-1\}} = \arg\min_y \sum_{t=T_i}^{T_i+S_i-1} \left(f(y_t, \theta_t) + d(y_t, y_{t-1})\right), \text{ where } y_{T_i-1} = z_{T_i-1}$$

Therefore, plugging in the original optimal solution z^* results in a larger cost in Equation (G.5).

Lastly, Equation (G.5) and Equation (G.6) only differ by the initial point at time step T_i, where Equation (G.5) uses z_{T_i-1} and Equation (G.6) uses $z^*_{T_i-1}$. Thus the difference is bounded by the maximal switching cost B.

We can reorganize the inequality in Equation (G.7) to get:

$$\begin{aligned}
&\sum_{t=T_i}^{T_i+S_i-1} \left(f(z_t, \theta_t) + d(z_t, z_{t-1})\right) - \sum_{t=T_i}^{T_i+S_i-1} \left(f(z^*_t, \theta_t) + d(z^*_t, z^*_{t-1})\right) \\
\leq\ & \text{Reg}_{T_i}^{T_i+S_i-1}(z_{T_i-1}) + B \\
\leq\ & 2L \sum_{s=T_i}^{T_i+S_i-1} \varepsilon_s^{(t)} + B \\
=\ & 2B
\end{aligned}$$

where the last inequality is by the choice of the dynamic planning window S_i such that $2L \sum_{s=T_i}^{T_i+S_i-1} \varepsilon_s^{(t)} \leq B$. Lastly, we can take summation over all the $i \in [I]$ to get:

$$\begin{aligned}
\text{Reg}_T &= \sum_{t=1}^{T} \left(f(z_t, \theta_t) + d(z_t, z_{t-1})\right) - \sum_{t=1}^{T} \left(f(z^*_t, \theta_t) + d(z^*_t, z^*_{t-1})\right) \\
&= \sum_{i=1}^{I} \left(\sum_{t=T_i}^{T_i+S_i-1} \left(f(z_t, \theta_t) + d(z_t, z_{t-1})\right) - \sum_{t=T_i}^{T_i+S_i-1} \left(f(z^*_t, \theta_t) + d(z^*_t, z^*_{t-1})\right) \right) \\
&\leq \sum_{i=1}^{I} 2B \\
&= 2BI
\end{aligned}$$

□

G.4 Proof of Corollary 1

Corollary 1. *If the uncertainty satisfies* $\varepsilon_{t+s-1}^{(t)} = O(\frac{s^a}{t^b})$, $\forall s, t \in \mathbb{N}$ *with* $a, b \in \mathbb{R}_{\geq 0}$, *we have:*

$$\mathrm{Reg}_T \leq \begin{cases} O(T^{1-\frac{b}{a+1}}) & \text{if } b < a+1 \\ O(\log T) & \text{if } b = a+1 \\ O(\log\log T) & \text{if } b > a+1 \end{cases}.$$

To prove Corollary 1, we need the following lemmas:

Lemma 8. *Given any fixed* $0 \leq \alpha$ *and the following recursive formula:*

$$T_1 = 1$$
$$T_{i+1} \geq T_i + A \cdot T_i^{\alpha}, \forall i \geq 1.$$

We can show:

$$T_i \geq \begin{cases} c \cdot i^{\beta} & \text{if } \alpha < 1 \\ (A+1)^{i-1} & \text{if } \alpha = 1 \\ (A+1)^{\alpha^{(i-2)}} & \text{if } \alpha > 1 \end{cases}$$

where $\beta = \frac{1}{1-\alpha}$ *if* $\alpha < 1$. *The constant* $c \in \mathbb{R}_{\geq 0}$ *satisfies* $c \leq (\frac{e^{\beta}}{A})^{1-\alpha} = eA^{\alpha-1}$ *and* $c \leq 1$.

We prove three different cases separately.

- **Case 1** ($\alpha < 1$) (this is deferred to the end).
- **Case 2** ($\alpha = 1$).
- **Case 3** ($\alpha > 1$).

Proof of Case 2. The recursive formula reduces to $T_{i+1} \geq (A+1)T_i$, where we can easily show that $T_i \geq (A+1)^{i-1}$. □

Proof of Case 3. The recursive formula can be written as $T_{i+1} \geq T_i^{\alpha}$ and $T_2 \geq A+1$. Thus we can simply unroll the recursion to get

$$T_i \geq T_{i-1}^{\alpha} \geq T_{i-2}^{\alpha^2} \geq \cdots \geq T_2^{\alpha^{(i-2)}} = (A+1)^{\alpha^{(i-2)}}$$

□

Proof of Case 1. We prove by induction.

Base case: Since $c \leq 1$, the base case is automatically satisfied by $1 = T_1 \geq c = c \cdot 1^\beta$.

Inductive step: By induction, assume $T_i \geq c \cdot i^\beta$. By our choice of c, we can see that $Ac^{\alpha-1} \geq e^\beta$, which implies:

$$Ac^{\alpha-1} i^{\alpha\beta} \geq e^\beta \cdot i^{\alpha\beta} = e^\beta \cdot i^{\beta-1} \tag{G.8}$$

where the second step follows from $\alpha\beta = \beta - 1$.

Therefore, we can lower bound T_{i+1} by:

$$\begin{aligned}
T_{i+1} &\geq T_i + AT_i^\alpha \\
&\geq (c \cdot i^\beta) + A(c \cdot i^\beta)^\alpha && \text{by } T_i \geq ci^\beta \\
&= c \cdot (i^\beta + Ac^{\alpha-1} i^{\alpha\beta}) \\
&\geq c \cdot (i^\beta + e^\beta i^{\beta-1}) && \text{by Equation (G.8)} \\
&\geq c \cdot (i+1)^\beta, && \text{by Lemma 9}
\end{aligned}$$

where we can apply Lemma 9 because $\beta = \frac{1}{1-\alpha} \geq 1$ for all $\alpha \in [0, 1)$. □

Lemma 9.

$$x^k + e^k x^{k-1} \geq (x+1)^k \quad \forall x \geq 1, k \geq 1 \tag{G.9}$$

Proof. Define a function $g(x, k) = x^k + e^k x^{k-1} - (x+1)^k$. We can check that $g(x, 1) = x + e - (x+1) > 0$. Next, we show that $g(x, k)$ is an increasing function in k when $x \geq 1$. Notice that the derivative $\frac{dg}{dk}$ can be written as:

$$\begin{aligned}
\frac{dg}{dk}\,|_{x,k} &= \log x \cdot x^k + e^k x^{k-1} + \log x \cdot e^k x^{k-1} - \log(x+1) \cdot (x+1)^k \\
&= \log x \cdot x^k + e^k x^{k-1} + \log x \cdot e^k x^{k-1} - \log x \cdot (x+1)^k - \log(\frac{1+x}{x}) \cdot (x+1)^k \\
&= \log x \cdot (x^k + e^k x^{k-1} - (x+1)^k) + e^k x^{k-1} - \log(1 + \frac{1}{x}) \cdot (x+1)^k \\
&\geq \log x \cdot g(x, k) + \left(e^k x^{k-1} - \frac{1}{x} \cdot (x+1)^k \right)
\end{aligned} \tag{G.10}$$

where the last inequality is due to $\log(1 + \frac{1}{x}) \leq \frac{1}{x}$.

