Activity Allocation in an Under-Resourced World: Toward Improving Engagement with Public Health Programs via Restless Bandits

a dissertation presented
by
Jackson A. Killian
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 Jackson A. Killian

Activity Allocation in an Under-Resourced World: Toward Improving Engagement with Public Health Programs via Restless Bandits

Abstract

Artificial intelligence (AI) tools are being developed widely within society to improve decision-making, especially in resource-constrained settings like public health. However, developing effective AI tools for public health is complicated by scale, the time-varying and intervention-dependent nature of individuals' behavior, and scarcity — of intervention resources, historical data, and real-time observations. Many of these challenges are unaddressed in literature and can lead to poor outcomes if ignored in real-world AI support systems for public health. For example, previous works have modeled the problem of delivering interventions to improve engagement with health programs as a restless bandit, a widely-studied framework in which a set of stochastic arms are controlled by a planner with a limited intervention budget. However, these works do not account for the uncertainty that results from estimating the dynamics of stochastic arms from noisy observations and historical data. To address this, I introduce the first methods for computing *uncertainty-robust* restless bandit policies, across a range of assumptions on prior information and observability. I achieve this by designing novel approaches to efficiently search large policy and uncertainty spaces, e.g., a new multi-agent deep reinforcement learning paradigm in which a centralized *budget*-network communicates with per-arm *policy*-networks to learn globally optimal policies in an environment controlled by a regret-maximizing *adversary*-network. In union with this work, I advance art within the more computationally intensive generalization of restless bandits that finds policies which balance many *types* of interventions with unique costs and effects, e.g., a phone call vs. in-person visit; my works identify and exploit functional structures to design new algorithms that scale. This dissertation tackles several such challenges driven by identifying the key missing capabilities of interdisciplinary collaborators, especially tuberculosis healthcare workers and workers within a maternal health non-profit in India.

Contents

Author List

The following authors contributed to

- Chapter 1: Andrew Perrault, Milind Tambe.
- Chapter 2: Arpita Biswas, Sanket Shah, Milind Tambe.
- Chapter 3: Bryan Wilder, Amit Sharma, Vinod Choudhary, Bistra Dilkina, Milind Tambe.
- Chapter 4: Aditya Mate, Haifeng Xu, Andrew Perrault, Milind Tambe. Co-first author contribution by Aditya Mate.
- Chapter 5: Lily Xu, Arpita Biswas, Milind Tambe.
- Chapter 6: Arpita Biswas, Lily Xu, Shresth Verma, Vineet Nair, Aparna Taneja, Aparna Hegde, Neha Madhiwalla, Paula Rodriguez Diaz, Sonja Johnson-Yu, Milind Tambe. Co-first author contributions by Arpita Biswas, Lily Xu, and Shresth Verma.

Listing of figures

For Marie, Anthony, Ethel, and Albert, whose sacrifices gave me unimaginable opportunity.

Acknowledgments

Thank you to everyone who has supported me in reaching this stage of my research career and in writing this dissertation. First, thank you to my advisor, Milind Tambe, for five years of intriguing discussions, perpetual guidance, and an unrelenting pursuit for impact. Your strength of vision and conviction of action are truly your greatest secret weapons, and the related lessons I learned from you will benefit me throughout my career. I am grateful for your mentorship.

Thank you also to Andrew Perrault, who was an invaluable technical mentor to me especially in the early years of my degree. Without your careful, exhaustive, and instructive guidance, reaching this stage would surely have been a far more arduous and winding affair. I am grateful for your tutelage and your stalwart friendship.

Thank you to my dissertation committee members, Elena Glassman, Maimuna Majumder, and David Parkes for your support and critical evaluation of my research and for thought-provoking philosophical discussions about my future career. Thank you also to my qualification committee members Finale Doshi-Velez, Yiling Chen, and David Parkes for your early examination of my progress as a researcher as well as your indelible words of encouragement. And thank you to my many other mentors through the years, formal and informal, who have been generous with their time and wisdom and who helped me reach this stage, especially Bistra Dilkina, Puneet Batra, Philip Nelson, Manish Jain, Amit Sharma, and Brian DeRenzi.

Thank you to all of my fellow co-authors and collaborators each of whom was immensely helpful in shaping or refining the work in these chapters. I have learned a great deal from each of you. Thank you also to my fellow lab mates and peers for countless hours of academic questioning, communal coffees and dinners, table tennis, long walks, and commiseration.

Thank you to everyone whom I have had the privilege to mentor. It has been a great joy to pass on some hard-won lessons, formal and informal, to future scientists for whom I have a great deal of respect. I have learned so much from each of you, and your intellect, enthusiasm, and resilience have been a consistent and remarkable source of inspiration for me.

Finally, thank you to my family. My partner, my rock, Jessica Stubbing, has been an unwavering source of love, reassurance, encouragement and support. From excitedly pondering our future goals, to proofreading emails, to listening to me practice yet another presentation, to packing me a lunch before a long commute, to being patient with me promising that writing up some results would take "only 30 more minutes", to being there for me through my successes and failures, you have been a

truer partner than I could have ever asked for. I am so grateful for you. And thank you for inspiring me every day with your passionate, selfless, and unrelenting pursuit to improve mental health around the world. It has been a joy and a privilege to support each other through our doctoral degrees. To my parents, thank you for giving me countless opportunities to explore my interests, diligently cultivating them, encouraging me to achieve, and instilling in me the greatest resilience. Thank you to Mom for your fierce lessons in love and courage. Thank you to Dad for your ineradicable teachings on honor and gratitude. To my brother Dan, you are my first and closest friend, and you have an unparalleled ability to help me lift my head up in even the hardest times. Thank you for sharing your incredible strength with me, for your constant guidance, and for always believing in me.

0

Introduction

The goal of my research is to enable robust and equitable health decision-making in real-world multi-agent systems. Moreover, the specific research questions I tackle are driven by identifying the key missing capabilities of interdisciplinary health organizations with whom I collaborate. My body of work is evidence that such a research model is a fruitful means to jointly produce disciplinary advances and tools with potential for real-world impact.

In this dissertation, I tackle methodological challenges focused on one key goal: developing ar-

tificial intelligence (AI) to improve the delivery of public health programs in low-resource settings. Previously, this has been in varied contexts, e.g., training AI to detect disease from low-cost sensors like chest x-rays [131] or voice recordings [173], training AI to predict key health markers from satellite imagery [20], and training AI to forecast the progression of pandemics [7]. Each of these studies produce AI tools which may provide public health workers with additional intelligence at a moment in time. However, they leave unanswered how to convert that intelligence into sustained improvements in decision-making. This forms the core view of this dissertation, which aims to develop AI tools that translate limited information into *effective sequential decision recommendations* for public health workers whom are placed at the center of a challenging, ongoing, resource-constrained decision-making problem to manage their patients' evolving health behavior.

I believe that working closely with domain experts and stakeholders is critical to developing responsible AI tools that respond to the actual needs of on-ground practitioners. In pursuit of this, in developing this dissertation, I collaborated with two public health organizations in India, the first focused on eliminating tuberculosis (TB) and the second on improving maternal and child health. Our primary partner in combating TB was the Mumbai office of the National TB Elimination Program (NTEP) in India.* The NTEP provides free antibiotic medication to patients diagnosed with TB throughout their 6-month to 2-year treatment courses. Health workers with the NTEP help patients navigate treatment, refill their medications, and remain adherent to their 2-5 pill daily regimens, complicated by many factors, e.g., medication side-effects that make it challenging to work. Each NTEP health worker in Mumbai may manage hundreds of patients.

Our partner in improving maternal health was the non-profit, ARMMAN [10]. They provide a range of free services to democratize access to evidence-based, life-saving health information for new and expecting mothers across India. Specifically, we partnered in support of their program, mMitra,

*Previously, the NTEP was named the Revised National Tuberculosis Control Programme (RNTCP). The name was changed in 2020.

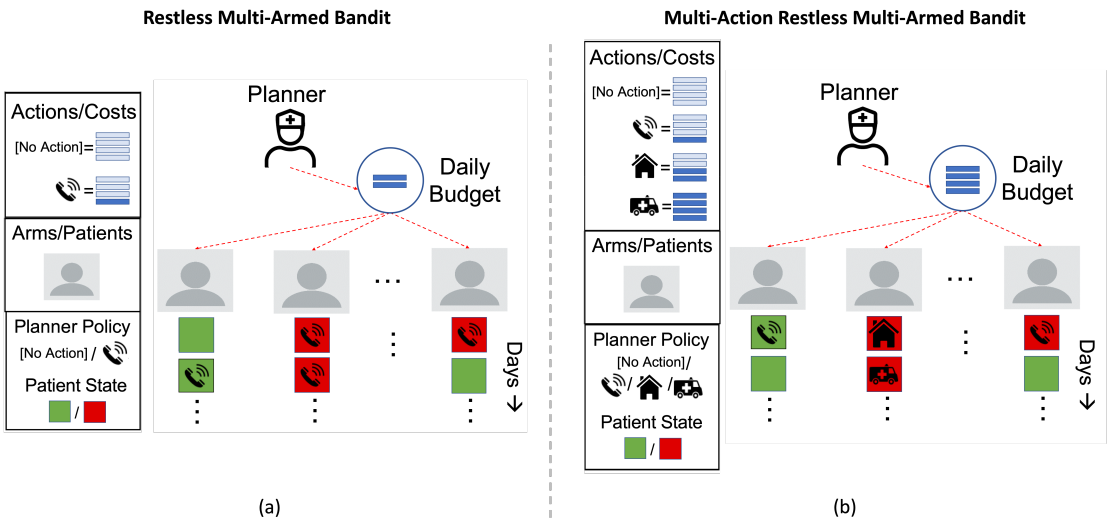


Figure 1: (a) A restless multi-armed bandit (RMAB). One planner (e.g., health worker), acts on a set of arms (e.g., patients), each with their own state (e.g., green=adherent, red=non-adherent), subject to a per-timestep budget constraint (e.g., two calls per day). In an RMAB, there are only two types of actions, i.e., "no action" and "action" (e.g., "call" in the figure). **(b)** The multi-action generalization of RMABs (MARMABs), in which the planner has many types of actions to choose from, each with varied costs/effects, subject to one per-timestep budget constraint. In both RMABs and MARMABs, there is always at least one action type that has zero cost so that the budget constraint is always satisfiable.

which provides free twice-weekly automated voice recordings to new and expecting mothers with information tailored to their gestational age or the newborn's age. At any given time, tens of thousands of mothers are enrolled in the program, but the program faces waning engagement over time for 30-40% of mothers. To address this disengagement, ARMMAN employs a team of workers who deliver targeted service calls to mothers, e.g., explaining the value of the calls or correcting technical issues for the mother. The team has capacity to contact hundreds of mothers per week.

For both of our partners, health workers share an arduous task in common: deciding how to allocate their limited time and resources each week across large cohorts of patients such that engagement with their evidence-based programs is maximized. Specifically, I study these problems through the lens of restless multi-armed bandits (RMABs) [163], a sequential decision-making framework in which a centralized planner (e.g., health worker) manages a set of N arms (e.g., patients), each with

a state following a stochastic process (e.g., adhering or not adhering to medication each day). The planner manages the arms by selecting a subset for intervention each time period. This process is visualized in Figure 1(a). The restricted budget on interventions couples the evolution of the arms' states, making the setting a complex combinatorial problem. As I applied ideas from these modeling frameworks across the TB and maternal health domains — through in-person workshops with stakeholders, presentations at interdisciplinary conferences, and regular virtual meetings with collaborators across continents — I identified two major recurring methodological challenges that form the basis of this dissertation work, and they are concretized in the following problem statement.

0.1 Problem Statement and Contributions

In this dissertation I address the question:
How can we jointly incorporate (i) resource heterogeneity and (ii) uncertainty in restless bandits for improving public health program engagement?

In addressing this question, my key contributions are (1) optimization algorithms for scaling the joint planning of heterogeneous intervention resources (2) machine learning algorithms for learning arm dynamic models from confounded data and with combinatorial budget constraints (3) a formal model for a new *collapsing* type of partial observability common across public health engagement problems, with a scalable algorithm for computing optimal policies and (4) algorithms for computing minimax regret-robust policies over interval model uncertainties that derive from noisy historical data, including a new multi-agent reinforcement learning paradigm.

0.2 Summary

0.2.1 Scaling Heterogeneous Action Planning

Especially in the TB care setting, health workers have more than one option for how they intervene on patients, e.g., texting, calling, traveling to visit a person in their home, or even escalating their case to a supervisor. However, the majority of RMAB works are restricted to planning binary actions, i.e., "to act or not to act". There is limited previous work studying the generalization which handles heterogeneous intervention resources, and these previous works have only been specialized to certain sub-problems. To address this, I derive multiple algorithms for use on *general* Multi-action RMABs (MARMABs, visualized in Figure 1(b)) using Lagrangian relaxation techniques, leading to several contributions. First, I develop algorithm which adapts to the inherent complexity of a problem's data via bound optimization, achieving excellent scale-up by leveraging a key insight that often only a subset of arms are involved in the optimal policies of real-world planning problems. The bounds leverage underlying problem convexity to quickly and provably converge to the well-performing Lagrange policy. I also develop SampleLam, a fast sampling technique for estimating the Lagrange policy, and derive a concentration bound to investigate its convergence properties. I then derive best and worst case computational complexities for our algorithms as well as our main competitor, showing improvements in the best case. Finally, I provide experimental results comparing the algorithms to baselines on simulated distributions, including one motivated by the TB intervention task, where a worker decides between texts, calls or an expensive "escalation" action. Our approach achieves significant, up to ten-fold speedups over more general methods without sacrificing performance. This demonstrates the importance of multi-action planning, as well as its potential for use in real-world-scale problems. Chapter 1 lays the groundwork for the multi-action setting studied in this dissertation. Chapters 2 and 5 build off these principles to enable online learning and planning under model uncertainty in the challenging multi-action generalization of RMABs.

0.2.2 Learning from Confounded History and Constrained Exploration

In the above work, I establish principles for solving MARMABs assuming arm dynamics are known, which itself is PSPACE-Hard[122]. However, in the real-world, arm dynamics must be estimated from data (i) via real-time exploration and/or (ii) via estimation from observational data.

For (i) I develop two approaches, both of which reason carefully about restricted budgets *while learning* in order to converge to effective policies. The first tackles this problem by learning per-arm and per-action indexes that capture resource efficiency, using a two-timescale Q-learning and convex parameter estimation procedure. This builds from a method for learning similar indexes in the binary-action setting — I derive a generalized update rule and convergence proof for the multi-action case. However, this approach requires restrictive assumptions on problem structure and is slow to converge. By contrast, my second approach directly learns the well-performing and more general Lagrange policy for MARMABs by learning to minimize the Lagrange bound through a variant of Q-learning. The approach involves an approximation step that offers planners a tradeoff between accuracy and efficiency, backed by guarantees. Both methods (1) take an epsilon-greedy exploration approach, where I establish a good heuristic for selecting random feasible combinatorial actions and (2) utilize a replay buffer that weights observed samples by the inverse of their action cost which improves learning efficiency.

For (ii) I study an observational dataset on daily TB adherence data, enabled by Digital Adherence Technologies (DATs), an increasingly popular method for verifying patient medication adherence. I analyze a dataset from Mumbai with 17,000 patients and 2.1M dose records. I lay the groundwork for learning from this real-world data, giving a method for avoiding the effects of unobserved interventions in training data used for machine learning. Specifically, the data is missing records of health worker interventions which, if ignored when training a classifier, can produce antithetical recommendations. I develop a screening procedure that levereges domain knowledge

from our NTEP partners to carefully split the training dataset where intervention signal may be present, to train classifier that avoids this antithetical behavior. I then construct a deep learning model, demonstrate its interpretability, and show how it can be adapted and trained in three different clinical scenarios to better target and improve patient care.

0.2.3 Planning Under Model and Observation Uncertainty

In the above works I give methods for handling uncertainty in the presence of uninformative priors, or under specific structures in which confounders can be screened out. I now give more general methods for incorporating prior information when planning, which is especially important when planning horizons are short, i.e., there are fewer opportunities for learning in real-time.

I first handle uncertainty due to partial observability. In many cases, health workers can observe the engagement of patients when they act, but otherwise the patient engagement is unknown, e.g., when DATs are unavailable. However, using historical information, we can help workers reason about probabilities of engagement *between* observations that can improve policies. In a co-first author capacity, I propose and study Collapsing Bandits, a new RMAB setting which follows the above structure. I give conditions under which the new model admits the asymptotically optimal Whittle index policy, then build fast algorithms for computing the policy in a closed form. The proofs and algorithm hinge on the monotonic evolution of the planner's "belief" about the patient's engagement state, i.e., the key structure that captures the planner's uncertainty between observations.

I then give the first methods for handling model uncertainty in RMABs and MARMABs. Nearly all restless bandit techniques assume stochastic dynamics are precisely known. However, in many real-world settings, dynamics are estimated with significant *uncertainty*, e.g., via historical data, which can lead to bad outcomes if ignored. To address this, I develop an algorithm to compute minimax regret–robust policies for the MARMAB setting. The approach uses a double oracle

framework (oracles for *agent* and *nature*), which is often used for single-process robust planning but requires significant new techniques to accommodate the combinatorial nature of MARMABs. Specifically, I design a deep reinforcement learning (RL) algorithm, DDLPO, which tackles the combinatorial challenge by learning an auxiliary "λ-network" in tandem with per-arm policy networks, greatly reducing sample complexity, with guarantees on convergence. DDLPO, of general interest, implements our reward-maximizing agent oracle. I then tackle the challenging regret-maximizing nature oracle, a non-stationary RL challenge, by formulating it as a multi-agent RL problem between a policy optimizer and adversarial nature who selects worst-case environments from the continuous uncertainty set over all arm dynamics models. I demonstrate through simulated experiments,that the approach could improve outcomes in the 100-arm problems sizes of the scale encountered by our TB partners.

However, we also wish to implement this system for our maternal health partner organization, which has 100,000s of arms. This requires significant new techniques to search the massive policy and uncertainty spaces. In a co-first author capacity, to jointly handle the realities of *large scale* and *model uncertainty*, I first instantiate a grouping abstraction, which combines arms with similar behavior into a single planning unit. I then derive a new per-arm notion of regret (Whittle index regret) to further decouple the problem and give a computable guide for searching the massive uncertainty and policy spaces. Using this notion, I establish a new weighted index heuristic for the planner oracle and a binary quadratic program for optimizing Whittle indices over uncertain transition probability intervals for the adversary oracle. I then prove a key theoretical result that planning over grouped arms achieves the same minimax regret–optimal strategy as planning over individual arms, under a technical condition. Finally, using real-world data from ARMMAN, I show that the approach produces robust policies that reduce minimax regret by up to 50%, halving the number of preventable missed voice messages to connect more mothers with life-saving maternal health information.

0.3 Dissertation Outline

This dissertation is organized as follows: Chapter 1 introduces RMABs and the heterogeneous action setting with techniques for scaling and adapting to real-world data. Then, chapters 2 and 3 introduce methods for learning from real-world data to improve planning. Next, chapters 4-6 discuss planning under various forms of uncertainty. Finally, chapter 7 contains a conclusion and future directions. Background and related works are given in each chapter.

0.4 A Note on the Use of AI Tools from this Dissertation

As has been noted countless times, all models are wrong, but some are useful[22]. The tools in this dissertation are best seen not as prescriptive, but as descriptive; a way to transform large, messy datasets into some specific view of the world paired with helpful insights about how to positively influence that world. While we take great care in these chapters to find evidence that our models are useful, they are inevitably imperfect, since they cannot capture all the rich complexities of the real world. For example, they do not capture the tone of voice of a TB patient during intake, nor the possibility that two patients may be friends supporting each other's adherence habits. Thus it is important to view the tools' recommendations not as inherently authoritative, but rather as an output from an additional tool in a larger toolbox to support decision making. Workers interacting with the recommendations should have the opportunity to contextually calibrate their trust in the recommendations[15,14]: when are the recommendations useful and when do they fail? How do the recommendations interact with insights only captured by human interaction?

Additionally, at a higher level, viewing our AI tools as descriptive can help take a critical view of the systems they model. Our tools may recommend intervention to a certain patient, but what are the characteristics of the underlying model that led to that? What are the characteristics that led to another patient *not* being recommended for intervention? If we ignore resource constraints

and simulate providing interventions to *all* patients, which groups would be left behind, based on our models? Then, what data led to our models reaching that conclusion and, critically, what are the realities of the real-world system that generated that data? We can advocate for changes in real-world systems as they exist today using our tools to generate evidence regarding the answers to such questions; e.g., to set research and policy agendas that seek justice in healthcare systems, where all patients affirmatively receive what they need to achieve the highest level of health [128].

1

Beyond "To Act or Not to Act": Fast Lagrangian Approaches to General Multi-Action Restless Bandits

1.1 Introduction

RMAB, the state-based generalization of classic Multi-Armed Bandits [152], have been studied extensively for solving a diverse set of problems including machine replacement [54,137], sensing and wireless network scheduling [13,177,5,32,112], job scheduling [175,61,172] anti-poaching patrol scheduling [130], and healthcare [92,103,17]. In classic Multi-Armed Bandits, a planner must select k out of N arms on which to act for each of L rounds in a way that maximizes reward produced by the arms. In RMABs, additional complexity is introduced in that the reward on each arm depends on the action as well as an internal state that evolves according to an independent two-action Markov Decision Process (MDP). It has been shown that this problem is, in general, PSPACE-hard to solve exactly [122], but highly effective heuristics are known to exist [163,16].

However, a critical limitation of RMAB frameworks is they only allow for 2 actions: act or not act. This is restrictive for many real-world cases where planners have various actions at their disposal with varying degrees of cost and effect. For example, a system manager may need to balance preventative maintenance, full repair, and job scheduling each with different costs and effects on throughput [26]. In anti-poaching, the planner could allocate different levels of patrol effort to differ-

ent targets, where more effort has higher cost and higher deterrent effect on poachers[119]. In public health, a community health worker could have several options for intervening with a patient, such as calling, visiting in person, or escalating patients to a more intense treatment[169]. Traditional RMABs simply cannot model these complexities, restricting planners to a world where their only choices are to, e.g., call or not call. Rather, the planner needs to simultaneously optimize the use of all of the tools in their toolbelt each day, subject to a per-day time or cost budget B. This process is visualized in Fig. 1.

To model such problems, we consider an under-examined generalization of RMABs that allow for multiple action types per arm, which we call Multi-Action RMABs (MARMABs). Previous work has considered extending the classical RMAB notion of *indexability* and corresponding *index policies* to MARMABs[52]. In both traditional and Multi-action RMABs, index policies are desirable because: (1) they decompose the problem in a manner that scales well and (2) when indexability holds, they are asymptotically optimal[161,63]. However, both deriving index policies and verifying indexability is notoriously difficult, and largely requires special problem structure[54,52]. Our goal is thus to develop fast, well-performing policies for a broader class of MARMABs where no structure is assumed and indexability cannot be readily verified. We bypass the task of deriving index policies by taking a more general Lagrangian relaxation approach that leads to an auxiliary problem of computing a policy that minimizes the Lagrange bound. Computing this "Lagrange policy" is desirable because it recovers the performance of the index policies when they exist, but is readily computable regardless of problem structure (see appendix A for additional discussion).

The Weakly Coupled MDP (WCMDP) literature offers a method to compute the Lagrange policy for WCMDPs. Here, we recognize WCMDPs as a generalization of MARMABs and identify that this approach can be used to compute the Lagrange policy for MARMABs. However, the approach relies on solving a large linear program (LP) that scales quadratically in the number of arms and states. Setting up and solving this LP quickly becomes infeasible for large problem sizes as we

show later in experiments. To address this issue, we investigate and utilize basic structural properties of general MARMABs to create scalable algorithms for computing the Lagrange policy on any MARMAB problem, leading to the following contributions:

(i) Bound optimization algorithm: We develop BLam, an iterative bound optimization method for computing the Lagrange policy. BLam leverages problem convexity to derive progressively tighter upper and lower bounds on the Lagrange policy via a series of small LPs. We provide key technical results that prove this method converges to the policy that minimizes the Lagrange bound and provide experimental evaluation of its runtime on various distributions.

(ii) Sampling algorithm: We develop a sampling-based algorithm, SampleLam, which trades off the guarantees of BLam for speed. SampleLam chooses a random subset of arms, rapidly computes a statistic about the desirability of allocating budget to each arm, then combines the statistics to construct an estimated Lagrange policy for the full problem. We derive a concentration bound to prove the method converges, then use insights from the bound to inform how the algorithm carries out sampling.

(iii) Complexity Results: We derive best and worst case computational complexities for our methods as well as our main competitor. Our exact algorithm, BLam, achieves $\approx \sqrt{N}$ improvement and SampleLam achieves a factor of N improvement in the best case;

(iv) Experimental evaluation: We compare our algorithms to baselines on synthetic distributions with different underlying structure, including one motivated by a real-world public health challenge. Our algorithms scale up to ten times better than a more general baseline without sacrificing performance, and readily adapt to each problem with minimal tuning. Thus our work newly makes available multiple avenues for computing well-performing policies on new MARMABs at scale, without the need for the user to first arduously derive a problem-specific index policy.

1.2 Related Work

Previous work extends the traditional RMAB notion of indexability to MARMABs[52,63]. However, their analysis is restricted to a subclass of MARMABs with special monotonic structure, whereas we build algorithms for general MARMABs. "Superprocesses" are an alternative multi-action extension where a primary planner distributes a limited set of sub-planners who act on arms without constraint[101,176,154]. This structure does not generally apply to MARMABs since they do not constrain the number of agents that can be acted on each round. Very recent work[108] designs a Monte-Carlo rollout approach for estimating traditional and multi-action RMAB policies when a restricted set of "threshold" policies are optimal, but our algorithms do not assume this structure.

Also related are WCMDPs in which a planner operates N independent MDPs subject to a set of arbitrary constraints over actions.[110] derive methods for handling "global" resource constraints over all rounds, whereas we address round-by-round constraints.[60], the main baseline we compare against, derive a Lagrangian relaxation on the general form of a WCMDP and give an LP for minimizing the Lagrange bound. In contrast, we leverage the single constraint nature of MARMABs to greatly speed up the computation of the Lagrange bound compared to[60].[4] give an approximate dynamic programming method that achieves a tighter bound and better performing policies than the Lagrange approach to WCMDPs. However, it scales exponentially, restricting it to small problem sizes.[55] develop a Lagrange approach for solving WCMDPs with MDPs that grow exponentially in problem parameters, restricting them to approximation techniques. In contrast, we develop a method that exactly computes the Lagrange policy.

Finally, our work is related to a large body of work developing Lagrangian methods for solving traditional RMABs[163,54,121,16]. We generalize these settings to allow for multiple actions. Moreover, the methods we develop here reduce in the binary action case to the widely-used, well-performing, Whittle index policy[163,54].

1.3 Preliminaries

A MARMAB consists of a set of N arms, each associated with a Markov Decision Process (MDP)[129]. An MDP $\{\mathcal{S}, \mathcal{A}, r, T, \beta\}$ consists of a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a state-dependent bounded reward function $r : \mathcal{S} \rightarrow \mathbb{R}$, a transition function T, where $T(s, a, s')$ gives the probability of transitioning to state s' when action a is taken from state s, and a discount factor $\beta \in [0, 1)$. An MDP *policy* $\pi : \mathcal{S} \rightarrow \mathcal{A}$ maps states to actions. The long-term *discounted reward* starting from state $s_0 = s$ is defined as

$$R_\beta^\pi(s) = E\left[\sum_{t=0}^{\infty} \beta^t r(s_{t+1} \sim T(s_t, \pi(s_t), s_t'))|\pi, s_0 = s\right] \tag{1.1}$$

Each arm i in a MARMAB is an MDP with an action set $\mathcal{A}^i$ of size M^i and corresponding action cost vector $\boldsymbol{C}^i \in \mathbb{R}^{M^i}$. We assume all action sets and costs are the same for all arms (and henceforth drop the subscript i), but all techniques in this chapter extend in a straightforward manner to general action sets and costs. Without loss of generality, we assume that the elements c_j of $\boldsymbol{C}$ are ordered ascending. Also, to align with the standard bandit assumption that an arm can be "not played" at no cost, we set $c_0 = 0$. Each round, the planner must select one action for each of the N arms such that the sum cost of all actions do not exceed a budget B. Formally, the planner must choose a decision matrix $\boldsymbol{X} \in \{0, 1\}^{N \times M}$ with elements denoted $x_{i,j}$ such that

$$\sum_{j=0}^{M-1} x_{i,j} = 1 \; \forall i \in 0 \ldots N-1 \qquad \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} x_{i,j} c_j \leq B \tag{1.2}$$

where the first constraint enforces one action per arm and the second enforces the budget. The planner's goal is to maximize their discounted reward across the arms over time, subject to these constraints. Let $\boldsymbol{s} = [s^0, s^1, \ldots, s^{N-1}]$ represent the vector of all arm states. The planner's goal can be

represented by the following constrained Bellman equation

$$J(\boldsymbol{s}) = \max_{\boldsymbol{X}}\{\sum_{i=0}^{N-1} r^i(s^i) + \beta E[J(\boldsymbol{s}')|\boldsymbol{s}, \boldsymbol{X}]| \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} x_{i,j}c_j \leq B\} \tag{1.3}$$

While Eq. 1.3 could be solved directly via value iteration, $J(s) \in \mathcal{S}^N$, and the number of feasible actions over which to take the max for each $J(\boldsymbol{s})$ is also exponential in N, making this approach intractable for non-trivial problem sizes. The key insight, though, is that the value functions and actions are only coupled due to the shared budget constraint over all arms. Therefore to simplify the problem, we relax the budget constraint and add it as a penalty to the objective with a Lagrange multiplier λ as follows:

$$\begin{aligned} &J(\boldsymbol{s}, \lambda) = \\ &\max_{\boldsymbol{X}}\{\sum_{i=0}^{N-1} r^i(s^i) + \lambda(B - \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} x_{i,j}c_j) + \beta E[J(\boldsymbol{s}')|\boldsymbol{s}, \boldsymbol{X}]\} \end{aligned} \tag{1.4}$$

The value functions then decouple as desired, giving:

$$J(\boldsymbol{s}, \lambda) = \frac{\lambda B}{1-\beta} + \sum_{i=0}^{N-1} V^i(s^i, \lambda), \text{ where } \forall i, \tag{1.5}$$

$$V^i(s^i, \lambda) = \max_{a_j^i \in \mathcal{A}}\{r^i(s^i) - \lambda c_j + \beta \sum_{s^{i\prime}} T(s^i, a_j^i, s^{i\prime}) V^i(s^{i\prime}, \lambda)\} \tag{1.6}$$

See[4] for a complete proof. Notice that for a given value of λ, Eq. 1.5 can be solved using a fast method like value iteration to solve for the individual V^is, where V^i and its corresponding actions are now in $\mathcal{S}$ and $\mathcal{A}$, respectively. However, the choice of λ will be critical when using the resulting value functions to derive policies for our bandits. For instance, $\lambda = 0$ would correspond to ignoring the budget constraint while planning which clearly will not be optimal in general. Alternatively, as

$\lambda \rightarrow \infty$, the optimal policy in each value function is to never act since all actions will have effectively infinite cost except for $c_0 = 0$. To gain insight about how to set the value of λ we recast the problem as an LP, rewriting Eq. 1.5 by leveraging the known LP solution to the value function [129]:

$$
\begin{aligned}
J(\boldsymbol{s}, \lambda) = & \min_{V^i(s^i,\lambda),\lambda} \frac{\lambda B}{1-\beta} + \sum_{i=0}^{N-1} \mu^i(s^i) V^i(s^i, \lambda) \\
& \text{s.t. } V^i(s^i, \lambda) \geq r^i(s^i) - \lambda c_j + \beta \sum_{s^{i\prime}} T(s^i, a^i_j, s^{i\prime}) V^i(s^{i\prime}, \lambda) \\
& \forall i \in \{0, ..., N-1\}, \;\; \forall s^i \in \mathcal{S}, \;\; \forall a_j \in \mathcal{A}, \text{ and } \lambda \geq 0
\end{aligned}
\tag{1.7}
$$

Where $\mu^i(s^i) = 1$ if s^i is the start state for arm i and is 0 otherwise. That we minimize over λ and that $\lambda \geq 0$ is a classic Lagrangian result, motivated by making $J(\boldsymbol{s}, \lambda)$ a tight-as-possible upper bound on $J(\boldsymbol{s})$. Intuitively, and matching how problem-specific index policies have been derived in previous work [163,52], we want to derive a policy from the V^is that provide the tightest bound on $J(s)$. So that our algorithms can generally apply to any MARMAB, our approaches will solve Eq. 1.7 in its general form.

The above derivation was first given by [60] for WCMDPs, and as suggested therein, clearly one can directly solve Eq. 1.7 using any LP solver. However, Eq. 1.7 has $N|\mathcal{S}| + 1$ variables and $N|\mathcal{S}||\mathcal{A}|$ constraints. Further, the current lowest known computational complexity for solving an LP is $\mathcal{O}(n^{2+\frac{1}{18}})$, where n is the number of variables [69], implying that directly solving Eq. 1.7 has computational complexity $\approx \mathcal{O}(N^2|S|^2)$ (derived in section 1.4). The key to our approach will be separating the computation of the λ that minimizes Eq. 1.7, henceforth λ_{min}, and the corresponding V^is that solve Eq. 1.7 in a way that provides vast speedups. Herein we derive exact and heuristic methods for computing λ_{min}, each of which has an improved best case complexity in N by a factor of $\sqrt{N}$ or better.

1.4 Bound Optimization With BLam

BLam is our exact approach to computing the Lagrange policy. We first give an overview, noting theorems where relevant that are derived in the next section. The main idea is rooted in the form of the functions $V^i(s^i, \lambda)$ in Eq. 1.7, visualized in blue in Fig. 1.1. To exactly compute Eq. 1.7 requires adding $|\mathcal{S}||\mathcal{A}|$ constraints and $|\mathcal{S}|$ variables to the LP for each of the N arms' value functions $V^i(s^i, \lambda)$. Instead, we will build special approximations to each $V^i(s^i, \lambda)$ that are represented in the LP each with just one variable and a constant number of constraints, achieving vast speedups. The approximations are constructed by rapidly testing for the slope of $V^i(s^i, \lambda)$ at various test points λ_{test} using value iteration, then creating a piecewise linear combination of the slopes. The key is we construct two special types of approximations: one that upper bounds the slope of $V^i(s^i, \lambda)$ and one that lower bounds it, shown in Fig. 1.1 in green and red, respectively.

We then use the insight that the $V^i(s^i, \lambda)$ in Eq. 1.7, are indeed convex decreasing functions of λ (Prop. 1.4.1), implying that Eq. 1.7 is minimized when the combined per-unit *decrease* to the objective brought by the convex $V^i(s^i, \lambda)$ functions is equal to or less than the constant per-unit *increase* to the objective brought by $\frac{\lambda B}{1-\beta}$. In other words, λ_{min} is the point where the negative sum of slopes of $V^i(s^i, \lambda)$ is equal to $\frac{B}{1-\beta}$ (Prop. 1.4.2). Crucially, if we replace any $V^i(s^i, \lambda)$ with a convex function with strictly more negative slope (i.e., a lower bound), the value of λ at which the negative sum of slopes equals $\frac{B}{1-\beta}$ could only increase, giving an upper bound on λ_{min}. The converse also holds, i.e., replacing with upper bound convex functions gives a lower bound on λ_{min} (Thm. 1.4.3). This constitutes the core tradeoff in our approach: the more $V^i(s^i, \lambda)$ are replaced with approximations in the LP, the faster it will execute, but the looser the bounds will be. We handle this by first "bounding out", i.e., replacing $V^i(s^i, \lambda)$ with its approximation, all but a small number K processes to get loose bounds on λ_{min} rapidly. We then iteratively add back $V^i(s^i, \lambda)$s to the LP until the bounds on λ_{min} are with a pre-specified ε. With minimal tuning, the test points can be set to create

tight enough bounds that BLam will converge after only a small number of iterations, leading to great speed increases.

The algorithm proceeds in two parts. In BLamPrecompute, given in Alg. 1.4.1, we compute the upper and lower bound approximations of of the arms, passing in the MDP parameters of the arms, along with a list G of points λ_{test} at which to approximate the slopes. VI in Alg. 1.4.1 denotes value iteration and $\mathcal{U}/\mathcal{L}$ will contain the pieces of the piecewise upper and lower bounds for $V^i(s^i, \lambda)$ for all arms and states. BLamPrecompute runs once at the beginning of simulation.

BLam, given in Alg. 1.4.2, runs on each round of the MARMAB to compute λ_{min} for the current set of arm states $\boldsymbol{s}$ (line 2 of Alg. 1.4.2 selects the bounding functions for the current state of each arm). Using the piecewise bounded versions of $V^i(s^i, \lambda)$, it constructs a special LP, BLamLP, given in Eq.1.8 below, that produces upper and lower bounds on λ_{min} by replacing $V^i(s^i, \lambda)$ with their bounded counterparts. It loops, replacing successively more $V^i(s^i, \lambda)$ in lieu of their bounded forms, until the resulting bounds on λ_{min} are within ε. BLam terminates by running one final value iteration with the appropriate λ_{min}, the result of which solves Eq. 1.7 *without constructing or solving the full LP, leading to vast speed ups*. The resulting value functions will be used to construct a final policy in section 1.6.

1.4.1 BLam: Derivation

To bound the slope of $V^i(s^i, \lambda)$, we rely on it having a convex form.