The second term in Equation (G.10) can be written as:

$$e^k x^{k-1} - \frac{1}{x} \cdot (x+1)^k = \frac{1}{x} \left((ex)^k - (x+1)^k \right) > 0 \tag{G.11}$$

which is always satisfied because $ex > x + 1 \; \forall x \geq 1$.

Therefore, Equation (G.10) and Equation (G.11) together guarantee that if the value $g(x,k) \geq 0$, then its derivative is positive $\frac{dg}{dk}\mid_{x,k} > 0$ because every term in Equation (G.10) is positive. So now we have $g(x,1) > 0$ and the derivative $\frac{dg}{dk}\mid_{x,k} > 0$ if $g(x,k) \geq 0$.

Lastly, we just need to ensure that the function is always non-negative. Given fixed x, define $U = \{k > 1 \mid g(x,k) < 0\}$. We will prove by contradiction by assuming U is non-empty. Given that U is not empty, we can choose $u = \inf\{k : k \in U\}$. By the continuity of function g, $g(x,u) \leq 0$. Since $g(x,1) > 0$ and the continuity of g, we can find $g(x,1+\varepsilon) > 0$ for all $\varepsilon \in B(0,r)$ in a small open ball. Thus $u \geq 1+\varepsilon > 1$. Now by the mean value theorem applied on $g(x,1) > 0$ and $g(x,u) \leq 0$, we can find a value $v \in (1,u)$ such that $g(x,v) = \frac{g(x,u)-g(x,1)}{u-1} < 0$. However, we have proven that if $g(x,k) \geq 0$ then we know $\frac{dg}{dk}\mid_{x,k} > 0$. Since we have $g(x,v) < 0$, this implies $g(x,v) < 0$ as well with $v \in (1,u)$ and thus $v \in U$, which contradicts to the definition of u, i.e., the infimum of the set U. Thus the assumption that U is non-empty is incorrect. We conclude that U is empty. Thus for any given x, we have $g(x,k) \geq 0$ for all k, which implies the original inequality.

□

Now we are ready to prove Corollary 1.

Proof of Corollary 1. First, let S_i denote the size of the i-th planning window in Algorithm 10 for each $i \in [I]$. Let $T_i = \sum_{j=1}^{i-1} S_j + 1$ denote the start time of the i-th planning part.

In the i-th iteration of Algorithm 10 starting at time T_i, S_i is chosen such that S_i is the largest integer* satisfying $2L \sum_{s=T_i}^{T_i+S_i-1} \varepsilon_s^{(T_i)} \leq B$. This implies $2L \sum_{s=T_i}^{T_i+S_i} \varepsilon_s^{(T_i)} > B$ and we can estimate S_i by:

$$B < 2L \sum_{s=1}^{S_i+1} \varepsilon_{T_i+s-1}^{(T_i)} \leq 2LD \sum_{s=1}^{S_i+1} \frac{s^a}{T_i^b} \leq \frac{2LD}{a+1} \frac{(S_i+2)^{a+1}}{T_i^b}$$

for some constant $D > 0$. This suggests:

$$(\frac{a+1}{2D})^{\frac{1}{a+1}} (\frac{B}{L})^{\frac{1}{a+1}} T_i^{\frac{b}{a+1}} \leq S_i + 2 \leq 3S_i, \qquad AT_i^{\frac{b}{a+1}} \leq S_i$$

where $A = \frac{1}{3}(\frac{a+1}{2D})^{\frac{1}{a+1}} (\frac{B}{L})^{\frac{1}{a+1}} = \Theta((\frac{B}{L})^{\frac{1}{a+1}})$ is a constant dependent on the maximal switching cost B and the Lipschitzness L.

Therefore, we have

$$T_1 = 1, \quad T_{i+1} = T_i + S_i \geq T_i + AT_i^{\frac{b}{a+1}}$$

*We need $B \geq \varepsilon_{T_i}^{(T_i)}$ to ensure that we can at least choose $S_i \geq 1$. In the extreme case where $B < \varepsilon_{T_i}^{(T_i)}$, it implies that the uncertainty is too large while the switching cost is relatively small. Thus it is ideal to re-plan every time step because switching is cheap. The analysis of balancing switching cost and future planning does not apply.

where Lemma 8 can be applied to get:

$$T_i \geq \begin{cases} c i^{\frac{a+1}{a+1-b}} & \text{if } b < a+1 \\ (A+1)^{i-1} & \text{if } b = a+1 \\ (A+1)^{\frac{b}{a+1}(i-2)} & \text{if } b > a+1 \end{cases}$$

with the choice of the constant $c = \min(1, eA^{\frac{a+1-b}{a+1}})$. Lastly, since $T_I \leq T$, we can bound the total iteration I by:

$$T \geq T_I \geq \begin{cases} c I^{\frac{a+1}{a-b+1}} & \text{if } b < a+1 \\ (A+1)^{I-1} & \text{if } b = a+1 \\ (A+1)^{\frac{b}{a+1}(I-2)} & \text{if } b > a+1 \end{cases}$$

which gives:

$$I \leq \begin{cases} (\frac{T}{c})^{\frac{a-b+1}{a+1}} \\ \log_{A+1} T + 1 \\ \log_{A+1} \log_{\frac{b}{a+1}} T + 2 \end{cases} = \begin{cases} O(T^{1-\frac{b}{a+1}}) & \text{if } b < a+1 \\ O(\log T) & \text{if } b = a+1 \\ O(\log \log T) & \text{if } b > a+1 \end{cases}$$

By applying Theorem 16 and substituting the total number of iterations I by the above inequality, we get:

$$\mathrm{Reg}_T \leq \Theta(BI) = \begin{cases} O(T^{1-\frac{b}{a+1}}) & \text{if } b < a+1 \\ O(\log T) & \text{if } b = a+1 \\ O(\log \log T) & \text{if } b > a+1 \end{cases}$$