Proposition 1.4.1. *$V^i(s^i, \lambda)$ is convex decreasing in λ, and as $\lambda \to \infty$, $\frac{dV^i(s^i,\lambda)}{d\lambda} \to 0$*

Proof. This follows directly from Eq. 1.6, but can be shown via induction that since $V^i(s^i, \lambda)$ is a max over piecewise linear convex functions of λ, it is also piecewise linear convex, and since $\lambda c_j \geq 0$, it must be weakly decreasing in λ. Furthermore, since $c_0 = 0$ in MARMABs, as $\lambda \to \infty$, the one time charge λc_j of any action a_j s.t. $j > 0$ becomes greater than any long-term achievable

Algorithm 1.4.1: BLamPrecompute

Data: T, R, C, N, G, β

$\mathcal{D} = []$; `// hold slopes at each arm test point`

$\varepsilon_0 = 1e\text{-}3$;

for $i = 1, \ldots, N$ **do**

  for $j = 1, \ldots, |G[i]|$ **do**

    $\lambda_{test} = G[i, j]$;

    $R_\lambda, R_{\lambda+\varepsilon_0} = R[i]$;

    for $x \in 1, \ldots, |C|$ **do** subtract action costs

      $R_\lambda[x] -= \lambda_{test} * C[x]$;

      $R_{\lambda+\varepsilon_0}[x] -= (\lambda_{test} + \varepsilon_0) * C[x]$;

    $\mathcal{D}[i, j]$ = (VI($T[i], R_{\lambda+\varepsilon_0}, \beta$) - VI($T[i], R_\lambda, \beta$))/$\varepsilon_0$

$\mathcal{U}, \mathcal{L}$ = BuildBounds($\mathcal{D}$);

return $\mathcal{U}, \mathcal{L}$

reward, therefore the optimal policy must always choose not to act. At that point, $V^i(s^i, \lambda) = E[\sum_{t=0}^{\infty} \beta^t r(s) | \pi(a) = 0, \forall a]$ which does not depend on λ. □

Let λ_u (λ_ℓ) correspond to the λ which solves Eq. 1.7 when $V^i(s^i, \lambda)$ are replaced in the objective by $\mathcal{L}$ ($\mathcal{U}$). Note that the lower bound functions $\mathcal{L}$ will be used to derive *upper bounds* on the value of λ_{min} and vice versa.

Next, we give a helpful intermediate result.

Proposition 1.4.2. *The optimal solution to Eq. 1.7 will be found at the value of λ in which the negative sums of the slopes of $V^i(s^i, \lambda)$ w.r.t. λ become less than or equal to $\frac{B}{1-\beta}$.*

Proof. Assume λ^* corresponds to an optimal solution to Eq. 1.7 and the negative sums of the slopes of convex decreasing $V^i(s^i, \lambda)$ are greater than $\frac{B}{1-\beta}$. Then λ^* can be increased by ε and the objective value would decrease, i.e., $J(\boldsymbol{s}, \lambda^* + \varepsilon) < J(\boldsymbol{s}, \lambda^*)$ giving a contradiction. □

We now can prove our main result:

Theorem 1.4.3. $\lambda_\ell \leq \lambda_{min} \leq \lambda_u$

Algorithm 1.4.2: BLam

Data: $T, R, C, N, B, \beta, s, G, \mathcal{U}, \mathcal{L}, \varepsilon$, kStep

```
/* Only need bounds for current arm states */
```

GetCoeffsForState($\mathcal{U}$, $\mathcal{L}$, s);

Sort($\mathcal{U}$, $\mathcal{L}$, T, R);

st = PickStart($\mathcal{L}$, $\sqrt{N}$);

for $k \in$ [*st, st+kStep, ..., N*] **do**

  $\lambda_u = \text{BLamLP}(T[:k], R[:k], B, C, \beta, s, \mathcal{L})$;

  $\lambda_\ell = \text{BLamLP}(T[:k], R[:k], B, C, \beta, s, \mathcal{U})$;

  if $\lambda_u - \lambda_\ell \leq \varepsilon$ **then** break ;

$V(i, s) = []$ `// ` $Nx|\mathcal{S}|$ `array to hold value functions`

$\lambda_{min} = (\lambda_u - \lambda_\ell)/2$;

for $i = 1, \ldots, N$ **do**

  $R_\lambda = R[i]$;

  for $x \in 1, \ldots, |C|$ **do** subtract action costs

    $R_\lambda[x] -= \lambda_{min} * C[x]$;

  $V[i] =$VI$(T[i], R_\lambda, \beta)$;

return V

Proof. The proof is best seen by considering λ_{min} which solves $J(\boldsymbol{s}, \lambda)$, i.e., Eq. 1.7. We start with $\lambda_{min} \leq \lambda_u$: Let $\mathcal{V}$ denote the set of $V^i(s^i, \lambda)$ in the objective of Eq. 1.7. Further, let $\mathcal{V}^b$ denote the set of $V^i(s^i, \lambda)$ which will be replaced by $\mathcal{L}^b \subset \mathcal{L}$. Now replace all $\mathcal{V}^b$ with their corresponding $\mathcal{L}^b$, name this new LP $J^{\lambda_u}(\boldsymbol{s}, \lambda)$ and name its optimal solution λ_u. By definition, at all values of λ, the slope of $V^i(s^i, \lambda)$ is greater than the slope of $\mathcal{L}^b$. Thus, at λ_{min}, the negative sums of the slopes of $V^i \in \mathcal{V} \backslash \mathcal{V}^b$ plus $\mathcal{L}^b$ is weakly greater than the negative sums of the slopes of $V^i \in \mathcal{V}$. By Prop. 1.4.2, we must have that $J^{\lambda_u}(\boldsymbol{s}, \lambda) \leq J(\boldsymbol{s}, \lambda)$, and respectively $\lambda_u \geq \lambda_{min}$.

$\lambda_{min} \geq \lambda_l$: The proof follows similarly. □

We now describe a quick method for computing $\mathcal{U}$ and $\mathcal{L}$. To construct these piecewise linear convex functions that will serve as the bounds, we make use of the insight that the slope of $V^i(s^i, \lambda)$ can be rapidly computed at any *test point* λ_{test} by calculating $(V^i(s^i, \lambda = \lambda_{test} + \varepsilon_0) - V^i(s^i, \lambda =$

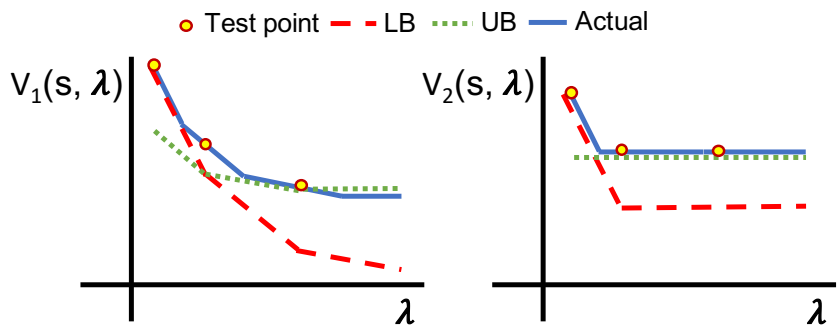


Figure 1.1: Constructing bounds on the slope of $V^i(s^i, \lambda)$ for two different arms with three test points. Note: bounds are with respect to the *slope*, not the value of the function.

$\lambda_{test}))/\varepsilon_0$ where both $V^i(s^i, \lambda)s$ can be quickly computed via value iteration and $\varepsilon_0 \approx 0$. Let G^i represent the set of test points for a given arm. The larger G^i, the tighter the bounds will be, but the higher the up-front computational cost. Thus the choice of both the size and the exact elements of G^i represent a set of parameters that can be tuned to maximize performance on a given distribution of V^is. However, at minimum, G^i must include $\lambda_{test} = 0$ since proposition 1.4.1 implies that the minimum slope of any V^i occurs at $\lambda = 0$. Once the slope at all the test points are computed, they are used to construct $\mathcal{U}$ and $\mathcal{L}$ via standard linear equations—the exact process is given in Appendix A.1. Let $\mathcal{U}^i(G^i_k, m)$ and $\mathcal{U}^i(G^i_k, b)$ be the slopes and intercepts, respectively, for each piece k of the upper bounding function $\mathcal{U}^i$ for arm i. Define $\mathcal{L}^i(G^i_k, *)$ similarly.

Now, we can compute λ_u and λ_ℓ. To start, we choose K arms to include in Eq. 1.7 in their $V^i(s^i, \lambda)$ form, while the other $N - K$ arms will be replaced by their bounded counterparts. To

compute λ_u, we replace the $N-K$ arms with $\mathcal{L}$ to get the following LP:

$$
\begin{aligned}
J^{\lambda_u}(\boldsymbol{s},\lambda) = \min_{V^i,\lambda,z^j} \; & \frac{\lambda B}{1-\beta} + \sum_{i=0}^{K} \mu^i(s^i) V^i(s^i,\lambda) + \sum_{j}^{N-K} z^j \\
\text{s.t. } & V^i(s^i,\lambda) \geq r^i(s^i) - \lambda c_j + \beta \sum_{s^{i\prime}} T(s^i, a^i_j, s^{i\prime}) V^i(s^{i\prime},\lambda) \\
& \qquad \forall i \in \{0,\ldots,K\}, \;\; \forall s^i \in \mathcal{S}, \;\; \forall a_j \in \mathcal{A} \\
& z^j \geq \mathcal{L}^j(G^j_k, m) * \lambda + \mathcal{L}^j(G^j_k, b) \\
& \qquad \forall k \in \{0,\ldots,|G^j|\}, \;\; \forall j \in \{0,\ldots,N-K\} \\
& \lambda \geq 0
\end{aligned}
\tag{1.8}
$$

where z^j are auxiliary variables to represent the piecewise linear convex functions $\mathcal{L}^j$ via the $|G^j|$ constraints on z^j. To compute λ_ℓ we construct a similar LP using $\mathcal{U}^i(G^i_k, *)$.

One important choice is in selecting the first K arms. Intuitively, the best $V^i(s^i,\lambda)$ to include in Eq. 1.8 are those with the loosest bounds. One proxy for looseness is the slope of the last segment, i.e., the steeper the slope, the looser the bound, since we know the slope of all $V^i(s^i,\lambda)$ go to 0 eventually (Prop. 1.4.1). Therefore, we first sort arms in ascending order by this criteria (line 3 in Alg. 1.4.2). To set K, we note that Prop. 1.4.2 implies that the negative sum of slopes of $V^i(s^i,\lambda)$ and $\mathcal{L}^i$ must be less than or equal to $B/(1-\beta)$ for some value of λ to find a solution. Since $\mathcal{L}^i$ are convex decreasing, if the negative sum of slopes of all the *trailing* segments of $\mathcal{L}^i$ are greater than $B/(1-\beta)$, then the LP will be unbounded. Thus, to guarantee the existence of a bounded solution, we set K to pick the first K arms in slope sorted order, such that the negative sum of slopes of all the trailing segments of $\mathcal{L}^i$ is less than $B/(1-\beta)$. We then set $K = \max(K, \sqrt{N})$ (line 4 Alg. 1.4.2).

Once λ_u and λ_ℓ are computed once, we iterate to include K_{step} more arms in the LP such that $K \mathrel{+}= K_{step}$ then repeat until the algorithm converges to within a difference ε. A straightforward in-

duction argument shows that as K grows (and the set of bounded arms shrinks), the bounds become progressively tighter and are guaranteed to be exact when $K = N$. Once λ_{min} is determined, we use value iteration to rapidly solve Eq. 1.7, the result of which we will use to derive feasible policies in Section 1.6.

1.4.2 BLam: Computational Complexity

In BLamPrecompute, BLam computes $\mathcal{U}^i(G^i_k, *)$ and $\mathcal{L}^i(G^i_k, *)$ for all $V^i(s^i, \lambda)$, which requires two runs of value iteration for each arm for each test point G^i_k. Assuming all arms use the same number of test points, states and actions, this scales as $\mathcal{O}(NG^i VI(|\mathcal{S}|, |\mathcal{A}|))$ where $VI()$ is the computational complexity of value iteration. While an exact complexity of value iteration is elusive, it is known to be much faster than the LP formulation [129]. Thus, its complexity will be dominated by the LP solves that occur in BLam— the same applies for the value iteration that runs at the end of BLam each round.

To compute a policy for each round, BLam constructs Eq. 1.8 as an LP which has $K|S| + (N-K)$ variables, $K|S||A|$ constraints with $|S|$ terms, and $(N-K)G^i$ constraints with two terms. Although the constraints associated with the $(N-K)$ auxiliary variables only have two non-zero coefficients, we conservatively assume that the matrix for this LP is dense in order to adopt the best known LP complexity result [69]. In the best case, BLam would provide tight bounds on λ_{min} after just one iteration. So setting $K = \sqrt{N}$ and assuming $G^i \ll N$, the per-round complexity is

$$\Omega(\sqrt{N}|S|^2|A| + N|S|^2 + N^{\frac{3}{2}}|S| + N^2) \tag{1.9}$$

Where the first term is the LP setup time to add constraints (which dominates the time to add variables) and the last three terms are the LP solve complexity, which is approximately square in the number of variables. Applying the same reasoning to the direct LP solve approach, which has $N|S|$

variables and $N|S||A|$ constraints gives the following best (and worst) case complexity

$$\mathcal{O}(N|S|^2|A| + N^2|S|^2) \tag{1.10}$$

Thus, BLam has a strictly better best-case complexity in the problem size. However, in the worst case, setting $K_{step} = \sqrt{N}$, BLam would require the full $\sqrt{N}$ iterations to get a tight bound on λ_{min}. In this case, the LP setup time would match the naive LP approach, but successive solves would become more expensive. Using basic summation, this gives a worst-case complexity of

$$\mathcal{O}(N|S|^2|A| + N^{\frac{5}{2}}|S|^2) \tag{1.11}$$

Which, handily, is only $\sqrt{N}$ worse than the naive approach. However, we will show in experiments that the typical run time and scaling of BLam is much faster than the naive approach in practice.

1.5 SampleLam

In some cases, especially very large problem sizes, speed can be more critical than performance. Thus, next, we give an algorithm for quickly computing a heuristic estimate of λ_{min} based on sampling. The approach is grounded in the mathematical interpretation of λ_{min}, i.e., that λ_{min} captures the willingness to violate the budget B given all V^i processes. In this algorithm, we will estimate our willingness to violate the budget for each arm *individually* given equal shares of the budget, solving a series of singleton LPs, then combine that knowledge to generate an estimate for λ_{min}

The algorithm is given in Alg. 1.5.1. It first chooses K processes at random to run through the subroutine QuickLP, which solves Eq. 1.7 with a single arm and modified budget $\frac{B}{N}$ giving a value λ^i that estimates the value of playing that arm. It then creates an estimate of λ_{min} by taking the sample mean of λ^i. Finally, it uses this estimate plus value iteration to solve for Eq. 1.7, which again we

Algorithm 1.5.1: SampleLam

Data: $T, R, C, N, B, \beta, r_{max}, c_{min}$

$N_{samples} = \frac{\log(N) r_{max}}{c_{min}}$;

inds = RandomChoice($[1, \ldots, N], N_{samples}$);

T, R = T[inds], R[inds];

$\lambda^i_{list} = []$;

for $i = 1 \ldots, N_{samples}$ **do**

  $\lambda^i = \text{QuickLP}(T[i], R[i], \frac{B}{N}, C, \beta)$;

  λ^i_{list}.append(λ^i)

$V(i, s) = []$ `// $Nx|\mathcal{S}|$ array to hold value functions`

$\lambda_{min} = \text{Mean}(\lambda^i_{list})$;

for $i = 1, \ldots, N$ **do**

  $R_\lambda = R[i]$;

  for $x \in 1, \ldots, |C|$ **do** subtract action costs

    $R_\lambda[x] -= \lambda_{min} * C[x]$;

  $V[i] = \text{VI}(T[i], R_\lambda, \beta)$;

return V

will use to derive feasible policies in section 1.6.

Although this method is not guaranteed to converge to the value of λ_{min}, it is very fast, and works well in practice on distributions which have an approximately normal distribution of budget across arms under the true λ_{min} policy. An example of such a distribution and the SampleLam estimate of λ_{min} is given in Fig. 1.2a.

1.5.1 SampleLam: Concentration Bound and Complexity

To understand SampleLam's convergence properties, i.e., convergence to the sample mean of λ^i, we derive a concentration bound below. The derivation first relies on showing that the distribution of λ^is is sub-Gaussian.

Theorem 1.5.1. *λ^i are $\frac{\sigma^2}{n}$-sub-Gaussian with $\sigma^2 = \frac{1}{4}\left(\frac{r_{max}}{c_{min}(1-\beta)}\right)^2$*

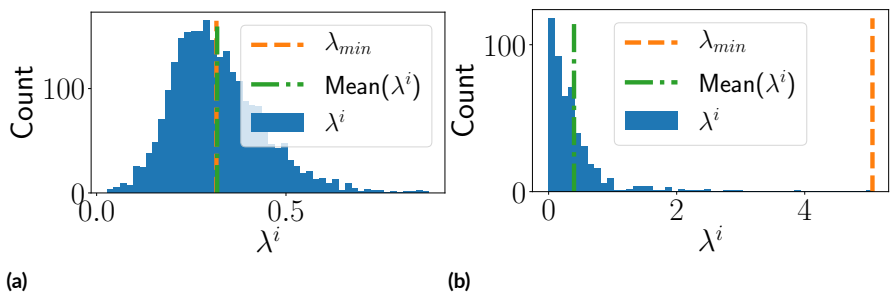


Figure 1.2: (a) λ^i is normally distributed about a mean equal to λ_{min} (b) λ_{min} is determined by a select few "important" arms, not equal to the sample mean. BLam is better suited for this case.

The proof involves showing $0 \leq \lambda^i \leq \frac{r_{max}}{c_{min}(1-\beta)}$ and is given in Appendix A.2. We can now use sub-Gaussianness to derive a concentration bound relating the number of samples to a confidence parameter $1-\delta$ on the estimate of the sample mean of λ^i.

Theorem 1.5.2. *The number of samples n needed to estimate the sample mean of λ^i within an error ε and with confidence $1-\delta$ is lower bounded as:*

$$n \geq \frac{1}{2\varepsilon^2}\left(\frac{r_{max}}{c_{min}(1-\beta)}\right)^2 \log\left(\frac{1}{\delta}\right) \tag{1.12}$$

The proof, given in Appendix A.2, uses the Hoeffding bound and Thm. 1.5.1. This bound gives the insight that the greater the reward to cost ratio, the more samples we need to well-estimate the mean. We account for this by including a factor of $\frac{r_{max}}{c_{min}}$ in the setting for K in the SampleLam algorithm. The drawback of this approach is that λ_{min} is not guaranteed to be close to the sample mean of λ^i in general. An adversarial case is shown in Fig. 1.2b in which SampleLam would compute an arbitrarily bad estimate for λ_{min}. Setting $K = \log(N)\frac{r_{max}}{c_{min}}$, the best and worst case complexity for SampleLam is:

$$\mathcal{O}\left(\log(N)\frac{r_{max}}{c_{min}}|S|^2|A| + N * VI(|\mathcal{S}|, |\mathcal{A}|)\right) \tag{1.13}$$

Where the first term is the cost of setting up $\log(N)\frac{r_{max}}{c_{min}}$ LPs, which dominates the solve time, and the second term is the cost of the final value iteration.

1.6 Computing a Policy

Finally, once λ_{min} is finalized, and the resulting value functions from Eq. 1.7 have been computed, we use the value functions to compute the one-step greedy policy implied by the bound. To do this, we expand the value functions to compute the *action-value function*, Q, which captures the long term value for acting in a given state in each arm. We then choose actions by solving a modified knapsack where $Q^i(s^i, a, \lambda_{min})$ are the values subject to their respective action costs, the budget B, and a constraint that ensures only one action is taken per arm. The knapsack LP is given in Appendix A.3, with an algorithm for computing $Q^i(s^i, a, \lambda_{min})$ from value functions.

1.7 Experiments

We test our algorithms on two synthetic settings. In each, we compare the discounted sum of rewards, using discount factor 0.95, averaged over all arms N, over $L = 40$ rounds. All results are averaged over 25 simulations. We compare our methods against the following baselines: **Nobody**: Take a_0 which has no cost on every arm; **VfNc**: Solves Eq. 1.7 with $\lambda = 0$, effectively ignoring all future constraints, then follows Section 1.6; **Hawkins**: Solves Eq. 1.7 directly using an LP solver, then follows Section 1.6. We also include several versions of BLam using various stopping criterion ε, noted in each plot as BLam{ε}. Larger ε will lead to faster running times but looser bounds on λ_{min}, and thus worse performance in general. All algorithms were implemented in Python 3.6 and use Gurobi version 9.0.3 to solve LPs via the gurobipy interface. All value iterations were computed using a lightly modified version of pymdptoolbox version 4.0b3 [35].

First, we explore an example distribution where VfNc and Haw-kins fail arbitrarily in terms

of performance and runtime respectively. In this distribution, there are three types of agents: **(1) Greedy:** Must take increasingly expensive actions to collect increasingly high reward. Once the required action is not taken, the agent never produces reward again. This is modeled with a single chain of states, each with unit-increasing reward, reachable only by an action with unit-increasing cost. Failure to take the next action leads to a dead state. **(2) Reliable:** Must take the cheapest non-zero action every round to achieve reward 1. If the arm is not played for any round, it never produces reward again. This is modeled with a simple 2-state chain, in which the final state recurs with the proper action, otherwise it goes to a dead state. **(3) Easy:** Always gives reward of 1 regardless of action. We make the proportion of (1) and (2) equal and set the budget so that all of (1) or (2) could be played (or some mix), but not more. Clearly, the optimal policy is to always play the Reliable agents since committing to the Greedy agents will eventually leave the planner only collecting reward from the Easy agents. However, the Greedy agents will look most attractive to VfNc since, without accounting for cost constraints, it will wrongly assume it can always pay the future cost to obtain increasingly larger reward. Hawkins and BLam, using their constraint-based reasoning, will instead commit to the Reliable agents. However, BLam will automatically detect the significant structure within the problem, such as the existence of Easy agents as well as a simple form for the V^i of Reliable agents, to build tight bounds on the Lagrange policy using only a small subset of agents in the coupled LP, leading to a significant speedup over Hawkins. The performance and runtimes of the algorithms tested on a population with 0.25, 0.25, 0.5 mix across Greedy, Reliable, and Easy agents with a budget of $0.25N$ and 30 actions (subsequently, 31 states) are shown in Fig. 1.3 and 1.4 confirming these insights. For BLam, all arms used test points $G^i = \{0\}$. Here, SampleLam gives good but variable performance in exchange for running approximately twice as fast as BLam.

Finally, we test our algorithms on a more rigorous simulation motivated by a real-world public health care challenge, namely, tuberculosis care in India. In this real-world setting, a single community health worker manages up to 200 patients throughout the course of their 6-month an-

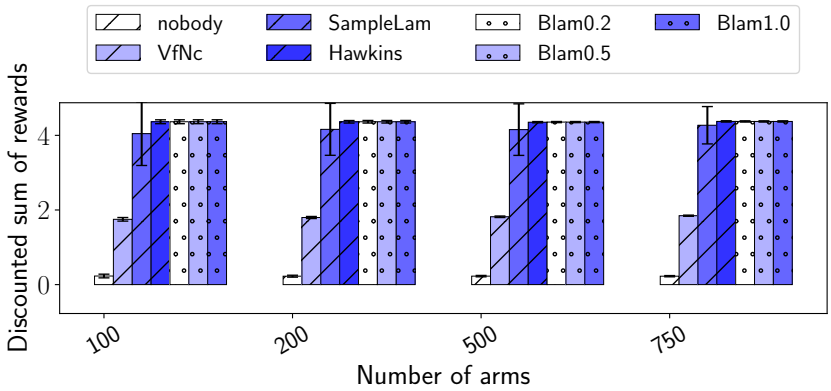


Figure 1.3: Ignoring future constraints leads to bad policies.

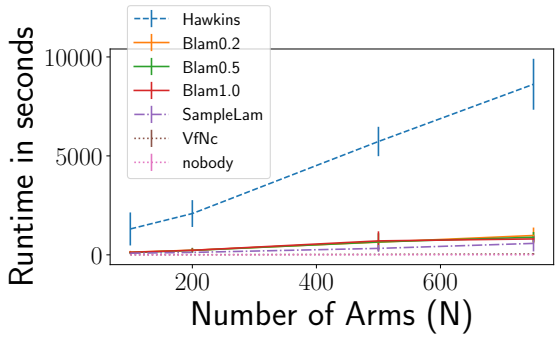


Figure 1.4: BLam and SampleLam scale better than Hawkins.

tibiotic regimen, monitoring and encouraging patients to take their daily medications. The health worker has a range of actions they can take on each patient aimed at improving their adherence, each with varying cost and effectiveness: call the patient (cheap), visit the patient in their home (semi-expensive), escalate the patient (very expensive). Because the worker's time and resources are limited, the number and types of actions they can take each day across all patients are also limited.

We model this problem as follows. In the simulation, each patient state is a tuple of (adherence level, treatment phase, day of treatment). The first entry captures the patient's previous d days of adherence. The second entry is binary and captures the "phase" of treatment: the intensive phase which lasts for the first IPL rounds, and the continuation phase which lasts from round IPL to the

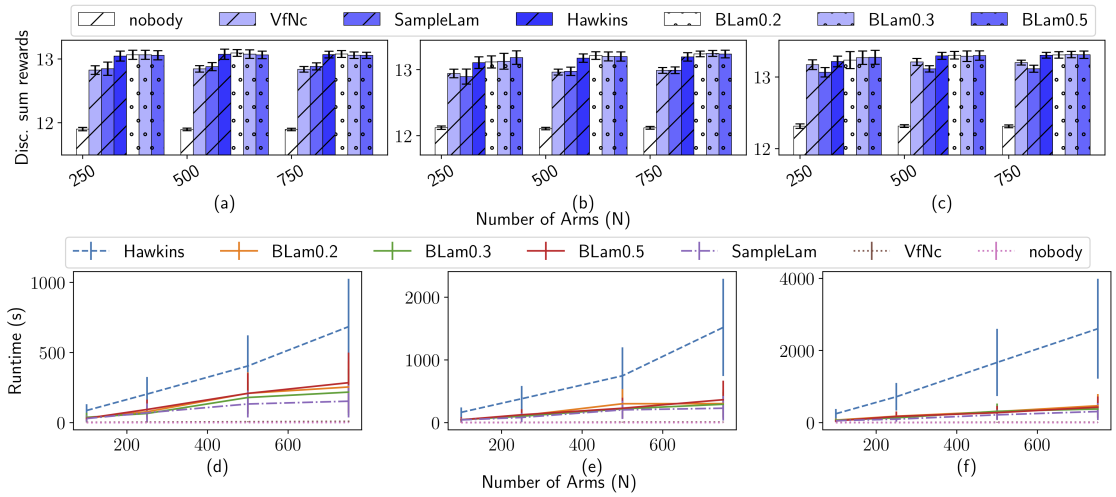


Figure 1.5: Rewards (top row) and runtimes (bottom row) on the health care dataset with budget $0.1N$. Columns represent $d = 3, 4, 5$ adherence levels, respectively. At all values of ε, BLam significantly outperforms VfNc. Further, the Hawkins LP scales quadratically in the number of states on each arm, while BLam identifies problem structure that keep the underlying LPs small, making speedups more dramatic as the problem size increases.

end. During the intensive phase, patients tend to have better adherence and are more responsive to intervention. During the continuation phase, both effects tend to degrade and patients may drop out (i.e., adherence of 0). The final entry captures time and can take any of $IPL + 2$ values. The first IPL values count days in the intensive phase and the next two are recurrent states that represent the continuation phase and the dropout state.

In one relevant dataset that captured daily treatment adherence of TB patients in India over the course of a year[82], patients followed four distinct modes: **(1) High adherence:** adhere daily regardless of health worker action. This makes up the majority of the data; **(2) Low adherence:** Very low adherence regardless of health worker action. **(3) Receptive patients:** Irregular adherence but can benefit from intervention. On average, their adherence drops during the continuation phase, suggesting that interventions become less effective. **(4) Dropout patients:** Like receptive patients but have probability of dropping out during the continuation phase.

We implement each of these patient types in our simulation and include them in the follow-

ing mix respectively: **0.64, 0.01, 0.175, 0.175**. This mix matches the number of High and Low adherence patients observed in the data, and splits the remaining portions evenly. At the start of simulation, each patient is in the maximum adherence state since in the real world, patients begin treatment in person. We run experiments with $d = 3, 4, 5$ adherence levels and set $IPL = 2d$. The health worker's action types are as follows: **(1) No action:** Take no action (c=0); **(2) Call:** Moderately increased probability that patient will increase adherence state by 1 (c=1). **(3) Visit:** Significant increased probability that patient will increase adherence state by 1 (c=2). **(4) Escalate:** Near-certain probability that patient will return to the maximum adherence state. If a patient is in the dropout state, there is a small probability they return to the continuation phase (c = B). Finally, rewards are defined as (adherence level)/d, so more rewards are received for patients at higher adherence levels.

We simulate this setting for many parameter combinations. For BLam we report results using test points $G^i = \{0, 0.1, 0.2, 0.5\}$, though we found that, in general, most sets of 3 or 4 evenly spaced points worked well. Fig. 1.5 shows the performance and runtime for the dataset with budget of $0.1N$ for $d = 3, 4$, and 5 adherence levels. With such a small budget, the tradeoff between individual actions is important. In Fig. 1.5 we see that all versions of BLam significantly outperform VfNc. Crucially, all versions of BLam also scale much better than Hawkins. In fact, as the number of states in the underlying problem grows the speed ups become even more dramatic ranging from a 2 times speedup with $d = 3$ to a 5 times speedup with $d = 5$. This is because the Hawkins LP scales quadratically in the number of states of each arm, while the BLam algorithms are able to identify problem structure that keep the underlying LPs small with its bounding techniques, making speedups more dramatic as the problem size increases.

In Appendix A.4, we run experiments varying the budget between $0.1N$, $0.2N$, $0.5N$, with $d = 4$ adherence levels. As the budget increases, resources are less constrained, so all methods tend to collapse to the same reward. However, again, BLam's adaptivity allows it to recognize when the problem is less constrained to automatically converge even more quickly to the optimal solution.

These results demonstrate the exemplary ability for our approach to scale well without sacrificing performance on a dataset whose technical structural conditions have no been established a priori. That is, our algorithm can perform exceptionally with minimal tuning, while avoiding undertaking the considerable effort of deriving an index policy and the existence thereof.

1.8 Conclusion

Our work makes available multiple avenues for computing well-performing policies on new MARMABs at scale. We demonstrate that our algorithms offer vast speedups and can be readily adapted to new problems without the need for the user to first arduously derive a problem-specific index policy, as was previously the case. These advances make multi-action RMABs newly accessible, laying the groundwork for wider study of this important framework.

2

Q-Learning Lagrange Policies for Multi-Action Restless Bandits

2.1 Introduction

Restless Multi-Armed Bandits (RMABs) are a versatile sequential decision making framework in which, given a budget constraint, a planner decides how to allocate resources among a set of in-

dependent processes that evolve over time. This model, diagrammed in Fig. 2.1, has wide-ranging applications, such as in healthcare [92,103,17], anti-poaching patrol planning [130], sensor monitoring tasks [67,54], machine replacement [137], and many more. However, a key limitation of these approaches is they only allow planners a binary choice—whether or not to allocate a resource to an arm at each timestep. However, in many real world applications, a planner may choose among multiple actions, each with varying cost and providing varying benefits. For example, in a community health setting (e.g., Figure 2.1), a health worker who monitors patients' adherence to medication may have the ability to provide interventions via text, call, or in-person visit. Such *multi-action* interventions require varying amount of effort (or cost), and cause varying effects on patients' adherence. Given a fixed budget, the problem for a health worker is to decide what interventions to provide to each patient and when, with the goal of maximizing the overall positive effect (e.g., the improvement of patients' adherence to medication).

Owing to the improved generality of *multi-action RMABs* over binary-action RMABs, this setting has gained attention in recent years [52,63,81]. However, critically, all these papers have assumed the *offline* setting, in which the dynamics of all the underlying processes are assumed to be known before planning. This assumption is restrictive since, in most cases, the planner will not have perfect information of the underlying processes, for example, how well a patient would respond to a given type of intervention.

To address this shortcoming in previous work, this chapter presents the first algorithms for the *online* setting for multi-action RMABs. Indeed, the online setting for even binary-action RMABs has received only limited attention, in the works of Fu et al. [46], Avrachenkov and Borkar [12], and Biswas et al. [18,19]. These papers adopt variants of the Q-learning update rule [159,158], a well studied reinforcement learning algorithm, for estimating the effect of each action across changing dynamics of the systems. These methods aim to learn Whittle indexes [163] over time and use them for choosing actions. In the offline version, it has been shown that these indexes lead to an optimal selection

policy when the RMAB instances meet the *indexability* condition. However, these methods only apply to binary-action RMABs. This chapter presents two new algorithms for online multi-action RMABs to address the shortcomings of previous work and presents an empirical comparison of the approaches. The chapter provides three key contributions:

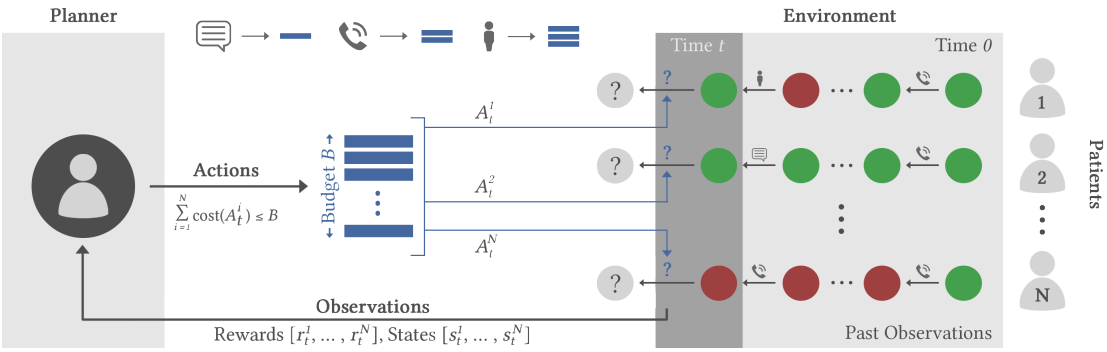


Figure 2.1: Schematic of a multi-action RMAB. At each timestep, t, the planner (e.g., health worker) takes one action on each of N processes (e.g., patients). The sum cost of actions each timestep must not exceed a budget, B. After taking actions at each timestep, the planner observes the rewards and state transitions of the processes, which the planner uses to improve their action selection in the future. The goal is to maximize reward.

1. **We design Multi-action Index-based Q-learning (MAIQL).** We consider a multi-action notion of indexability where the index for each action represents the "fair charge" for taking that action [52]. If the dynamics of the underlying systems were known beforehand, an optimal policy for multi-action indexable RMABs would choose actions based on these indexes when a linear structure on the action costs is assumed [63]. We establish that, when these dynamics are unknown and are required to be learned over time, MAIQL provably converges to these indexes for any multi-action RMAB instance following the assumptions on cost and indexability. However, these assumptions can be limiting, and in addition, the algorithm requires a two-timescale learning procedure that can be slow and unstable.

2. **We propose a more general algorithm, Lagrange Policy Q-learning (LPQL).** This method takes a holistic back-to-the-basics approach of analyzing the Lagrangian relaxation of the multi-

action RMAB problem and learning to play the *Lagrange policy* using the estimated Q values which are updated over time. This policy converges more quickly than MAIQL and other benchmark algorithms, is applicable to problems with arbitrary cost structures, and does not require the indexability condition.

3. **We demonstrate the effectiveness of MAIQL and LPQL as compared to various baselines on several experimental domains,** including two synthetically generated domains and derived from a real-world dataset on medication adherence of tuberculosis patients. Our algorithms converge to the state-of-the-art offline policy much faster than the baselines, taking a crucial step toward real-world deployment in online settings.*

2.2 Related Work

The *restless multi-armed bandit* (RMAB) problem was introduced by Whittle [163] where he showed that a relaxed version of the offline RMAB problem can be solved optimally using a heuristic called the *Whittle index policy*. This policy is shown to be asymptotically optimal when the RMAB instances satisfy the *indexability* condition [161]. Moreover, Papadimitriou and Tsitsiklis [122] established that solving RMABs is PSPACE-hard, even for the special case when the transition rules are deterministic.

Since then, a vast literature have studied various subclasses of RMABs and provided algorithms for computing the Whittle index. Lee et al. [92] study the problem of selecting patients for screening with the goal of maximizing early-stage cancer detection under limited resources. Mate et al. [103] consider bandits with two states to model a health intervention problem, where the uncertainty collapses after an active action. They showed that the model is indexable and gave a mechanism for computing the Whittle index policy. Bhattacharya [17] models the problem of maximizing the

*Code available at: https://github.com/killian-34/MAIQL_and_LPQL

coverage and spread of health information with limited resources as an RMAB and proposes a hierarchical policy. Similarly, several other papers[54,144,96] give Whittle indexability results for different subclasses of RMABs where there are only two possible actions.

For more than two actions, Glazebrook et al.[52,63] extended Whittle indexability to multi-action RMABs where the instances are assumed to have special monotonic structure. Along similar lines, Killian et al.[81] proposed a method that leverages the convexity of an approximate Lagrangian version of the multi-action RMAB problem.

Also related to multi-action RMABs are weakly coupled Markov decision processes (WCMDP). The goal of a WCMDP is to maximize reward subject to a set of constraints over actions, managing a finite number of independent Markov decision processes (MDPs). Hawkins[60] studied a Lagrangian relaxation of WCMDPs and proposed an LP for minimizing the Lagrange bound. On the other hand, Adelman and Mersereau[4] provide an approximation algorithm that achieves a tighter bound than the Lagrange approach to WCMDPs, trading off scalability. A more scalable approximation method is provided by Gocgun and Ghate[55].

However, these papers focused only on the offline versions of the problem in which the dynamics (transition and observation models) are known apriori. In the online setting, there has been some recent work on binary-action RMABs. Gafni and Cohen[47] propose an algorithm that learns to play the arm with the highest *expected* reward. However, this is suboptimal for general RMABs since rewards are state- and action-dependent. Addressing this, Biswas et al.[18] give a Q-learning-based based algorithm that acts on the arms that have the largest difference between their active and passive Q values. Fu et al.[46] take a related approach that adjust the Q values by some λ, and use it to estimate the Whittle index. Similarly, Avrachenkov and Borkar[12] provide a two-timescale algorithm that learns the Q values as well as the index values over time. However, their convergence proof requires indexability and that all arms are homogeneous with the same underlying MDPs. We use the two-timescale methodology and define a multi-action indexability criterion to provide a general

framework to learn multi-action RMABs with provable convergence guarantees. Our work is the first to address the multi-action RMAB setting online.