□

G.5 Proof of Corollary 2

Corollary 2. *Given $\varepsilon^{(t)}_{t+s-1} = \Omega(\frac{s^a}{t^b})$ for all $t, s \in \mathbb{N}$ with $0 \leq b$, there exist instances such that for any randomized algorithm, the expected regret is at least:*

$$\mathbb{E}[\mathrm{Reg}_T] \geq \begin{cases} \Omega(T^{1-b}) & \textit{if } b < 1 \\ \Omega(\log T) & \textit{if } b = 1 \\ \Omega(1) & \textit{if } b > 1 \end{cases}.$$

Proof. Let $\varepsilon_s^{(t)} = \frac{1}{t^b} = O(\frac{s^a}{t^b})$ for all $t, s \in \mathbb{N}$ with $\frac{1}{t^b} < \frac{1}{2}$. We construct a sequence of one-dimensional incoming traffic $\theta_t = \begin{cases} \frac{1}{2} + \frac{1}{t^b} \\ \frac{1}{2} - \frac{1}{t^b} \end{cases}$ and a one-dimensional feasible set $\mathcal{Z} = \{0, 1\}$. The prediction given by the predictive model is $\theta_s^{(t)} = \frac{1}{2}$ for all $t, s \in \mathbb{N}$, whose predictive error satisfies the bound $\left\|\theta_s - \theta_s^{(t)}\right\| \leq \frac{1}{t^b} = \varepsilon_s^{(t)}$. Assume that the cost function is defined by $f(z, \theta) = L \|z - \theta\|$ and there is no switching cost $d(z, y) = 0$.

Under this construction, if all the incoming traffics are given in advance, the optimal cost within T time steps is:

$$L \sum_{i=1}^{T} \left(\frac{1}{2} - \frac{1}{t^b} \right) = \frac{LT}{2} - L \sum_{i=1}^{T} t^{-b} \leq \begin{cases} \frac{LT}{2} - \frac{1}{1-b} T^{1-b} & \text{if } b < 1 \\ \frac{LT}{2} - \log T & \text{if } b = 1 \\ \frac{LT}{2} - \Theta(1) & \text{if } b > 1 \end{cases}$$

where we can just choose $z_t = 1$ if θ_t is closer to 1 and 0 otherwise.

On the other hand, if the incoming traffics are not given in advance, any decision made at time step t produces cost $L(\frac{1}{2} + \frac{1}{t^b})$ with probability $\frac{1}{2}$ and cost $L(\frac{1}{2} - \frac{1}{t^b})$ with probability $\frac{1}{2}$, which gives expected cost $\frac{L}{2}$ and a cumulative cost $\frac{LT}{2}$. Therefore, the expected cumulative regret is at least $\begin{cases} \Theta(T^{1-b}) & \text{if } b < 1 \\ \Theta(\log T) & \text{if } b = 1 \\ \Theta(1) & \text{if } b > 1 \end{cases}$. □

G.6 Iterative Algorithm for Offline Problem with Switching Cost

Given imperfect predictions and the planning windows, we can reduce the online problem to an offline problem. This section focuses on solving the following offline combinatorial optimization problem with switching cost.

$$\min_{z_t \in \mathcal{Z}} \sum_{t=1}^{S} f(z_t, \theta_t) + d(z_t, z_{t-1}). \tag{G.12}$$

Solving Equation (G.12) is challenging because the combinatorial structure of the decision $z_t \in \mathcal{Z}_t$ and the additional temporal dependency caused by the switching cost $d(z_t, z_{t-1})$.

Algorithm 16: Iterative algorithm for offline problems

Initialization: Let $J = 10$ and $z_t = z_0$ for all $t \in [S]$.
for $j \in [J]$ **do**
  for $t \in [S]$ **do**
    Let $c = 0.5$ if $j < J$ otherwise $c = 1$.
    Solve Equation (G.13) with z_{t-1}, z_{t+1}, c to update z_t.

Decomposition and Iterative Algorithm

If we fix the assignments z_{t-1}, z_{t+1}, finding the optimal solution at time step t reduces to the following problem with $c = 1$:

$$z_t = \operatorname*{argmin}_{z \in \mathcal{Z}_t} f(z, \theta_t) + c(d(z, z_{t-1}) + d(z, z_{t+1})). \tag{G.13}$$

Compared to Equation (G.12), Equation (G.13) avoids the temporal dependency across multiple time steps and largely reduces the number of binary variables. In practice, solving Equation (G.13) is more tractable than solving Equation (G.12).

This observation motivates the idea of iteratively fixing the neighbor decisions z_{t-1}, z_{t+1} and updating the decision at time step t for all $t \in [S]$. We use $z_t = z_0$ to initialize all decisions. Then we can iteratively solve Equation (G.13) with different t to update the decision z_t. This method decouples the temporal dependency and reduces the problem to a standard combinatorial optimization of function f with additional regularization terms. We can use mixed integer linear program or any other approximation algorithms to solve Equation (G.13).

Moreover, we can notice that any improvement made by solving Equation (G.13) with $c = 1$ provides the same improvement to Equation (G.12). This suggests that the optimal decision of Equation (G.12) is a fixed point of Equation (G.13) when $c = 1$.

Theorem 31. *The optimal sequence $\{z_t^*\}_{t \in [S]}$ of Equation* (G.12) *is a fixed point of Equation* (G.13) *with $c = 1$.*

Proof. Suppose that $\{z_s^*\}_{s \in [S]}$ optimizes Equation (G.12). For any $t \in [S]$, if we can find z_t' gets a positive improvement in Equation (G.13) with $c = 1$:

$$0 < \delta = \big(f(z_t^*, \theta_t) + d(z_t^*, z_{t-1}^* + d(z_t^*, z_{t+1}^*)\big) - \big(f(z_t', \theta_t) + d(z_t', z_{t-1}^*) + d(z_t', z_{t+1}^*)\big)$$

Then the new sequence $\{z_1^*, \cdots, z_{t-1}^*, z_t', z_{t+1}^*, \cdots, z_S^*\}$ gets the same improvement with:

$$\text{cost}(\{z_s^*\}_{s \in [S]}) - \text{cost}(\{z_1^*, \cdots, z_{t-1}^*, z_t', z_{t+1}^*, \cdots, z_S^*\}) = \delta > 0 \tag{G.14}$$

where $\text{cost}(\{z_1^*, \cdots, z_{t-1}^*, z_t', z_{t+1}^*, \cdots, z_S^*\})$ is strictly smaller than the optimal value $\text{cost}(\{z_s^*\}_{s \in [S]})$, which violates the optimality assumption of $\{z_s^*\}_{s \in [S]}$. This implies that we cannot find z_t' that gives

a strictly smaller objective in Equation (G.13), which also implies that z_t^* is a fixed point to Equation (G.13) with $c = 1$ using z_{t-1}^* and z_{t+1}^* as the neighbor decisions. □

Theorem 31 ensures that the iterative process in Equation G.13 stops updating at the optimal solution. However, in practice, there could be multiple fixed points and suboptimal points due to the combinatorial structure. This can be problematic because the iterative process in Equation G.13 can stop at many different suboptimal solution without further improving the solution quality. To avoid getting stuck by suboptimal solutions, we use a smaller scaling constant $c = 0.5$ to relax the iterative update, and use $c = 1$ in the final step to strengthen the solution. The iterative algorithm is described in Algorithm 16, which can be used to replace Line 6 in Algorithm 10.

H
Appendix to Chapter 10

H.1 Implementation Details

We implement a differentiable PyTorch module to compute a sample of the followers' equilibria. The module takes the leader's strategy as input and outputs a Nash equilibrium computed in the forward pass using the relaxation algorithm. We use a random initialization to run the relaxation algorithm, which can reach to different equilibria depending on different initialization. Given the sampled equilibrium $\boldsymbol{z}^*$ computed in the forward pass, the backward pass is implemented by PyTorch autograd to compute all the second-order derivatives to express Equation 10.5. The backward pass solves the linear system in Equation 10.5 analytically to derive $\frac{d\boldsymbol{z}^*}{d\pi}$ as an approximate of the equilibrium flow.

This PyTorch module is used in all three examples in our experiment. The implementation is flexible as we just need to adjust the followers' objectives and constraints, the relaxation algorithm and the gradient computation all directly apply.

H.2 Proofs of Theorem 19 and Theorem 20

Theorem 19. *If $v(\boldsymbol{z}^*, \pi)$ is the equilibrium flow of the stochastic equilibrium oracle $\mathcal{O}(\pi)$, we have:*

$$\frac{d}{d\pi} \mathop{\mathbb{E}}_{\boldsymbol{z}^* \sim \mathcal{O}(\pi)} f(\boldsymbol{z}^*, \pi) = \mathop{\mathbb{E}}_{\boldsymbol{z}^* \sim \mathcal{O}(\pi)} \left[f_\pi(\boldsymbol{z}^*, \pi) + f_{\boldsymbol{z}}(\boldsymbol{z}^*, \pi) \cdot v(\boldsymbol{z}^*, \pi) \right]. \tag{10.8}$$

Proof. To compute the derivative on the left-hand side, we have to first expand the expectation

because the equilibrium distribution is dependent on the environment parameter π:

$$\begin{aligned}\frac{d}{d\pi}\mathop{\mathbb{E}}_{\boldsymbol{z}\sim\mathcal{O}(\pi)} f(\boldsymbol{z},\pi) =& \frac{d}{d\pi}\int_{\boldsymbol{z}\in\mathcal{Z}} f(\boldsymbol{z},\pi)p(\boldsymbol{z},\pi)d\boldsymbol{z} \\ =& \int_{\boldsymbol{z}\in\mathcal{Z}}\left(p(\boldsymbol{z},\pi)\frac{\partial}{\partial\pi}f(\boldsymbol{z},\pi) + f(\boldsymbol{z},\pi)\frac{\partial}{\partial\pi}p(\boldsymbol{z},\pi)\right)d\boldsymbol{z} \\ =& \mathop{\mathbb{E}}_{\boldsymbol{z}\sim\mathcal{O}(\pi)} f_\pi(\boldsymbol{z},\pi) + \int_{\boldsymbol{z}\in\mathcal{Z}} f(\boldsymbol{z},\pi)\frac{\partial}{\partial\pi}p(\boldsymbol{z},\pi)d\boldsymbol{z} \end{aligned} \tag{H.1}$$

We further define $\Phi(\boldsymbol{z},\pi) = p(\boldsymbol{z},\pi)v(\boldsymbol{z},\pi)$. By the equilibrium flow definition in Equation 10.7, we have

$$\frac{\partial}{\partial\pi}p(\boldsymbol{z},\pi) = -\nabla_{\boldsymbol{z}}\cdot\Phi(\boldsymbol{z},\pi)$$

Therefore, the later term in Equation H.1 can be computed by integration by parts and Stokes' theorem:

$$\begin{aligned}&\int_{\boldsymbol{z}\in\mathcal{Z}} f(\boldsymbol{z},\pi)\frac{\partial}{\partial\pi}p(\boldsymbol{z},\pi)d\boldsymbol{z} \\ =& -\int_{z\in\mathcal{Z}} f(\boldsymbol{z},\pi)\nabla_{\boldsymbol{z}}\cdot\Phi(\boldsymbol{z},\pi)d\boldsymbol{z} \\ =& -\int_{z\in\mathcal{Z}}\nabla_{\boldsymbol{z}}\cdot(f(\boldsymbol{z},\pi)\Phi(\boldsymbol{z},\pi))d\boldsymbol{z} + \int_{\boldsymbol{z}\in\mathcal{Z}} f_{\boldsymbol{z}}(\boldsymbol{z},\pi)\Phi(\boldsymbol{z},\pi)d\boldsymbol{z} \\ =& -\oint_{\partial\mathcal{Z}} f(\boldsymbol{z},\pi)\Phi(\boldsymbol{z},\pi)dS + \int_{\boldsymbol{z}\in\mathcal{Z}} f_{\boldsymbol{z}}(\boldsymbol{z},\pi)\Phi(\boldsymbol{z},\pi)d\boldsymbol{z}\end{aligned}$$