2.3 Preliminaries and Notations

A Multi-action RMAB instance consists of N arms and a budget B on the total cost. Each arm $i \in [N]$ follows an MDP [129]. We define an MDP $\{\mathcal{S}, \mathcal{A}, \mathcal{C}, r, T, \beta\}$ as a finite set of states $\mathcal{S}$, a finite set of M actions $\mathcal{A}$, a finite set of action costs $\mathcal{C} := \{c_j\}_{j \in \mathcal{A}}$, a reward function $r : \mathcal{S} \to \mathbb{R}$, a transition function $T(s, a, s')$ denoting the probability of transitioning from state s to state s' when action a is taken, and a discount factor $\beta \in [0, 1)^{\dagger}$. An MDP *policy* $\pi : \mathcal{S} \to \mathcal{A}$ maps states to actions. The long-term *discounted reward* of arm i starting from state s is defined as

$$J^i_{\beta,\pi^i}(s) = E\left[\sum_{t=0}^{\infty} \beta^t r^i(s^i_t) | \pi^i, s^i_0 = s\right] \tag{2.1}$$

where $s^i_{t+1} \sim T(s^i_t, \pi^i(s^i_t), \cdot)$. For ease of exposition, we assume the action sets and costs are the same for all arms, but our methods will apply to the general case where each arm has arbitrary (but finite) state, action, and cost sets. Without loss of generality, we also assume that the actions are numbered in increasing order of their costs, i.e., $0 = c_0 \leq c_1 \leq \ldots, c_M$. Now, the planner must take decisions for all arms jointly, subject to two constraints each round: (1) select one action for each arm and (2) the sum of action costs over all arms must not exceed a given budget B. Formally, the planner must choose a decision matrix $\boldsymbol{A} \in \{0, 1\}^{N \times M}$ such that:

$$\sum_{j=1}^{M} \boldsymbol{A}_{ij} = 1 \quad \forall i \in [N] \qquad\qquad \sum_{i=1}^{N} \sum_{j=1}^{M} \boldsymbol{A}_{ij} c_j \leq B \tag{2.2}$$

$^{\dagger}\beta$ is only included under the discounted reward case, as opposed to the average reward case which we address later.

Let $\overline{\mathcal{A}}$ be the set of decision matrices respecting the constraints in 2.2 and let $\boldsymbol{s} = (s^1, \ldots, s^N)$ represent the initial state of each arm. The planner's goal is to maximize the total discounted reward of all arms over time, subject to the constraints in 2.2, as given by the constrained Bellman equation:

$$J(\boldsymbol{s}) = \max_{\boldsymbol{A} \in \overline{\mathcal{A}}} \left\{ \sum_{i=1}^{N} r^i(s^i) + \beta E[J(\boldsymbol{s}')|\boldsymbol{s}, \boldsymbol{A}] \right\} \tag{2.3}$$

However, this corresponds to an optimization problem with exponentially many states and combinatorially many actions, making it PSPACE-Hard to solve directly [122]. To circumvent this, we take the Lagrangian relaxation of the second constraint in 2.2 [60]:

$$\begin{aligned} &J(\boldsymbol{s}, \lambda) = \\ &\max_{\boldsymbol{A}} \left\{ \sum_{i=1}^{N} r^i(s^i) + \lambda(B - \sum_{i=1}^{N} \sum_{j=1}^{M} \boldsymbol{A}_{ij} c_j) + \beta E[J(\boldsymbol{s}', \lambda)|\boldsymbol{s}, \boldsymbol{A}] \right\} \end{aligned} \tag{2.4}$$

Since this constraint was the only term coupling the MDPs, relaxing this constraint decomposes the problem except for the shared term λ. So Eq. 2.4 can be rewritten as (see [4]):

$$J(\boldsymbol{s}, \lambda) = \frac{\lambda B}{1 - \beta} + \sum_{i=1}^{N} \max_{a_j^i \in \mathcal{A}^i} \{(Q^i(s^i, a_j^i, \lambda)\} \tag{2.5}$$

$$\begin{aligned} &\text{where } Q^i(s^i, a_j^i, \lambda) = \\ &r^i(s^i) - \lambda c_j + \beta \sum_{s'} T(s^i, a_j^i, s') \max_{a_j \in \mathcal{A}^i} \{(Q^i(s', a_j, \lambda)\} \end{aligned} \tag{2.6}$$

In Eq. 2.6, each arm is effectively decoupled, allowing us to solve for each arm independently for a given value of λ. The choice of λ, however, affects the resulting optimal policies in each of the arms. One intuitive interpretation of λ is that of a "penalty" associated with acting – given a fixed budget B, a planner must weigh the cost of acting λc_j against its ability to collect higher rewards. Thus, as

λ is increased, the optimal policies on each arm will tend to prefer actions that generate the largest "value for cost".

The challenges we address are two-fold: (1) How to learn policies online that can be tuned by varying λ and (2) How to make choices for the setting of λ that lead to good policies. Our two algorithms in Sections 2.4 and 2.5 both build on Q-Learning to provide alternative ways of tackling these challenges – where MAIQL builds on the rich existing literature of "index" policies, LPQL goes "back to basics" and provides a more fundamental approach based on the Lagrangian relaxation discussed above.

2.4 Algorithm: MAIQL

Our first algorithm will reason about λ's influence on each arm's value function independently. Intuitively, this is desirable because it simplifies one size-N problem to N size-one problems that can be solved quickly. Our goal will be to compute *indexes* for each action on each arm that capture a given action's value, then greedily follow the indexes as our policy. Such an *index policy* was proposed by Whittle [163] for binary-action RMABs, in which the index is a value of λ such that the optimal policy is indifferent between acting and not acting in the given state. This policy has been shown to be asymptotically optimal under the indexability condition [161].

Glazebrook et al. [52] and Hodge & Glazebrook [63] extended the definition and guarantees, respectively, of the Whittle index to multi-action RMABs that satisfy the following assumptions:

1. Actions have equally spaced costs, i.e., after normalization, the action costs can be expressed as $\{0, 1, \ldots, M-1\}$.

2. The utility of acting is submodular in the cost of the action.

For such multi-action RMABs, Glazebrook et al. [52] defines multi-action indexability and the multi-action index as follows:

Definition 2.4.1 (Multi-action indexability). *An arm is multi-action indexable if, for every given action $a_j \in \mathcal{A}$, the set of states in which it is optimal to take an action of cost c_j or above decreases monotonically from $\mathcal{S} \to \varnothing$ as λ increases from $-\infty \to \infty$.*

Definition 2.4.2 (Multi-action index, λ^*_{s,a_j}). *For a given state s and action a_j, the multi-action index is the minimum λ^*_{s,a_j} that is required to become indifferent between the actions a_j and a_{j-1}:*

$$\lambda^*_{s,a_j} = \inf_{\lambda}\{Q(s, a_j, \lambda) \leq Q(s, a_{j-1}, \lambda)\} \tag{2.7}$$

$$= \lambda, \ s.t. \ Q(s, a_j, \lambda) = Q(s, a_{j-1}, \lambda) \tag{2.8}$$

where $Q(s, a_j, \lambda)$ is the Q-value of taking action a_j in state s with current and future rewards adjusted by λ.

Given these multi-action indices, Hodge & Glazebrook [63] suggest a way to greedily allocate units of resources that is asymptotically optimal – assume that the arms are in some state $\boldsymbol{s}$, then iterate from $1 \ldots B$ and in each round allocate a unit of resource to the arm with the highest multi-action index associated with the next unit of resource. Specifically, if $\theta = \langle \theta^1 \ldots \theta^N \rangle$ units of resource have been allocated to each arm so far, then we allocate the next unit of resource to the arm with the highest $\lambda^*_{s^i, a_{\theta^i}}$. Given that the action utilities (λ^*_{s,a_j}) are submodular in a_j by assumption, this *multi-action index policy* leads to the allocation in which the sum of multi-action index values across all the arms are maximized.

Given the policy's theoretical guarantees, an index-based solution to the multi-action RMAB problem is attractive. The question then is how to calculate the value of the multi-action indices λ^*_{s,a_j}. In the online setting (when the RMAB dynamics are unknown apriori), Avrachenkov & Borkar [12] proposes a method for estimating the Whittle indexes for binary-action RMABs and, in addition, proves that this algorithm's estimate converges to the Whittle index.

In this section, we describe the **M**ulti-**A**ction **I**ndex **Q**-**L**earning (MAIQL) algorithm. Our algorithm generalizes the update rule of the learning algorithm proposed by Avrachenkov & Borkar [12]. We consider the notion of multi-action indexability from Glazebrook et al. [52] to create an update rule that allows us to estimate the multi-action indexes (Section 2.4.1). In addition, we use the multi-action indexability property to show that the convergence guarantees from Avrachenkov & Borkar [12] are preserved in this multi-action extension (Section 2.4.2).

2.4.1 Algorithm

From Equation 2.8, we observe that if we could estimate the Q values for all the possible values of λ, we would know the value of λ^*_{s,a_j}. This is not possible in general, but we can convert this insight into an update rule for estimating λ^*_{s,a_j} in which we update the current estimate in the direction such that Eq. 2.8 is closer to being satisfied. Based on this, we propose an iterative scheme in which Q values and λ^*_{s,a_j} are learned together.

An important consideration is that, because the Q and λ^*_{s,a_j} values are inter-dependent, it is not straightforward to learn them together, since updating the estimate of one may adversely impact our estimate of the other. To combat this, we decouple the effects of learning each component by relegating them to separate time-scales. Concretely, this means that an adaptive learning rate $\alpha(t)$ for the Q values and $\gamma(t)$ for λ-values are chosen such that $\lim_{t\to\infty} \frac{\gamma(t)}{\alpha(t)} \rightarrow 0$, i.e., the Q values are learned on a fast-time scale in which λ values can be seen as quasi-static (details in appendix B.1 and B.3). The resultant two time-scale approach is given below.

To calculate the multi-action index, for a given state $s \in \mathcal{S}$ and action $a_j \in \mathcal{A}$, we store two sets of values: (1) the Q values for all states and actions, $Q(s, a_j)\ \forall s \in S,\ a_j \in \mathcal{A}$, and (2) the current estimate of the multi-action index λ_{s,a_j}. All the Q and λ values are initiated to zero. Then, for a given state s in which we take action a_j, we observe the resultant reward r and next state s', then perform the following updates:

1. **Q-update:** At a fast time-scale (adjusted by $\alpha(\nu(s, a_j, t))$), update to learn the correct Q values as in standard Q-learning:

$$Q_\lambda^{t+1}(s, a_j) = Q_\lambda^t(s, a_j) + \alpha(\nu(s, a_j, t))\big[[r(s) - \lambda_{s,a_j}^t c_j \\ -f(Q_\lambda^t) + \max_{a_j'} Q_\lambda^t(s', a_j')] - Q_\lambda^t(s, a_j)\big] \tag{2.9}$$

   where $\nu(s, a_j, t)$ is a "local-clock" that stores how many times the specific $Q_\lambda(s, a_j)$ value has been updated in the past, and $f(Q_\lambda^t) = \frac{\sum_{s,a_j} Q_\lambda^t(s,a_j)}{\sum_{s,a_j} 1}$ is a function whose value converges to the optimal average reward[2]. We give the average reward case to align with the traditional derivation of binary-action Whittle indexes, but this update (and related theory) can be extended easily to the discounted reward case.

2. λ**-update:** Then, at a slower time-scale (adjusted by a function $\gamma(t)$), we update the value of λ_{s,a_j}^t according to:

$$\lambda_{s,a_j}^{t+1} = \lambda_{s,a_j}^t + \gamma(t) \cdot (Q_\lambda^t(s, a_j) - Q_\lambda^t(s, a_{j-1})) \tag{2.10}$$

Note that the updates described in the paragraph above correspond to the estimation of a single multi-action index. To efficiently estimate $\lambda^*(s, a_j) \ \ \forall s, a$, we make use of the fact that our algorithm, like the Q-learning algorithm on which it is based, is off-policy – an off-policy algorithm does not require collecting samples using the policy that is being learned. As a result, rather than learn each of these multi-action index values sequentially, we learn them in parallel based on the samples drawn from a single policy.

Specifically, since learning each index value requires imposing the current estimate λ on all current and future action costs, and since a separate index is learned for all arms, states, and non-passive actions, $N(M-1)|\mathcal{S}|$ separate Q-functions (each a table of size $|\mathcal{S}| \times M$) and λ-values must be maintained, requiring $\mathcal{O}(NM^2|\mathcal{S}|^2)$ memory. However, since the estimation of each index is in-

dependent, each round, the index and its Q-function can be updated in parallel, keeping the process efficient, but requiring $\mathcal{O}(NM|\mathcal{S}|)$ time if computed in serial. To take actions, we follow an ε-greedy version of the multi-action index policy – which, when not acting randomly, greedily selects indices in increasing size order for each arm's current state, taking $\mathcal{O}(NM)$ time – and store the resultant $\langle s, a, r, s' \rangle$ tuple in a replay buffer. The replay buffer is important because, in the multi-action setting, each (s, a) pair is not sampled equally often; specifically, especially when B is small, it is less likely to explore more expensive actions. After every fixed number of time-steps of running the policy, we randomly pick some $\langle s, a, r, s' \rangle$ tuples from the replay buffer with probability weighted inversely to the number of times the tuple has been used for training, and update the Q values associated with each of the multi-action indexes and the $\lambda_{s,a}$ estimate for the sampled (s, a).[‡] The resulting algorithm is guaranteed to converge to the multi-action indexes. Pseudocode is given in appendix B.3.

2.4.2 Theoretical Guarantees

The attractiveness of the MAIQL approach comes from the fact that, if the problem is multi-action indexable, the indexes can always be found. Formally, we show:

Theorem 2.4.3. *MAIQL converges to the optimal multi-action index $\lambda^*_{s,a}$ for a given state s and action a under Assumptions 1, 2, 3, and the problem being multi-action indexable.*

Proof Sketch. **At the fast time-scale:** We can assume $\lambda_{s,a}$ to be static. Then, for a given value of $\lambda_{s,a} = \lambda'$, the problem reduces to a standard MDP problem, and the Q-learning algorithm converges to the optimal policy.

At the slow time-scale: We can consider the fast-time scale process to have converged, and we have the optimal Q values $Q^*_{\lambda'}$ corresponding to the current estimate of $\lambda_{s,a}$. Then, by the multi-

[‡]all algorithms in this chapter will be equipped with the replay buffer for fairness of comparison.

action indexability property, we know that if $\lambda < \lambda^*$ an action of weight a or higher is preferred. As a result, $Q^*_{\lambda_{s,a}}(s,a) - Q^*_{\lambda_{s,a}}(s,a-1) > 0$, and so $\lambda^{t+1} > \lambda^t$. When $\lambda > \lambda^*$, the opposite is true and so $\lambda^{t+1} < \lambda^t$. As a result, we constantly improve our estimate of λ such that we eventually converge to the optimal multi-action index, i.e., $\lim_{t\to\infty} \lambda^t \to \lambda^*_{s,a}$. □

The detailed proof follows along the lines of Avrachenkov & Borkar [12], and can be found in appendix B.1. However, while they consider convergence in the binary-action case, our approach generalizes to the multi-action setting. The crux of the proof lies in showing how the multi-action index generalizes the properties of the Whittle index in the multi-action case, and leads to convergence in the slow time-scale.

2.4.3 MAIQL Limitations

The main limitations of MAIQL are (1) it assumes multi-action indexability and equally-spaced action costs to be optimal and (2) it learns on two time-scales, making convergence slow and unstable in practice. i.e., for the convergence guarantees to hold, MAIQL must see "approximately" infinitely many of all state-action pairs before updating λ once. This can be difficult to ensure in practice for arms with transition probabilities near 0 or 1, and for problems where the budget is small, since many more samples of (s,a) pairs with cheap actions will be collected than ones with expensive actions.

2.5 Algorithm: LPQL

In this section, we provide a more fundamental approach by studying the problem of minimizing $J(\cdot,\lambda)$ (Equation 2.5) over λ. By minimizing this value, we aim to compute a tight bound on Eq. 2.3, the value function of the original, non-relaxed problem, then follow the policies implied by the bound, i.e., the Lagrange policy. However, computing $J(\cdot,\cdot)$ requires the $Q^i(s,a,\lambda)$ val-

ues which in turn require the knowledge of transition probabilities (as shown in Equation 2.6). In absence of the knowledge of transition probabilities, we propose a method, called **L**agrange **P**olicy **Q**-**L**earning (LPQL). This method learns a representation of $Q^i(s, a_j, \lambda)$ by using samples obtained from the environment via a mechanism similar to MAIQL. However, rather than estimating $Q^i(s, a_j, \lambda)$ with the purpose of estimating some downstream value of λ (i.e., indexes), now the goal is to estimate the entire curve $Q^i(s, a_j, \lambda)$ with respect to λ. It is straightforward to show that $Q^i(s, a_j, \lambda)$ is convex decreasing in λ [60], meaning that once we have a representation of $Q^i(s, a_j, \lambda)$, minimizing $J(\cdot, \cdot)$ simply corresponds to a one-dimensional convex optimization problem that can be solved extremely quickly.

In addition to its speed, this approach is desirable because it is designed for RMAB instances without specific structures, i.e., LPQL accommodates arbitrary action costs and needs no assumption on indexability. It does so by computing the Lagrange policy, which is asymptotically optimal for binary-action RMABs regardless of indexability [161], and works extremely well in practice for multi-action settings [81]. LPQL enjoys these benefits, and further, is designed to work on a single learning timescale, making its convergence faster and more stable than MAIQL.

In the offline setting, $J(\cdot, \cdot)$ can be minimized by solving this linear program (LP), which can be derived directly from Eq. 2.5 [60]:

$$\begin{aligned}
&\min_{\lambda} J(\boldsymbol{s}, \lambda) = \min_{V^i(s^i,\lambda),\lambda} \frac{\lambda B}{1-\beta} + \sum_{i=0}^{N-1} \mu^i(s^i) V^i(s^i, \lambda) \\
&\text{s.t. } V^i(s^i, \lambda) \geq r^i(s^i) - \lambda c_j + \beta \sum_{s^{i\prime}} T(s^i, a_j^i, s^{i\prime}) V^i(s^{i\prime}, \lambda) \\
&\forall i \in \{0, \ldots, N-1\}, \;\; \forall s^i \in \mathcal{S}, \;\; \forall a_j \in \mathcal{A}, \text{ and } \lambda \geq 0
\end{aligned} \tag{2.11}$$

where $\mu^i(s^i) = 1$ if s^i is the start state for arm i, else it is 0, and $V^i(s^i, \lambda) = \max_{a_j}\{Q^i(s^i, a_j, \lambda)\}$. To learn $Q^i(s^i, a_j, \lambda)$ in the offline setting, we will build a piecewise- linear convex representation

of the curve by estimating its value at various points λ_p. To do this, we keep a three-dimensional vector for each arm $Q(s, a_j, \lambda) \in \mathbb{R}^{|\mathcal{S}| \times M \times n_{lam}}$ where n_{lam} is the number of points λ_p at which to estimate the curve. For now, we choose the set of λ_p to be an equally spaced grid between 0 and some value $\lambda_{\max}$. Since $V^i(s, \lambda)$ is convex decreasing in λ, the largest possible value of λ that could be a minimizer of $J(\cdot, \cdot)$ is the λ where $\frac{dQ^i(s,a_j,\lambda)}{d\lambda} = 0$. Killian et al.[81] show that this value is no greater than $\frac{\max\{r\}}{\min\{C\}(1-\beta)}$, so this will serve as $\lambda_{\max}$ unless otherwise specified.