Therefore, we have

$$\begin{aligned}\frac{d}{d\pi}\mathop{\mathbb{E}}_{\boldsymbol{z}\sim\mathcal{O}(\pi)} f(\boldsymbol{z},\pi) =& \mathop{\mathbb{E}}_{\boldsymbol{z}\sim\mathcal{O}(\pi)} f_\pi(\boldsymbol{z},\pi) + \int_{\boldsymbol{z}\in\mathcal{Z}} f(\boldsymbol{z},\pi)\frac{\partial}{\partial\pi}p(\boldsymbol{z},\pi)d\boldsymbol{z} \\ =& \mathop{\mathbb{E}}_{\boldsymbol{z}\sim\mathcal{O}(\pi)} f_\pi(\boldsymbol{z},\pi) - \oint_{\partial\mathcal{Z}} f(\boldsymbol{z},\pi)\Phi(\boldsymbol{z},\pi)dS + \int_{\boldsymbol{z}\in\mathcal{Z}} f_{\boldsymbol{z}}(\boldsymbol{z},\pi)\Phi(\boldsymbol{z},\pi)d\boldsymbol{z} \\ =& \mathop{\mathbb{E}}_{\boldsymbol{z}\sim\mathcal{O}(\pi)} f_\pi(\boldsymbol{z},\pi) - \oint_{\partial\mathcal{Z}} f(\boldsymbol{z},\pi)p(\boldsymbol{z},\pi)v(\boldsymbol{z},\pi)dS + \int_{\boldsymbol{z}\in\mathcal{Z}} f_{\boldsymbol{z}}(\boldsymbol{z},\pi)p(\boldsymbol{z},\pi)v(\boldsymbol{z},\pi)d\boldsymbol{z} \\ =& \mathop{\mathbb{E}}_{\boldsymbol{z}\sim\mathcal{O}(\pi)}\left[f_\pi(\boldsymbol{z},\pi) + f_{\boldsymbol{z}}(\boldsymbol{z},\pi)v(\boldsymbol{z},\pi)\right]\end{aligned}$$

where the term $\oint_{\partial\mathcal{Z}} f(\boldsymbol{z},\pi)p(\boldsymbol{z},\pi)v(\boldsymbol{z},\pi)dS = 0$ because $p(\boldsymbol{z},\pi) = 0$ at the boundary $\partial\mathcal{Z}$. This concludes the proof of Theorem 19. □

Notice that in the proof of Theorem 19, we only use integration by parts and Stokes' theorem,

where both of them apply to Riemann integral and Lebesgue integral. Thus, the proof of Theorem 19 also works for any measure zero jumps in the probability density function.

Theorem 20. *Given the leader's strategy π and a sampled equilibrium $\boldsymbol{z}$, if (1) the KKT matrix at $(\boldsymbol{z}, \pi)$ is invertible and (2) the equilibrium $\boldsymbol{z}$ is sampled with a fixed probability locally, the solution to Equation 10.5 is a homogeneous solution to Equation 10.7 and matches the equilibrium flow $v(\pi, \boldsymbol{z})$ locally.*

Proof. Since the KKT conditions hold for all equilibria, the given π and $\boldsymbol{z}$ must satisfy $KKT(\boldsymbol{z}, \pi) = 0$. The KKT matrix in Equation 10.5 is given by $\frac{\partial KKT}{\partial \boldsymbol{z}}$, the Jacobian matrix of the function $KKT(\boldsymbol{z}, \pi)$ with respect to $\boldsymbol{z}$. If the KKT matrix is invertible, by implicit function theorem, there exists an open set U containing π such that there exists a unique continuously differentiable function $h : U \to \mathcal{Z}$ such that $h(\pi) = \boldsymbol{z}$ and $KKT(h(\pi'), \pi') = 0$ for all $\pi' \in U$. Moreover, the analysis in Equation 10.5 applies, where $\frac{dh(\pi)}{d\pi} = \frac{d\boldsymbol{z}}{d\pi}$ matches the solution of Equation 10.5.

Lastly, the condition that the equilibrium $\boldsymbol{z}$ is sampled with a fixed probability density c locally implies the corresponding probability density function must satisfy $p(z', \pi') = c1_{\mathrm{KKT}(z',\pi')=0} = c1_{\boldsymbol{z}'=h(\pi')}$ for all $\pi' \in U$ in an open set locally*.

Now we can verify whether $p(\boldsymbol{z}', \pi')$ and $v(\boldsymbol{z}', \pi') = \frac{dh(\pi')}{d\pi}$ (independent of $\boldsymbol{z}'$) satisfy the partial differential equation of equilibrium flow as defined in Definition 17. We first compute the left-hand side of Equation 10.7 by:

$$\frac{\partial}{\partial \pi} p(\boldsymbol{z}', \pi') = \frac{\partial}{\partial \pi} c1_{\boldsymbol{z}'=h(\pi')} = c\delta_{\boldsymbol{z}'=h(\pi')} \frac{dh(\pi')}{d\pi} \tag{H.2}$$

where Equation H.2 is derived by fixing $\boldsymbol{z}'$, the derivative of a jump function $1_{\boldsymbol{z}'=h(\pi')}$ is a Dirac delta function located at $\boldsymbol{z}' = h(\pi')$ multiplied by a Jacobian term $\frac{dh(\pi')}{d\pi}$.

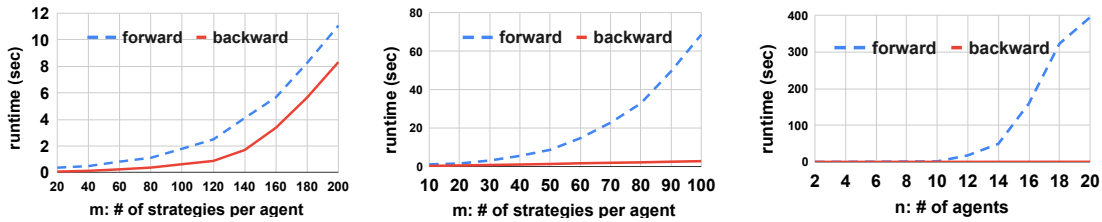
We can also compute the right-hand side of Equation 10.7 by:

$$\begin{aligned}
\nabla_z \cdot (p(\boldsymbol{z}', \pi') v(\boldsymbol{z}', \pi')) =& v(\boldsymbol{z}', \pi') \frac{\partial}{\partial \boldsymbol{z}} p(\boldsymbol{z}', \pi') + p(\boldsymbol{z}', \pi') \frac{\partial}{\partial \boldsymbol{z}} v(\boldsymbol{z}', \pi') && \text{(H.3)} \\
=& \frac{dh(\pi')}{d\pi} \frac{\partial}{\partial \boldsymbol{z}} c\mathbf{1}_{\boldsymbol{z}'=h(\pi')} \\
=& c\delta_{\boldsymbol{z}'=h(\pi')} \frac{dh(\pi')}{d\pi} && \text{(H.4)}
\end{aligned}$$

where the second term in Equation H.3 is 0 because we define $v(\boldsymbol{z}', \pi') = \frac{dh(\pi')}{d\pi}$, which is independent of $\boldsymbol{z}'$. Equation H.4 is derived by fixing π', the derivative of a jump function is a Dirac delta function located at $\boldsymbol{z}' = \pi'$.

The above calculation shows that Equation H.2 is identical to Equation H.4, which implies the left-hand side and the right-hand side of Equation 10.7 are equal. Therefore, we conclude that the

*We can choose the smaller subset U such that both the implicit function theorem and the locally fixed probability c both hold.



(a) Normal-form games with $n = 3$ followers and varied m strategies per follower.

(b) Stackelberg security games with $n = 5$ followers and varied m strategies per follower.

(c) Cyber insurance games with $m = 1$ dimensional investment decision and varied n.

Figure H.1: We compare the computation cost of equilibrium computation (forward) and the gradient access (backward) per iteration. Backward pass is cheaper than forward pass in all three domains. Gradient-based method runs a forward pass and a backward pass per iteration, while gradient-free method requires many forward passes to perform one step of local search.

choice of $v(\boldsymbol{z}', \pi') = \frac{d\boldsymbol{z}'}{d\pi} = \frac{dh(\pi')}{d\pi}$ is a homogeneous solution to differential equation in Equation 10.7 locally in $\pi' \in U$. By the definition of the equilibrium flow, $v(\boldsymbol{z}', \pi') = \frac{d\boldsymbol{z}'}{d\pi}$ is a solution to the equilibrium flow because we can subtract the homogeneous solution and define a new partial differential equation without region U to compute the solution outside of U. □

H.3 Limitation of Theorem 19 and Theorem 20

Although Theorem 19 always holds, the main challenge preventing us from directly applying Theorem 19 is that we do not know the equilibrium flow in advance. Given the probability density function of the equilibrium oracle, we can compute the equilibrium flow by solving the partial differential equation in Equation 10.7. However, the probability density function is generally not given.

Theorem 20 tells us that the derivative computed in Equation 10.5 is exactly the equilibrium flow defined by the partial differential equation when the sampled equilibrium admits to an invertible KKT matrix and is locally sampled with a fixed probability. That is to say, when these conditions hold, we can treat the equilibrium sampled from a distribution over multiple equilibria as a unique equilibrium to differentiate through as discussed in the section of unique Nash equilibrium. These conditions are also satisfied when the sampled equilibrium is locally stable without any discontinuous jump, generalizing the differentiability of unique Nash equilibrium and globally isolated Nash equilibria to the case with only conditions on the sampled Nash equilibrium.

H.4 Dimensionality and Computation Cost

H.4.1 Dimensionality of Control Parameters

We discuss the solution quality attained and computation costs required by different optimization methods. To understand the results, it is useful to compare the role and dimensionality of the environment parameter π in each setting.

- *Normal-form games*: parameter π corresponds to the non-negative subsidies provided to each follower for each entry of its payoff matrix. We have $\dim(\pi) = n \prod_{i=1}^{n} m_i = nm^n$, where for simplicity we set $m_i = m$ for all i.
- *Stackelberg security games*: parameter π refers to the non-negative subsidies provided to each follower at each available target. Because each follower i can only cover targets $T_i \subseteq T$, we have $\dim(\pi) = \sum_{i=1}^{n} |T_i| = nm$, where we set $|T_i| = m$ for all i.
- *Cyber insurance games*: each insurance plan is composed of a premium and a coverage amount. Therefore in total, $\dim(\pi) = 2n$, the smallest out of the three tasks.

H.4.2 Computation Cost

In Figure H.1, we compare the computation cost per iteration of equilibrium-finding oracle (forward) and the gradient oracle (backward). Due to the hardness of the Nash equilibrium-finding problem, no equilibrium oracle is likely to have polynomial-time complexity in the forward pass (computing an equilibrium). We instead focus more on the computation cost of the backward pass (differentiating through an equilibrium).

As we can see in Equation 10.5, the complexity of gradient computation is dominated by inverting the KKT matrix with size $L = O(nm)$ and the dimensionality of environment parameter π since the matrix $\frac{d\boldsymbol{z}^*}{d\pi}$ is of size $L \times \dim(\pi)$. Therefore, the complexity of the backward pass is bounded above by $O(L^\alpha) + O(L^2 \dim(\pi)) = O(n^\alpha m^\alpha) + O(n^2 m^2 \dim(\pi))$ with $\alpha = 2.373$.

- In Figure H.1(a), the complexity is given by $O(n^2 m^2 \dim(\pi)) = O(n^3 m^{n+2}) = O(m^5)$ where we set $n = 3$ with varied m, number of actions per follower, shown in the x-axis.
- In Figure H.1(b), the complexity is $O(n^2 m^2 \dim(\pi)) = O(m^3)$ with $n = 5$ and varied m, number of actions per follower, shown in the x-axis.
- In Figure H.1(c), the complexity is $O(n^2 m^2 \dim(\pi)) = O(n^3)$ with $m = 1$ and varied number of followers n shown in the x-axis. The runtime of the forward pass increases drastically, while the runtime of the backward pass remains polynomial.

In all three examples, the gradient computation (backward) has polynomial complexity and is faster than the equilibrium finding oracle (forward). Numerical gradient estimation in gradient-free methods requires repeatedly accessing the forward pass, which can be even more expensive than our gradient computation.

H.5 Optimization Reformulation of the Stackelberg Problems with Multiple Followers

In this section, we describe how to reformulate the leader's optimization problem with multiple followers involved into an single-level optimization problem with stationary and complementarity constraints. Notice that this reformulation requires the assumption that all followers break ties in favor of the leader, while our gradient-based method can deal with arbitrary oracle access not limited to any tie-breaking rules.

H.5.1 Normal-Form Games with Risk Penalty

In this example, the followers' objectives are defined by:

$$f_i(\boldsymbol{z}, \pi) = U_i(\boldsymbol{z}) + \pi_i(\boldsymbol{z}) - H(z_i)/\lambda, \tag{H.5}$$

where U_i is the given payoff matrix and π_i is the subsidy provided by the leader. H is the Gibbs entropy denoting the risk aversion penalty.