On each round, an (s, a_j, r, s') tuple is sampled for each arm. We store estimates of Q for each state, action, and λ_p value, requiring $\mathcal{O}(Nn_{lam}|\mathcal{S}|\mathcal{M})$ memory. The update rule for $Q(s, a_j, \lambda_p)$ is:

$$Q^{t+1}(s, a_j, \lambda_p) = Q^t(s, a_j, \lambda_p) + \alpha(\nu(s, a_j, n)) * \\ \left[[r(s) - \lambda_p c_j + \beta \max_{a'_j \in \mathcal{A}} Q^t(s', a'_j, \lambda_p)] - Q^t(s, a_j, \lambda_p)\right] \tag{2.12}$$

Where β is the discount factor. Each round, we sample a (s, a_j, r, s') tuple per arm, and for each arm loop to update $Q^{t+1}(s, a_j, \lambda_p)\ \forall p$. As in MAIQL, this update can be parallelized but requires $\mathcal{O}(Nn_{lam})$ time if computed serially. To choose a policy each round, we compute the minimum of Eq. 2.5 by finding the point at which increasing λ_p (stepping from 0, $\frac{\lambda_{\max}}{n_{lam}}, \ldots, \lambda_{\max}$) results in zero or positive change in objective value, as computed via our estimates $Q(s, a_j, \lambda_p)$, taking $\mathcal{O}(n_{lam})$ time. As our estimates $Q(s, a, \lambda_p)$ converge, we approximate points exactly on the true $Q(s, a_j, \lambda)$ curve. Even at convergence, there will be some small approximation error in the slope of the line that will manifest as error in the objective value, but in the next subsection, we show that the approximation error can be made arbitrarily small as n_{lam} increases.

Once the minimizing value of λ (λ_{min}) is found, we follow the knapsack from[81] to select actions, i.e., we input $Q(s, a_j, \lambda_{min})$ as values in a knapsack where the costs are the corresponding c_j and the budget is B. We then use the Gurobi optimizer software[58] to solve the knapsack, then carry out the policy in accordance with the selected actions, taking $\mathcal{O}(NMB)$ time in total[81]. Pseudocode for

LPQL is given in appendix B.3.

2.5.1 Theoretical Guarantees

We establish that, given a $\lambda_{\max}$, a higher n_{lam} results in a better approximation of the upper bound of the policy return, given in Eq. 2.5. We show that, given a state profile $\boldsymbol{s} = \{s^1, \ldots, s^N\}$, the asymptotic values of $V^i(s^i, \lambda)$ obtained at equally spaced discrete set of λ values (over-)approximates Equation 2.11. The smaller the intervals are, the closer is the approximated value of $J(\boldsymbol{s}, \lambda)$ at all λ points that are not at the interval points. Before stating the theorem formally, we define the *Chordal Slope* Lemma. For ease of representation, we drop the notations $\boldsymbol{s}$ and s^i from functions $V()$ and $J()$ and also remove the superscript i.

Lemma 2.5.0.1 (The Chordal Slope Lemma[48]). *Let F be a convex function on (a, b). If $x_1 < x < x_2$ are in (a, b), then for points $P_1 = (x_1, F(x_1))$, $P = (x, F(x))$, and $P2 = (x_2, F(x_2))$, the slope of the straight line P_1P is less than or equal to the slope of the straight line $P1P2$.*

Theorem 2.5.1. *Let $V'(\cdot)$ be a convex piecewise-linear function over equally spaced intervals ($\Lambda := \{0, x, 2x, 3x, \ldots\}$) that approximates the convex decreasing function $V(\lambda)$, such that*

$$V(\lambda) = V'(\lambda) \text{ for all } \lambda \in \Lambda.$$

If values $V(\cdot)$ are replaced by values $V'(\cdot)$, then $J(\cdot)$ (Equation 2.11) is better approximated when the interval length x is small.

Proof. $V(\cdot)$ are convex functions of λ which implies that the function $J(\lambda)$, the sum of convex functions, is also a convex function of λ. Let us assume that the convex decreasing function $V(\lambda)$ is approximated by a convex continuous piecewise-linear function $V'(\lambda)$, over equally spaced values, taken from the set $\Lambda := \{0, x, 2x, 3x, \ldots\}$, such that $V(\lambda) = V'(\lambda)$ for all $\lambda \in \Lambda$. Thus, using

$V'(\cdot)$ values instead of $V(\cdot)$ values, we obtain an approximation $J'(\cdot)$ of the convex function $J(\lambda)$. The function $J'(\lambda)$ is a convex function with $J(\lambda) = J'(\lambda)$ for all $\lambda \in \Lambda$.

Now, let us assume two different values of x, say x_1 and x_2, where $x_1 < x_2$. The corresponding sets are $\Lambda_1 := \{0, x_1, 2x_1, \ldots\}$ and $\Lambda_2 := \{0, x_2, 2x_2, \ldots\}$. Considering Λ_1, and two points $\lambda_0 \geq 0$ and $\lambda_1 = \lambda_0 + x_1$, $J'(\cdot)$ is over approximated by a straight line $\bar{J}(\cdot)$ that connects $(\lambda_0, J(\lambda_0))$ and $(\lambda_1, J(\lambda_1))$. The equation for the line is given by:

$$\bar{J}_{\lambda_0, x_1, \lambda_1}(\lambda) = J(\lambda_0) + \frac{\lambda - \lambda_0}{x_1}(J(\lambda_1) - J(\lambda_0)) \; \forall \, \lambda_0 \leq \lambda \leq \lambda_1. \tag{2.13}$$

Similarly, considering Λ_2, the point λ_0, and point $\lambda_2 = \lambda_0 + x_2$ (where $x_1 < x_2$), $J'(\cdot)$ can be over approximated by a straight line $\bar{J}(\cdot)$ that connects $(\lambda_0, J(\lambda_0))$ and $(\lambda_2, J(\lambda_2))$. Thus, for any value of $\lambda \in [\lambda_0, \lambda_2]$, the difference $J'(\lambda) - J(\lambda)$ is given by:

$$\bar{J}_{\lambda_0, x_2, \lambda_2}(\lambda) = J(\lambda_0) + \frac{\lambda - \lambda_0}{x_2}(J(\lambda_2) - J(\lambda_0)) \; \forall \, \lambda_0 \leq \lambda \leq \lambda_2. \tag{2.14}$$

For a given $\lambda \in [\lambda_0, \lambda_1]$, the difference between the approximation obtained by Equation 2.14 and 2.13 is:

$$\begin{aligned} &(\lambda - \lambda_0)\left(\frac{J(\lambda_2) - J(\lambda_0)}{x_2} - \frac{J(\lambda_1) - J(\lambda_0)}{x_1}\right) \\ &\geq 0 \qquad\qquad (\because \lambda \geq \lambda_0 \text{ and } \textit{Lemma } 2.5.0.1) \end{aligned} \tag{2.15}$$

Thus, smaller the length of each interval, the corresponding surrogate $V'(\cdot)$ values can be used to obtain a better approximation of $J(\cdot)$ values. □

2.5.2 Extending LPQL Update Technique to Approximate MAIQL

The same tactic of approximating $Q(s, a_j, \lambda_p)$ can be used to create an **approximate version of MAIQL (MAIQL-Aprx)** that learns on a single timescale and is thus more sample efficient and stable. The algorithm follows much in the same way as LPQL, except that $Q(s, a, \lambda_p)$ are not used to minimize the LP. Instead, for each arm on each round, we compute the multi-action index for a given (s, a_j) by finding the $\text{argmin}_{\lambda_p} |Q(s, a_j, \lambda_p) - Q(s, a_{j-1}, \lambda_p)|$. We then choose actions according to the same greedy policy as MAIQL. We can show with the same logic as the LPQL approximation proof that with a large enough n_{lam}, the indexes can be approximated to an arbitrary precision. We investigate whether, due to its single timescale nature, this algorithm will have improved sample efficiency and convergence behavior compared to standard MAIQL.

2.6 Experimental Results

In this section, we compare our algorithms against both learning baselines (**WIBQL** (Avrachenkov & Borkar [12]) and **QL-λ=0**), and offline baselines (**Oracle LP**, **Oracle λ=0**, and **Oracle-LP-Index**).

WIBQL is designed to learn Whittle indexes for *binary*-action RMABs, but we adapt it to the multi-action setting by allowing it to plan using two actions, namely the passive action a_0 and a non-passive action a_j ($j > 0$) for the entire simulation. Clearly, this will be suboptimal in general, so we also design a stronger, multi-action baseline, **QL-λ=0**. This uses standard Q-learning to learn state-action values for each individual arm without reasoning about future costs or the shared budget between arms (i.e., $\lambda = 0$). At each step, the actions are chosen according to the knapsack procedure of LPQL. **Oracle λ=0** is the offline version of QL-λ=0 (i.e., it knows the transition probabilities). **Oracle LP** is the offline version of LPQL that solves Eq.2.11 using an LP solver, then follows the same knapsack procedure as LPQL. **Oracle-LP-Index** is an offline version of MAIQL that computes the multi-action indexes using an LP (see appendix B.3). Since the oracles are computationally

expensive, they are run for 1000 timesteps to allow their returns to converge, then are extrapolated.

All algorithms follow an ε-greedy paradigm for exploration where ε decays each round according to $\varepsilon_0 / \lceil \frac{t}{D} \rceil$ where ε_0 and D are constants. All algorithms were implemented in Python 3.7.4 and LPs were solved using Gurobi version 9.0.3 via the gurobipy interface[58]. All results are presented as the average (solid line) and interquartile range (shaded region) over 20 independently seeded simulations and were executed on a cluster running CentOS with Intel(R) Xeon(R) CPU E5-2683 v4 @ 2.1 GHz with 4GB of RAM.

2.6.1 Two Process Types

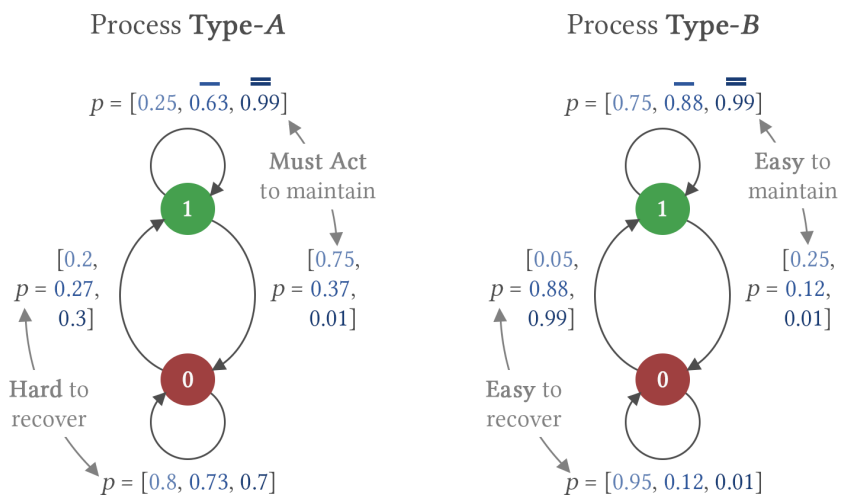


Figure 2.2: Two Process domain. Type-A arms need constant actions to stay in the good state (reward 1), whereas Type-B arms stay in the good state for many rounds after an action.

In the first experiment, we demonstrate how failing to account for cost and budget information while learning (i.e., QL-λ=0) can lead to poorly performing policies. The setting has two types of processes (arms), as in Fig. 2.2. Each has 3 actions, with costs 0, 1, and 2. Both arms have a good and bad state that gather 1 and 0 reward, respectively. The **Type-A** arm must be acted on every round while in the good state to stay there. However, in the bad state it is difficult to recover. This leads

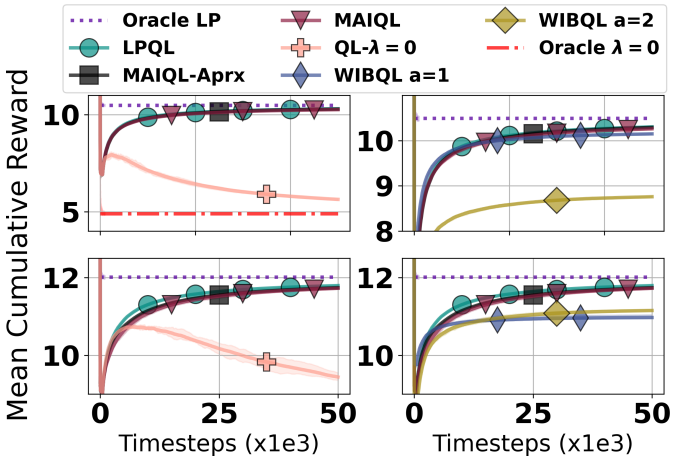


Figure 2.3: Results from Type-A v.s. Type-B domain with $N = 16$ and $B = 4$ (top row) and $B = 8$ (bottom row). Experiments in a row are the same, with different algorithms shown. Budget-agnostic learning converges to a highly suboptimal policy. Our algorithms converge to the best oracle policy, LPQL doing so the quickest. Binary-action planning underperforms except when a small budget forces the optimal policy to only use the cheapest action.

QL-λ=0 to learn that $Q(1, a_{j>0}, \lambda = 0) - Q(1, a_0, \lambda = 0)$ is large, i.e., acting in the good state is important for Type-A arms. Conversely, the **Type-B** arm will tend to stay in the good state even when not acted on, and when in the bad state, it can be easily recovered with any action. This leads QL-λ=0 to learn that $Q(1, a_{j>0}, \lambda = 0) - Q(1, a_0, \lambda = 0)$ is small. Thus QL-λ=0 will prefer to act on Type-A arms. However, if the number of Type-B arms is larger than the available budget, it is clearly better to spend the budget acting on Type-B arms since the action "goes farther", i.e., they may spend several rounds in the good state following only a single action, v.s. Type-A arms which are likely to only spend one round in the good state per action. Our budget-aware learning algorithms learn this tradeoff to converge to well-performing policies that greatly outperform cost-unaware planning.

We report the mean cumulative reward of each algorithm, i.e., its cumulative reward divided by the current timestep, averaged over all seeds. Fig. 2.3 shows the results with $N = 16$, with 25% of arms as Type-A and with 75% of arms as Type-B, over 50,000 timesteps. The top and bottom rows use $B = 4$ and $B = 8$, respectively. For ease of visual representation, each column shows

different combinations of algorithms – please note that the y-axis scales for each plot may be different. Fig. 2.4 shows results for the same arm type split and simulation length with $B = 8$, varying $N \in [16, 32, 48]$ (top to bottom). Parameter settings for each algorithm are included in appendix B.2. We see that each of our algorithms beat the baselines and converge in the limit to the Lagrange policy – equivalent to the multi-action index policy in this case – with the single-timescale algorithms converging quickest. Since the rewards obtained using Oracle-LP-Index coincide with Oracle LP, we do not plot the results for Oracle-LP-Index. Further, the plots demonstrate that the WIBQL algorithms underperform in general, except in cases where budgets are so small that the optimal policy effectively becomes binary-action (e.g., Fig 2.3 top right; $B = 4$). In the remaining experiments, WIBQL is similarly dominated and so is omitted for visual clarity. In both figures, interestingly, QL-λ=0 performs well at first while ε is large, suggesting that a random policy would outperform the $\lambda = 0$ policy. However, it eventually converges to Oracle-λ=0 as expected.

2.6.2 Random Matrices

The second experimental setting demonstrates LPQL's superior generality over index policies and its robustness to increases in the number of actions and variations in cost structure. In this setting, all transition probabilities, rewards, and costs are sampled uniformly at random, ensuring with high probability that the submodular action effect structure required for MAIQL's good performance will not exist. What remains to investigate is whether LPQL will be able to learn better policies than MAIQL in such a setting. Specifically, rewards for each state on each arm are sampled uniformly from $[0, 1]$, with $|\mathcal{S}| = 5$. Action costs are sampled uniformly from $[0, 1]^{|\mathcal{A}|}$, then we apply a cumulative sum to ensure that costs are increasing (but c_0 is set to 0). Fig. 2.5 shows results for $N = 16$ and $B = N|\mathcal{A}|/2$ as $|\mathcal{A}|$ varies in $[2, 5, 10]$ (top to bottom) over 50,000 timesteps. Note that B scales with $|\mathcal{A}|$ to ensure that optimal policies will include the additional action types, since the costs of the additional action types also scale with $|\mathcal{A}|$. Rewards are shown as a moving average with a

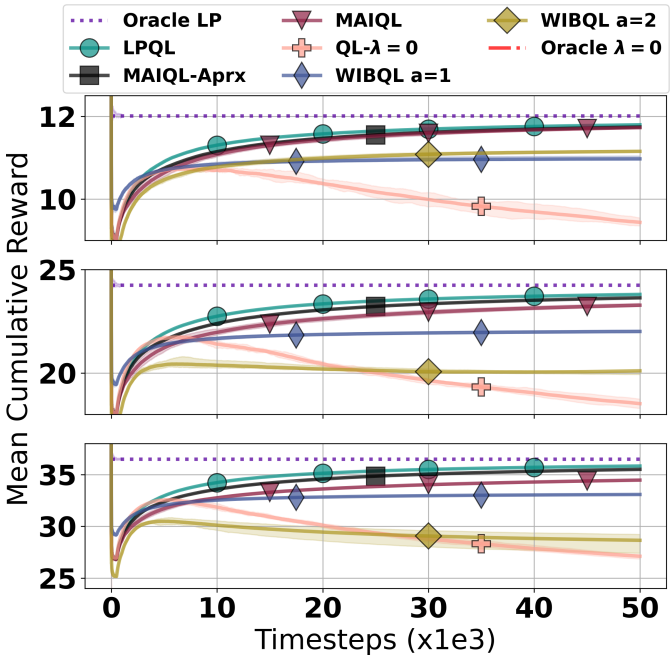


Figure 2.4: Results from the Two Process domain with $B = 8$ and $N \in [16, 32, 48]$ (top to bottom). Budget-agnostic converges to highly suboptimal policies, while our algorithms converge to the best oracle policy, with single-timescale versions doing so the quickest. Binary action planning underperforms with the $a = 2$ adaptation deteriorating as the budget becomes more constrained. Oracle $\lambda = 0$ (not shown) is dominated by all lines.

windows size of 100, which gives a clearer view of between-seed variance than the cumulative view. Fig. 2.5 shows that not only is LPQL able to learn much better policies than MAIQL and MAIQL-Aprx, which themselves converge to their oracle upper bound (Oracle-LP-Index), it does so with convergence behavior that is robust to increases in the number of actions, achieving near-optimal average returns at around 10k steps in each setting. Parameter settings for the different algorithms are again included in appendix B.2.