The leader's objective and the constraint are respectively defined by:

$$f(\boldsymbol{z}, \pi) = \sum_{i \in [n]} U_i(\boldsymbol{z})$$
$$g(\boldsymbol{z}, \pi) = \left(\sum_{i \in [n]} \pi_i(\boldsymbol{z}) \right) - B \leq 0.$$

Bilevel optimization formulation we can write the followers' best response into the leader's optimization problem:

$$\begin{aligned}
\max_{\pi} \quad & f(\boldsymbol{z}) = \sum_{i \in [n]} U_i(\boldsymbol{z}) = U(\boldsymbol{z}) \\
\text{s.t.} \quad & z_i \in [0,1]^{m_i}, \mathbf{1}^\top z_i = 1 && \forall i \in [n] \\
& z_i = \arg\max_{z \in \mathcal{Z}_i} f_i(z_i, z_{-i}, \pi) && \forall i \in [n] \\
& \pi(\boldsymbol{z}) \leq B
\end{aligned}$$

where f_i is defined in Equation H.5. By converting the inner-level optimization problem to its KKT conditions, we can rewrite the optimization problem as:

$$
\begin{aligned}
\min_{\pi,\boldsymbol{z},\lambda,\mu,\nu} \quad & -f(\boldsymbol{z}) = -U(\boldsymbol{z}) \\
\text{s.t.} \quad & z_i, \quad \mathbf{1}^\top z_i = 1 && \forall i \in [n] \\
& \lambda_i, \mu_i \in \mathbb{R}^{m_i}_{\geq 0}, \nu_i \in \mathbb{R} && \forall i \in [n] \\
& \lambda_{i,j} z_{i,j} = 0 && \forall i \in [n], j \in [m_i] \\
& \mu_{i,j}(1 - z_{i,j}) = 0 && \forall i \in [n], j \in [m_i] \\
& -\nabla_{z_i} f_i - \lambda_i + \mu_i + \nu_i \mathbf{1} = 0 && \forall i \in [n] \\
& \pi(\boldsymbol{z}) \leq B
\end{aligned}
$$

We add dual variables λ_i, μ_i to the inequality constraints $z_{i,j} \geq 0$ and $z_{i,j} \leq 1$ respectively. We also add dual variables ν_i to the equality constraints $\mathbf{1}^\top z_i = 1$. We can explicitly write down the gradient:

$$
\nabla_{z_i} f_i(z_i, z_{-i}, \pi) = (U_i + \pi_i)(z_{-i}) - \sum_j (1 + \log z_{ij})/\lambda \tag{H.6}
$$

where λ here is a specific constant (different from the Lagrangian multipliers), which is chosen to be 1 in our implementation.

H.5.2 Stackelberg Security Games With Multiple Defenders

The followers' objectives are defined by:

$$
f_i(\boldsymbol{z}, \pi) = \sum_{t \in T_i} (U_{i,t} + \pi_{i,t})(1 - y_t) p_t, \tag{H.7}
$$

where $U_{i,t}$ is the loss received by defender i when target t is successfully attacked, and $\pi_{i,t}$ is the corresponding reimbursement provided by the leader to remedy the loss. We define $y_t := 1 - \prod_i (1 - z_{i,t})$ to denote the effective coverage of target t, representing the probability that target t is protected under the overlapping protection patrol plan $\boldsymbol{z}$. Given the effective coverage of all targets, we assume the attacker attacks target t with probability $p_t = e^{-\omega y_t + a_t} / (\sum_{s \in T} e^{-\omega y_s + a_s})$, where $a_t \in \mathbb{R}$ is a known attractiveness value and $\omega \geq 0$ is a scaling constant.

The leader's objective and constraint are respectively defined by:

$$
f(\boldsymbol{z}, \pi) = \sum_{t \in T} U_t (1 - y_t) p_t
$$

$$g(\boldsymbol{z}, \pi) = \left(\sum_{i,t} \pi_{i,t}(1 - y_t)p_t \right) - B \leq 0,$$

where $U_t < 0$ is the penalty for the leader when target t is attacked without any coverage.

Bilevel optimization formulation Similarly, we can also write down the bilevel optimization formulation of the Stackelberg security games with multiple defenders as:

$$\begin{aligned}
\max_{\pi} \quad & f(\boldsymbol{z}) = \sum_{t \in T} U_t(1 - y_t)p_t \\
\text{s.t.} \quad & z_{i,t} \in [0, 1] && \forall i \in [n], t \in T_i \\
& y_t, p_t \in \mathbb{R} && \forall t \in T \\
& \sum_{t \in T_i} z_{i,t} = b_i && \forall i \in [n] \\
& y_t = 1 - \prod_{i: t \in T_i} (1 - z_{i,t}) && \forall t \in T \\
& p_t = \frac{e^{-\omega y_t + a_t}}{\sum_{s \in T} e^{-\omega y_s + a_s}} && \forall t \in T \\
& z_i = \arg\max_{z \in \mathcal{Z}_i} f_i(z_i, z_{-i}, \pi) && \forall i \in [n] \\
& \sum_{i,t} \left(\pi^u_{i,t}(1 - y_t)p_t + \pi^c_{i,t} y_t p_t \right) \leq B
\end{aligned}$$

where p_t is the probability that attacker will attack target t under protect scheme $\boldsymbol{z}$ and the resulting $\boldsymbol{y}$. The function f_i is defined in by:

$$f_i(\boldsymbol{z}, \pi) = \sum_{t \in T_i} (U_{i,t} + \pi_{i,t})(1 - y_t)p_t. \tag{H.8}$$

This bilevel optimization problem can be reformulated into a single level optimization problem if we assume all the individual followers break ties (equilibria) in favor of the leader, which is given by:

$$\begin{aligned}
\max_{\pi, \boldsymbol{z}, \lambda, \mu, \nu} \quad & \sum_{t \in T} U_t(1 - y_t)p_t \\
\text{s.t.} \quad & z_{i,t} \in [0, 1] && \forall i \in [n], t \in T_i \\
& y_t, p_t \in \mathbb{R} && \forall t \in T \\
& \sum_{t \in T_i} z_{i,t} = b_i && \forall i \in [n]
\end{aligned}$$

$$
\begin{aligned}
& y_t = 1 - \prod_{i:t \in T_i} (1 - z_{i,t}) && \forall t \in T \\
& p_t = \frac{e^{-\omega y_t + a_t}}{\sum_{s \in T} e^{-\omega y_s + a_s}} && \forall t \in T \\
& \lambda_{i,t}, \mu_{i,t} \in \mathbb{R}_{\geq 0}, \nu_i \in \mathbb{R}_{\geq 0} && \forall i \in [n], t \in T_i \\
& \lambda_{i,t} z_{i,t} = 0 && \forall i \in [n], t \in T_i \\
& \mu_{i,t}(1 - z_{i,t}) = 0 && \forall i \in [n], t \in T_i \\
& -\nabla_{z_i} f_i - \lambda_i + \mu_i + \nu_i \mathbf{1} = 0 && \forall i \in [n] \\
& \sum_{i,t} \left(\pi^{\mu}_{i,t}(1 - y_t) p_t + \pi^{c}_{i,t} y_t p_t \right) \leq B
\end{aligned}
$$

Similarly, we add dual variables $\lambda_{i,t}, \mu_{i,t}, \nu_i$ to constraints $z_{i,t} \geq 0$, $z_{i,t} \leq 1$, and $\sum_{t \in T_i} z_{i,t} = b_i$.