2.6.3 Medication Adherence

Finally, we run an experiment using data derived in-part from a real medication adherence domain[82]. The data contains daily 0-1 records of adherence from which transition probabilities can be

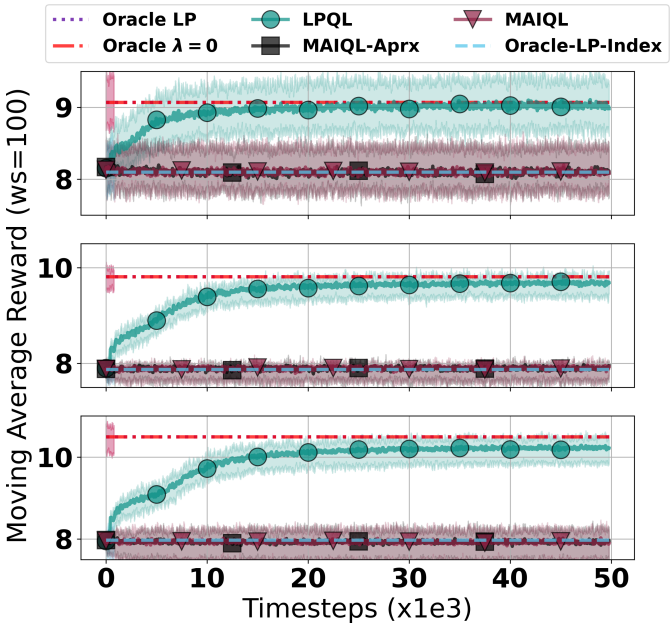


Figure 2.5: Moving average rewards from the random domain, for $|\mathcal{A}| \in [2, 5, 10]$ (top to bottom) using a window size (ws) of 100. Oracle LP and Oracle $\lambda = 0$ perform the same, as do MAIQL and MAIQL-Aprx. Oracle-LP-Index computes the index solution offline, demonstrating that MAIQL(-Aprx) are converging correctly, but the index policy performs poorly. LPQL converges quickly even as $|\mathcal{A}|$ increases.

estimated, assuming a corresponding 0-or-1 state (partial state history can also be accommodated). However, the data contains no records of actions and so must be simulated. In this experiment, we simulate actions that assume a natural "diminishing returns" structure in accordance with the assumption in section 2.4. One drawback is this estimation procedure creates uniform action effects across arms in expectation, i.e., a single "mode". However, in the real world we expect there to be multiple modes, representing patients' diverse counseling needs and response rates to various intervention types. To obtain multiple modes in a simple and interpretable way, we sample 25% of arms as Type-A arms from section 2.6.1, since they also have a binary state structure and are easily extended to accommodate partial state history. More details are given in appendix B.4. Note that, similar to Section 2.6.1, Oracle LP coincides with Oracle-LP-Index and hence, we do not plot the results for Oracle-LP-Index separately. Fig. 2.1 visualizes this domain.

Fig. 2.6 shows the results for the medication adherence domain with history lengths of 2, 3, and 4 (top to bottom), $N = 16$, $B = 4$, and 3 actions of cost 0, 1, and 2, over 100000 timesteps. Please see appendix B.2 for parameter settings. This demonstrates concretely that learning on a single timescale (LPQL and MAIQL-Aprx) clearly improves speed of convergence, and this becomes more pronounced as the size of the state space increases. To understand why, we analyzed the estimated transition matrices and found that many patients had values near 0 or 1. This makes it very rare to encounter certain states, making it difficult to obtain sufficient numbers of samples across all state action pairs for MAIQL's assumptions to hold, impeding its learning.

2.7 Conclusion

To the best of our knowledge, we are the first to provide algorithms for learning MARMABs in an online setting. We show that by following the traditional approaches to RMAB problems, i.e., seeking index policies in domains with structural assumptions, MAIQL is guaranteed to converge to the optimal solution as $t \to \infty$. However, it is not efficient, due to its two-timescale structure, and is limited in scope, due to its indexability assumption. We solve these challenges by going back to the fundamentals of RMABs to develop LPQL which works well regardless of the problem structure, and outperforms all other baselines in terms of both convergence rate and obtained reward. Towards a real-world RMAB deployment, our models would apply to settings that allow many repeat interactions over a long horizon, e.g., life-long medication adherence regimens[36]. However, since our algorithms require thousands of samples to learn, more work is needed to apply to many settings which may have short horizons. Still, this work lays a methodological and theoretical foundation for future work in online multi-action RMABs, a crucial step toward their real-world deployment.

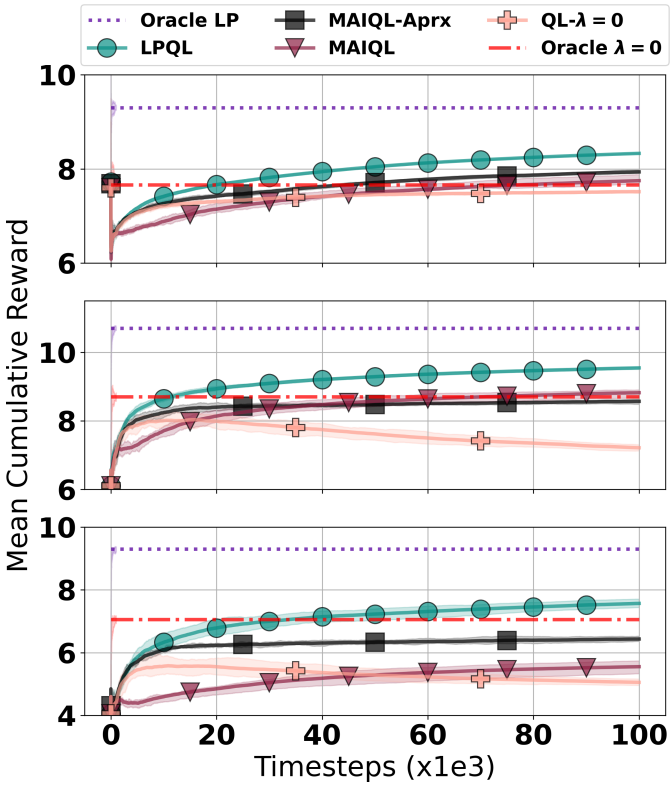


Figure 2.6: Mean cumulative reward on medication adherence domain with 16 patients, $B = 4$ and history length of 2, 3, and 4 (top to bottom). LPQL is the fastest to converge and converges to the best policies across all history lengths. MAIQL is slower to learn but does so eventually, where its approximate variant that learns on a single-timescale is more stable as the state size increases.

3

Learning to Recommend Interventions for Tuberculosis Patients using Digital Adherence Data

3.1 Introduction

The World Health Organization (WHO) reports that tuberculosis (TB) is among the top ten causes of death worldwide [167], yet in most cases it is a curable and preventable disease. The prevalence of TB is caused in part by non-adherence to medication, resulting in greater risk of death, reinfection and contraction of drug-resistant TB [151]. To combat non-adherence, the WHO recommends directly observed treatment (DOT), in which a health worker directly observes and confirms that a patient is consuming the required medication daily. However, requiring patients to travel to the DOT facility causes financial burden, and potentially social stigma due to public fear of the disease. Such barriers contribute to patients being lost to follow up, making TB eradication difficult. Thus, digital adherence technologies (DATs), which give patients flexible means to prove adherence, have gained global popularity [148].

DATs allow patients to be "observed" consuming their medication electronically, e.g. via two-way text messaging, video capture, electronic pillboxes, or toll-free phone calls. Health workers can then view real-time patient adherence on a dashboard such as Figure 3.1. In addition to improving patient flexibility and privacy, the dashboard enables health workers to triage patients and focus their

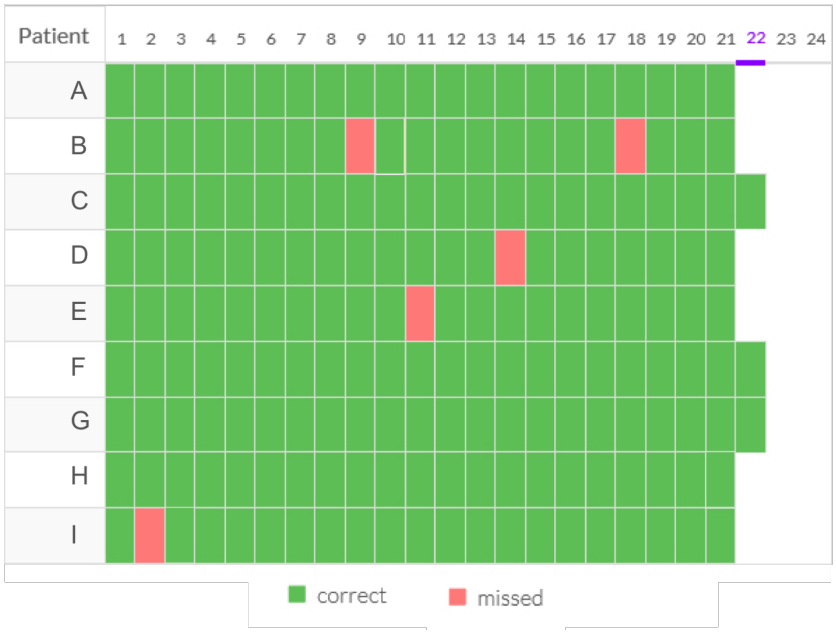


Figure 3.1: 99DOTS electronic adherence dashboard seen by health workers for a given month. Missed doses are marked in red while consumed doses are marked in green.

limited resources on the highest risk patients. Preliminary studies suggest that DATs can improve adherence in multiple disease settings [59,34,138], prompting its use and evaluation for managing TB adherence [49,97]. The WHO has even published a guide for the proper implementation of the technology in TB care [168].

In this chapter, we study how the wealth of longitudinal data produced by DATs can be used to help health workers better triage TB patients and deliver interventions to boost overall adherence of their patient cohort. The data we analyze is from Mumbai, India and comes from a partnership with the City TB Office of Mumbai; they have implemented a DAT by which patients prove adherence through daily toll-free calls. The DAT system was implemented with technical support from the healthcare technology company Everwell [45] and is known as 99DOTS [37]. In fact, Everwell supports implementations of 99DOTS throughout India where there were an estimated 2.7 million cases of TB in 2017 [167]. In Mumbai, patients enrolled in 99DOTS currently receive interventions according to

the following general guidelines. If they have not taken their medication by the afternoon, they (and their health worker) receive a text message reminder. If the patient still does not take their medication by some time later, the worker will call the patient directly. Finally, if a patient simply does not respond to these previous interventions after some number of days, they may be personally visited by a health worker. Note that many of these patients live in low-resource communities where each health worker manages tens to hundreds of patients; far more than they can possibly visit in a day. Thus, models that can identify patients at risk of missing doses and prioritize interventions by health workers are of paramount importance.

At first glance, the problem of predicting whom to target for an intervention appears to be a simple supervised machine learning problem. Given data about a patient's medication adherence through their calls to the 99DOTS system, one can train a machine learning model to predict whether they will miss medication doses in the future. However, such a model ignores the concurrent interventions from health workers as the data was collected, and can lead to incorrect prioritization decisions even when it is highly accurate. For instance, we might observe that missed doses are followed by a period of medication adherence: this does not mean that people with missed doses are more likely to take medication, but most likely that there was an intervention by a health worker after which the patient restarted their medication.

Thus, for *prescribing* interventions, we need to disentangle the effect of manual interventions from other underlying factors that result in missing a dose. However, since this data was collected via an extensive rollout to real patients, the data contains the effects of interventions carried out by health workers. As an additional challenge, health workers rarely record their interventions on the 99DOTS system, making it difficult to estimate their effects. While there is a well-developed literature on estimating heterogeneous treatment effects, standard techniques uniformly require knowledge of which patients received an intervention [113,38,11,149]. We note that such gaps will be common as countries eagerly adapt DAT systems in the hopes of benefiting low-income regions; to

support the delivery of improved care, we must be able to draw lessons from this messy but plentiful data.

In this work, therefore, we introduce *a general approach for learning from adherence data with unobserved interventions*, based on domain knowledge of the intervention heuristics applied by health workers. We construct a proxy for interventions present in the historical 99DOTS data and develop a model to help prioritize targets of interventions for health workers in three clinical scenarios:

Modeling Daily Non-Adherence Risk. We propose the prediction task: given adherence data up to a certain time period for patients not currently considered for intervention, predict risk of non-adherence in the next week. We then introduce machine learning models for this task, which enable health workers to accurately identify 21% more high-risk patients and catch nearly 76% more missed doses compared to the heuristics currently used in practice.

Predicting success of treatment. Next, we apply our framework to predict the final outcome at the end of the six-month treatment for a patient based on their initial adherence data. Like the previous model, this can be useful for health workers to prioritize patients who are at risk of an unfavorable outcome, even though their adherence might be high. Additionally, since this prediction applies over the course of several months (rather than just one week in the previous task), this model can be useful for public health officials to better plan for TB treatment in their area, e.g. by assigning or hiring additional health workers. We show that our model can be used to achieve city-wide treatment outcome goals at nearly 40% less cost than via baselines.

Decision Focused learning. Finally, building on recent work in end-to-end decision-focused learning[165], we build a machine learning model which is tailored for a specific intervention planning problem. In the planning problem, workers must balance travel costs while predicting which patients will benefit most from interventions. This example demonstrates how the modeling flexibility enabled by our approach allows us to fine-tune and extract additional gains for particular decision

support tasks (in this case, a 15% improvement over our earlier model).

With our proposed models, 99DOTS can leverage years of adherence data to better inform patient care and prioritize limited intervention resources. Additionally, the challenges we address are not unique to 99DOTS or TB adherence. DATs have been implemented for disease treatment regimens such as HIV and diabetes across the globe, and in each case health workers face the same challenge of prioritizing patient interventions. By enabling health workers to intervene before more missed doses, our model will directly contribute to saving the lives of those afflicted with TB and other diseases. That is why, though our model is not yet deployed, we are excited about our continued collaboration with the City TB Office of Mumbai and prospectively testing our model in the field.

3.2 Related Work

Outcomes and adherence research are well studied in the medical literature for a variety of diseases[75]. Traditionally, studies have attempted to identify demographic or behavioral factors correlated with non-adherence so that health workers can focus interventions on patients who are likely to fail. Tuberculosis in particular, given its lethality and prevalence in developing countries, has been studied throughout the world including in Ethiopia[142], Estonia[85], and India[136]. Typically these studies gather demographic and medical statistics on a cohort, observe their adherence and outcomes throughout the trial, then retrospectively apply survival[142,85] or logistic regression[136] analysis to determine covariates predictive of failure. Newer work has improved classification accuracy via machine learning techniques such as Decision Trees, Neural Networks, Support Vector Machines and more[74,66,139,105]. However, the conclusions connecting predictors to risk are largely the same as in previous medical literature. While such studies have improved patient screening at the time of diagnosis, they offer little knowledge about how risk changes *during* treatment. In

this work, we show how a patient's real-time adherence data can be used to track and predict risk changes throughout the course of their treatment. Previous studies likely did not address this question because accurately measuring patient adherence has historically been difficult.

However, in recent years, new technologies have made measuring daily adherence feasible in the context of many diseases such as HIV or stroke. One such common device is an electronic pill bottle cap that records the date/time when the cap is removed. While some previous work has used electronic cap data to determine predictors of non-adherence [127,125,33], almost no research has used the daily measurements made possible by the electronic cap to study changes in adherence over time. One study used the smart pillbox data to *retrospectively* categorize patient adherence [84], but our focus is on *prospective* identification of patients at risk of missing doses before failures occur. As such devices enter mainstream use, machine learning techniques like the ones we propose will play an important role in the treatment of a wide spectrum of diseases.

Methodologically, our work is related to the large body of research that deals with estimating the causal impact of interventions from observational data [113,38,11,149]. Given appropriate assumptions, such techniques allow for valid inferences about counterfactual outcomes under a different policy for determining interventions. However, they crucially require exact knowledge of when interventions were carried out. This information is entirely absent in our setting, requiring us to develop new methods for handling *unobserved* interventions in the training data.

3.3 Data Description

99DOTS provides each patient with a cover for every sleeve of pills that associates a hidden unpredictable phone number with each daily dose (note that one dose may consist of 2-5 pills). As patients expose pills associated with each dose, they expose one phone number per day. Each patient is instructed to place a toll-free call to the indicated number each day. 99DOTS counts a dose only

if the patient calls the correct number for a given day. Due to the sensitivity of the health domain, all data provided by our partners was fully anonymized before we received it. The dataset contains over 2.1 million dose records for about 17,000 patients, served by 252 health centers across Mumbai from Feb 2017 to Sept 2018. Table 3.1 summarizes the data. We now describe the available information in more detail.

Table 3.1: Data Summary. *Doses per patient was calculated only on patients enrolled at least 6 months before Sept 2018.

Metric	Count
Total doses recorded	2,169,976
—By patient call	1,459,908
—Manual (entered by health worker)	710,068
Registered phones	38,000
Patients	16,975
Health centers	252
Doses recorded per patient*	
—Quartiles	57/149/188
—Min/Mean/Max	1/136/1409
Active patients per center per month	
—Quartiles	7/18/35
—Min/Mean/Max	1/25/226

Patient Details. This is the primary record for patients who have enrolled with 99DOTS. The table includes demographic features such as weight-band, age-band, gender and treatment center ID. Also included are treatment start and end dates, whether treatment is completed or ongoing, and an "adherence string" which summarizes a patient's daily adherence. For patients who completed treatment, a treatment outcome is also assigned according to the standard WHO definitions [166] p. 5. We label "Cured" and "Treatment Complete" to be favorable outcomes and "Died", "Treatment failed", and "Lost to follow-up" to be unfavorable outcomes.

Mapping phone numbers to patients. Patients must call from a registered phone number for a dose to be counted by the 99DOTS system. Patients can register multiple phones, each of which will be noted in the Phone Map table. We filtered out phones that were registered to multiple patients since they could not be uniquely mapped to patients. Also, patients who had *any* calls from shared phones were filtered out to avoid analyzing incomplete call records. This removed <1% of the patients from the data set.

Call Log. The Call Log records every call received by 99DOTS, including from patients outside of Mumbai. It also includes "manual doses" marked by health workers. Manual doses allow workers to retroactively update a patient's adherence on the dashboard. For instance, if a patient missed a week of calls due to a cellular outage, the worker could update the record to account for those missed doses. We filtered the Call Log to only contain entries with patients and phones registered in Mumbai, then attached a Patient ID to each call by joining the filtered Call Log and Phone Map.

Patient Log. Each time a health worker interacts with a patient's data in the 99DOTS dashboard, an automatic note is generated describing the interaction. The Patient Log records each such event, noting the type of action, Patient ID, health worker ID, the health worker's medical unit, what action was taken, and a timestamp. We did not calculate features from this table as they tended to be sparse. However, this table was used for calculating our training labels as described in Section 3.4.