H.5.3 Cyber Insurance Games

The followers' objectives are defined by:

$$
f_i(\boldsymbol{z}, \pi) = -c_i z_i - \rho_i - (L_i - I_i) q_i - \gamma |L_i - I_i| \sqrt{q_i(1 - q_i)}, \tag{H.9}
$$

where c_i is the unit cost of the protection z_i and L_i is the loss when the computer is attacked. The insurance plan offered to agent i is defined as $z_i := (\rho_i, I_i)$, where ρ_i is the fixed premium paid to enroll in the insurance plan and I_i is the compensation received when the computer is attacked.

We assume the computer is attacked with a probability q_i, where $q_i = \sigma(-\sum_{j=1}^{n} w_{ij} z_j + v L_i)$ with σ being sigmoid function, a matrix $W = \{w_{ij} > 0\}_{i,j \in [n]}$ to represent the interconnectedness between agents, $v \geq 0$ to reflect the attacker's preference over high-value targets, and lastly it depends on the loss L_i incurred by agent i when attacked. This attack probability is a smooth non-convex function, which makes the reformulation approach hard and the non-convexity can lead to multiple equilibria reached by the followers.

The last term in Equation H.9 is the risk penalty to agent i. This term is the standard deviation of the loss received by agent i. We assume the agent is risk averse and thus penalized by a constant time of the standard deviation.

On the other hand, the leader's objective is defined by:

$$
f(\boldsymbol{z}, \pi) = \sum_{i=1}^{n} -I_i q_i + \rho_i
$$

where the leader's objective is simply the total revenue received by the insurer, which includes the

premium collected from all agents and the compensation paid to all agents.

The constraints are the individual rationality of each agent, where the customized insurance plan needs to incentivize the agent to purchase the insurance plan. In other words, the compensation I_i and premium ρ_i must incentivize agents to purchase the insurance plan by making the payoff with insurance no worse than the payoff without.

$$g_i(\boldsymbol{z}, \pi) = \left(-c_i z_i - L_i q_i - \gamma L_i \sqrt{q_i(1-q_i)}\right) - f_i(\boldsymbol{z}, \pi) \leq 0.$$

Bilevel optimization reformulation The bilevel optimization formulation for the cyber insurance domain with an external insurer is given by:

$$\begin{aligned}
\max_{\pi} \quad & f(\boldsymbol{z}) = \sum_{i=1}^{n} -I_i q_i + \rho_i \\
\text{s.t.} \quad & z_i \in [0, \infty) && \forall i \in [n] \\
& q_i = \sigma\left(-\sum_{j=1}^{n} w_{ij} z_j + v L_i\right) && \forall i \in [n] \\
& z_i = \arg\max_{z_i' \in \mathcal{Z}_i} f_i(z_i', z_{-i}, \pi) && \forall i \in [n] \\
& -c_i z_i - L_i q_i - \gamma L_i \sqrt{q_i(1-q_i)} \leq f_i(\boldsymbol{z}, \pi) && \forall i \in [n]
\end{aligned}$$

where $f_i(\boldsymbol{z}, \pi) = -c_i z_i - \rho_i - (L_i - I_i) q_i - \gamma \left\| L_i - I_i \right\| \sqrt{q_i(1-q_i)}$.

Reformulating this bilevel problem into a single level optimization problem, we have:

$$\begin{aligned}
\max_{\pi, \boldsymbol{z}, \lambda} \quad & f(\boldsymbol{z}) = \sum_{i=1}^{n} -I_i q_i + \rho_i \\
\text{s.t.} \quad & z_i \in [0, \infty), \lambda_i \in [0, \infty) && \forall i \in [n] \\
& q_i = \sigma\left(-\sum_{j=1}^{n} w_{ij} z_j + v L_i\right) && \forall i \in [n] \\
& z_i \lambda_i = 0 && \forall i \in [n] \\
& -c_i z_i - L_i q_i - \gamma L_i \sqrt{q_i(1-q_i)} \leq f_i(\boldsymbol{z}, \pi) && \forall i \in [n] \\
& -\nabla_{z_i} f_i - \lambda_i = 0 && \forall i \in [n]
\end{aligned}$$

with dual variables λ_i for the $z_i \geq 0$ constraint.

H.6 Experimental Setup

For reproducibility, we set the random seeds to be from 1 to 30 for NSGs and cyber insurance games, and from 1 to 100 for SSGs.

H.6.1 Normal-Form Games

In NFGs, we randomly generate the payoff matrix $U_i \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_n}$ of follower i with each entry of the payoff matrix randomly drawn from a uniform distribution $U(0, 10)$. We assume there are $n = 3$ followers. Each follower has three pure strategies to use $m_i = m = 3$ for all i. The risk aversion penalty constant is set to be $\lambda = 1$.

H.6.2 Stackelberg Security Games

In SSGs, we randomly generate the penalty $U_{i,t} < 0$ of each defender i associated to each target $t \in T_i \subset T$ from a uniform distribution $U_{i,t} \sim U(-10, 0)$. The leader's penalty $U_t < 0$ is also generated from the same uniform distribution $U_t \sim U(-10, 0)$. We assume there are $n = 5$ followers in total. There are $|T| = 100$ targets and each follower is able to protect $|T_i| = m = 50$ targets randomly sampled from all targets. Each follower can spend at most $b_i = 10$ effort on the available targets. The attractiveness values a_t used to denote the attacker's preference is randomly generated from a normal distribution $a_t \in \mathcal{N}(0, 1)$ with 0 mean and standard deviation 1. The scaling constant is set to be $\omega = 5$.

H.6.3 Cyber Insurance Games

In cyber insurance games, for each follower i, we generate the unit protection cost c_i from a uniform distribution $c_i \sim U(5, 10)$, and the incurred loss L_i from a uniform distribution $L_i \sim U(50, 100)$. We assume there are in total $n = 10$ followers. Each follower can only determine their own investment and thus $m = 1$. The entry of the correlation matrix $W \in \mathbb{R}^{n \times n}$ is generated from uniform distributions $W_{i,j} \sim U(0, 1)$ if $i \neq j$, and $W_{i,j} \sim U(1, 2)$ if $i = j$ to reflect the higher dependency on the self investments. We choose the risk aversion constant γ to be $\gamma = 0.01$.

H.7 Computing Infrastructure

All experiments except VI experiments were run on a computing cluster, where each node is configured with 2 Intel Xeon Cascade Lake CPUs, 184 GB of RAM, and 70 GB of local scratch space. VI experiments require a Knitro license and were run on a machine with i9-7940X CPU @ 3.10GHz with 14 cores and 128 GB of RAM. Within each experiment, we did not implement parallelization, so each experiment was purely run on a single CPU core.