3.4 Unobserved Interventions

The TB treatment system operates under tight resource limitations, e.g. one health worker may be responsible for more than 100 patients. Thus, it is critical that workers be able to accurately rank patient risk and prioritize interventions accordingly. Machine learning can be used to accomplish such risk ranking with promising accuracy, but it requires taking special care to understand how

intervention resources were allocated in the existing data.

Therefore, a key challenge is that users of the 99DOTS platform generally do not record interventions: workers may make texts, calls, or personal visits to patients to try to improve adherence, but these interventions are not routinely logged in the data. While far from ideal, such gaps are inevitable as countries with differing standards of reporting adopt DATs for TB treatment. Given the abundance of data created by DATs and their potential to impact human lives, we emphasize the importance of learning lessons in this challenging setting where unobserved interventions occur. We next resolve this challenge by formulating a screening procedure which identifies patients who were likely candidates for particular interventions. However, we first illustrate the pitfalls of training and using a risk model in this domain without our screening procedure.

Consider a naive model trained on the data as-is. Some of the data will be influenced by the historical interventions carried out by health workers. Thus, such a model will learn how to predict patient adherence *given existing worker behaviors*. Now consider the model's intended use, namely to recommend a new prioritization of limited resources based on risk. Then in deployment, some patients who would have received interventions under the historical policy would be judged not to require intervention by the new model. While such prioritization is desirable under resource constraints, naive models which ignore the impact of interventions in the dataset can actually *worsen* patient outcomes. For instance, assume we use the naive model to make a prediction about the patient from Section 3.1 who had a week of missed doses, an intervention, then a week of good adherence. By correctly predicting this patient's good adherence the naive model would recommend no intervention – but this patient's good adherence is *contingent on* the hidden intervention in the data. *Hence, the naive model will take resources away from exactly the patients who would benefit most.* To avoid such pitfalls arising from unobserved interventions, we must train and evaluate on data that is not influenced by such intervention effects. We now describe our general method for reshaping data around intervention effects to build valid models.

Intervention Proxy. Our goal is to use the available data to formulate a proxy for when an intervention is likely to have occurred, so that we can train our models on data points which are unaffected by interventions. *The key is to identify a conservative estimate for where interventions occur to ensure that data with intervention signals are not included.* First, we draw a distinction between different types of health worker interventions. Specifically, we consider a house visit to be a "resource-limited" intervention since workers cannot visit all of their patients in a timely manner. Generally, this is a last resort for health workers when patients will not respond to other methods. Alternatively, we consider calls and texts to be "non-resource-limited" interventions since they could feasibly be made on a large number of patients at very low cost. Note that the naive model in the previous section *could* make valid recommendations for actions *in addition* to normal health worker behaviors. For this reason, we develop a proxy only for resource-limited interventions since non-resource-limited interventions come virtually for "free".

To formulate our proxy, we first searched for health worker guidelines for carrying out house visits. The 2005 guide by India's Revised National Tuberculosis Control Program (RNTCP)[135] required that workers deliver a house visit after a single missed dose, but updated guides are far more vague on the subject. Both the most recent guide by the WHO[168] and by the RNTCP[134] leave house visits up to the discretion of the health worker. However, through discussions in Mumbai we learned that health workers prioritize non-adherent patients for resource-limited interventions such as house visits. Thus, we formulated our proxy based on the adherence dashboard seen by health workers.

The 99DOTS dashboard gives a daily "Attention Required" value for each patient. First, if a patient has an entry in the Patient Log (i.e. provider made a note about the patient) in the last 7 days they are automatically changed to "MEDIUM" attention, but this rule affects <1% of the labels. The remaining 99% of labels are as follows: If a patient misses 0 or 1 doses in the last 7 days, they are changed to "MEDIUM" attention, whereas if they miss 4 or more they are changed to "HIGH"

attention. Patients with 2-3 missed doses retain their attention level from the previous day. As our conservative proxy, we assumed that only "HIGH" attention patients were candidates for resource-limited interventions since the attention level is a health worker's primary summary of recent patient adherence. This "Attention Required" system for screening resource-limited interventions is generalizable to any daily adherence setting; one need only to identify the threshold for a change to HIGH attention.

With this screening system, we can identify sequences of days during which a patient was a candidate for a resource-limited intervention, and subsequently avoid using signal from those days in our training task. We accomplish this with our formulation of the real-time risk prediction task as follows.

Consider a given set of patients on the dashboard of a health worker at day t. Each patient will have an "Attention Required" value in {MEDIUM, HIGH} representing their risk for that day. Over the course of the next week up to $t + 7$, we will observe call behavior for each patient and so the attention for each patient may also change each day. Between $t + 1$ and $t + 7$, any patient that is at HIGH on a given day may receive a resource-limited intervention while those at MEDIUM may not. Note that a change from MEDIUM to HIGH on day t_i where $t + 1 \leq t_i \leq t + 7$ means that a patient missed 4 doses over days $[t_i - 6, t_i]$. Patients at HIGH attention are already known to the health worker, so the goal for our ML system is to help prevent transitions from MEDIUM to HIGH by predicting which patients are at greatest risk before the transition occurs and allowing a health worker to intervene early.

We formalize our prediction task as follows. For each patient who is MEDIUM at time t, use data from days $[t - 6, t]$ to predict whether or not they change to HIGH at any time t_i where $t + 1 \leq t_i \leq t + 7$. We now demonstrate that, with our intervention proxy, resource-limited intervention effects cannot effect labels in this formulation. First, if a patient stays at MEDIUM for all t_i, then the label is 0. Since the patient was at MEDIUM for all t_i, our proxy states that no resource-limited

intervention took place between our prediction time t and the time that produced the label, $t + 7$. Second, if a patient changes from MEDIUM to HIGH on day t_i, then on day t_i we establish that the label is 1. By our proxy, any resource-limited intervention effect must happen in $[t_i + 1, t + 7]$, since attention is established at the end of a day t_i. So again, we have that no resource-limited intervention took place between our prediction time t and the time that produced the label, t_i.

Since we ensure that no resource-limited interventions happen between our prediction time and the time the label is generated, we ensure that intervention effects cannot influence our labels. Now, if we predict that a patient will have good adherence we can safely recommend no intervention since our combined screening and training method guarantees that their good adherence *is not contingent on* an intervention. Thus our classifier is suited to make predictions that prioritize resource-limited interventions.

Despite messy data affected by unobserved interventions, this conservative, general proxy generates clean data without interventions. In the next section, we show how this approach leads to significant improvements in prediction performance and creates valid recommendations to enable interventions among patients at immediate risk of becoming non-adherent.

3.5 Real-Time Risk Prediction

We now build a model for the prediction task formalized in Section 3.4 which leverages our intervention screening proxy. We aimed to develop a model corresponding to the health worker's daily task of using their patients' recent call history to evaluate adherence risk with the goal of scheduling different types of interventions. Better predictions allow workers to proactively intervene with more patients before they miss critical doses.

Sample Generation. We started with the full population of 16,975 patients and generated training samples from each patient as follows. We considered all consecutive sequences of 14 days of call

data where the first 7 days of each sequence were non-overlapping. We excluded each patient's first 7 days and the last day of treatment to avoid bias resulting from contact with health workers when starting or finishing treatment. We then took two filtering steps. First, we removed samples where the patient had more than 2 doses manually marked by a provider during the input sequence since these patients likely had contact with their provider outside of the 99DOTS system. Second, we removed samples in which the patient did not miss any doses in the input sequence. These samples made up the majority of data but included almost no positive (HIGH risk) labels, which distorted training. Further, positive predictions on patients who missed 0 doses are unlikely to be useful; no resource-limited intervention can be deployed so widely that patients with perfect recent adherence are targeted. The above procedure generated 16,015 samples (2,437 positive).

Features. Each sample contained a time-series of call data and static features. The time series included two sequences of length 7 for each sample. First was a binary sequence of call data (1 for a call or manual dose and 0 for a miss.) The second sequence was a cumulative total of all doses missed up to that day, considering the patient's full history in the program. The static features included four demographic features from the Patient Table: weight-band, age-band, gender, and treatment center ID. Additional features were engineered from the patient Call Logs and captured a patient's *behavior* rather than just their adherence. For example, does the patient call at the same time every morning or sporadically each day? This was captured through the mean and variance of the call minute and hour. Other features included number of calls, number of manual doses, mean/max/variance of calls per day as well as days per call. We also included analogous features which used only *unique* calls per day (i.e. calls *to* unique phone numbers), or ignored manual doses. This process resulted in 29 descriptive features.

Models. We first tested standard models which use only the static features: linear regression, a random forest [124] (with 100 trees and a max depth of 5), and a support vector machine. The random forest performed best, so we exclude the others for clarity. To leverage the time series data we

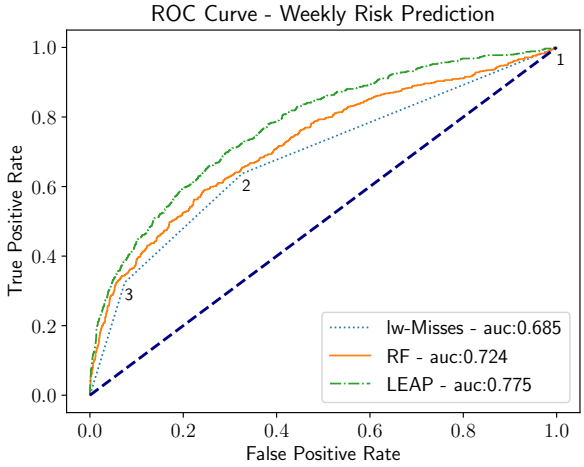


Figure 3.2: ROC Curve for the weekly risk prediction task comparing the missed call baseline (blue), Random Forest (yellow) and LEAP (green). Numbers under the blue curve give thresholds used to calculate the baseline's ROC curve.

also built a deep network, named LEAP (Lstm rEal-time Adherence Predictor), implemented with Keras[30] which takes both the time series and static features as input. LEAP has two input layers: 1) a LSTM with 64 hidden units for the time series input and 2) a dense layer with 100 units for the static feature input. We concatenated the outputs of these two layers to feed forward into another dense layer with 16 units, followed by a single sigmoid activation unit. We used a batch size of 128 and trained for 20 epochs.

Model Evaluation. To evaluate models we randomized all data then separated 25% as the test set. We used 4-fold grid search to determine the best model parameters. To deal with class imbalance, we used SMOTE to over-sample the training set[29] implemented with the Python library imblearn[93]. We also normalized features as percentiles using SKLearn[124] which we found empirically to work well. The baseline we compared against was the method used by the existing 99DOTS platform to asses risk, namely doses missed by the patient in the last week (lw-Misses).

Figure 3.2 shows the ROC curve of our models vs. the baseline. The random forest narrowly outperforms the baseline and LEAP clearly outperforms both. However, to evaluate the usefulness

Table 3.2: LEAP vs. Baseline - Missed Doses Caught

Method	True Positives	Doses Caught
Baseline	204	204
LEAP	248	360
Improvement	**21.6%**	**76.5%**

LEAP vs. baseline for catching missed doses with a fixed false positive rate. Our method learns behaviors indicative of non-adherence far earlier than the baseline, allowing for more missed doses to be prevented.

Table 3.3: LEAP vs. Baseline: Additional Interventions

TPR	Baseline FPR	LEAP FPR	Improvement
75%	50%	35%	**30%**
80%	63%	41%	**35%**
90%	82%	61%	**26%**

LEAP vs. baseline for implementing new interventions. At any TPR LEAP improves over the baseline FPR, allowing for more precisely targeted interventions.

of our methods over the baseline, we consider how each method might be used to plan house-visit interventions. Since this is a very limited resource, we set the strictest baseline threshold to consider patients for this intervention; that is 3 missed calls. Fixing the FPR of this baseline method, Table 3.2 shows how many more patients in the test set would be reached each week by our method (as a result of its higher TPR) as well as the improvement in number of missed doses caught. To calculate missed doses caught, we count only missed doses that occur before the patient moves to HIGH risk. *Our model catches 21.6% more patients and 76.5% more missed doses, demonstrating substantially more precise targeting than the baseline.*

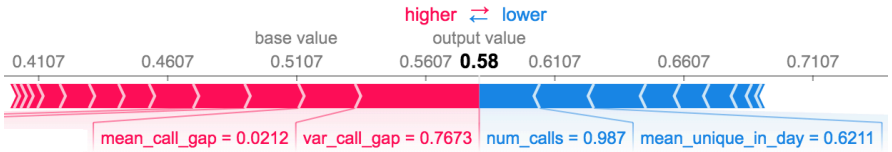
Table 3.3 shows that our model also outperforms the baseline as both the true positive rate

(TPR) and FPR increase, showcasing our model's greater discriminatory power. This is useful for non-resource-limited interventions such as calls or texts. Recall, that our screening procedure does not apply to this type of intervention, so our predictions may only recommend *additional* interventions. It is important that additional interventions be carefully targeted since repeated contact with a given patient reduces the efficacy of each over time[39]. This highlights the value of the greater precision offered by our model, since simply blanketing the entire population with calls and texts is likely counterproductive.

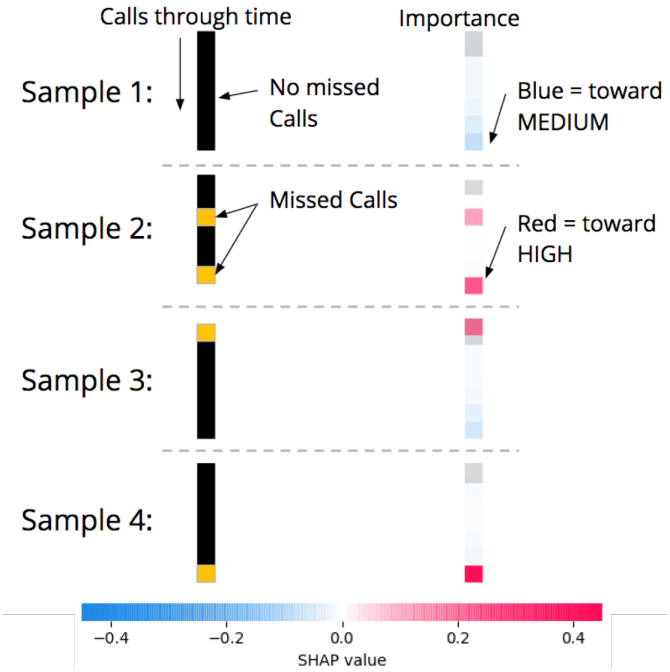
Interpretability. Our model has the potential to catch more missed doses than current methods. However, these gains cannot become reality without health workers on the ground delivering interventions based on the predictions. Interpretability is thus a key factor in our model's usefulness because health workers need to understand *why* our model makes its predictions to trust the model and integrate its reasoning with their own professional knowledge.

However, the best predictive performance was achieved with LEAP, a black-box network, rather than a natively interpretable model like linear regression. Accordingly, we show how a visualization tool can help users draw insights about our model's reasoning. We used the SHapley Additive exPlanations (SHAP) python library, which generates visualizations for explaining machine learning models[100]. Figure 3.3a shows how static features influence our model's prediction, where red features push predictions toward 1 (HIGH) and blue toward 0 (MEDIUM). Recall that features are scaled as percentiles. In the blue, we see that this patient makes an above-average number of calls each week pushing the prediction toward 0. However, in the red we see that this patient has a *very low average* but a *high variability* in time between calls. These features capture that this patient missed two days of calls, then made three calls on one day in an attempt to "back log" their previous missed calls. Our model learned that this is a high-risk behavior.

Figure 3.3b shows four different samples as input to the LSTM layer of our model. The left shows the binary input sequence as colored pixels where black is a call and yellow is a missed call.



(a) SHAP values for LEAP's dense layer features for a high-risk sample ($\geq$0.5).



(b) SHAP values for LEAP's LSTM layer input for 4 samples.

Figure 3.3: Visualization of the (a) dense layer and (b) LSTM layer of our weekly risk prediction model. Red values correspond to inputs that push predictions toward output of 1; blue values push toward output of 0.

On the right are SHAP values corresponding to each day of adherence data, and grey denotes the start of the call sequence. We see that the model learned that calls later in the week carry more weight than calls earlier in the week. In Sample 1, the bottom two pixels (the most recent calls) have blue SHAP values while the other pixels have SHAP value close to 0. In Sample 3, a single missed call at the beginning of the week combined with a call made at the end of the week result in essentially cancelling SHAP values. Sample 4 also has one missed call but on the last day of the week, resulting in a net positive SHAP value.

This visualization technique provides intuitive insights about the rules learned by our model. In deployment, workers could generate these visualizations for any sample on the fly in order to aid their decision-making process.

3.6 Outcome Prediction

Next we investigate how adherence data can be used to predict final treatment outcome. Traditional TB treatment studies model outcomes only as they relate to patient covariates such as demographic features. Exploiting daily real-time adherence data provided by DATs, we investigate how using the first k days of a patient's adherence enables more accurate, personalized outcome predictions. Note that intervention effects are still present in this formulation. However, our screening procedure will not apply since we predict over a period of several months, during which virtually all patients would have had repeated in-person contact with health workers.

Sample Generation and Features. We formalize the prediction task as follows: given the first k days of adherence data, predict the final binary treatment outcome. We considered "Cured" and "Treatment Complete" as favorable outcomes and "Died", "Lost to follow-up", and "Treatment Failure" as unfavorable outcomes. We only include patients who were assigned an outcome from these categories. Further, since patients with the outcome "Died" or "Lost to follow-up" exit the

program before the full 6 months of treatment, we removed those who were present for less than $k + 1$ days. Our final dataset contained 4167 samples with 433 unfavorable cases.

Through discussions in Mumbai, we learned that health workers often build a sense of a patient's risk of an unfavorable outcome within their first month of treatment. To model this process, we set k=35 for our prediction task, capturing the first month of each patient's adherence after enrollment in 99DOTS. (Note that this is not a general rule for health workers, but simply served as a motivation for our choice of k in this task.) Both the static features and the sequence inputs were the same as calculated for the weekly prediction task, but now taken over the initial 35 days. We included two versions of the health worker baseline: missed doses in the last week (lw-Misses) and total missed doses in 35 days (t-Misses).

Model Evaluation. We used the same models, grid search design, training process, and evaluation procedure as before. For the Random Forest we used 150 trees and no max depth. For LEAP, we used 64 hidden units for the LSTM input layer, 48 units for the dense layer input, and 4 units in the penultimate dense layer.

Figure 3.4 shows ROC curves for each model. Even the very simple baseline of counting the calls made in the last 7 days before the 35 day cutoff is fairly predictive of outcome suggesting that the daily data made available by DATs is valuable in evaluating which patients will fail from TB treatment. Our ML models display even greater predictive power, with LEAP performing the best, followed closely by the random forest. We highlight how LEAP's predictive power could help officials minimize the costs necessary to reach medical outcome goals for their city. For example, say Mumbai launches a new initiative to catch 80% of unfavorable outcomes (true positives in Figure 3.4) by hiring new health staff. Over the 17,000 patients in Mumbai, where 10% have unsuccessful outcomes as in our test set, an 80% catch rate requires saving 1360 patients. Using either baseline, achieving the 80% TPR requires a FPR of 70%, i.e., hiring additional staff to support *10710* total patients in this example scenario. However, using LEAP only incurs a FPR of 42%, translating to

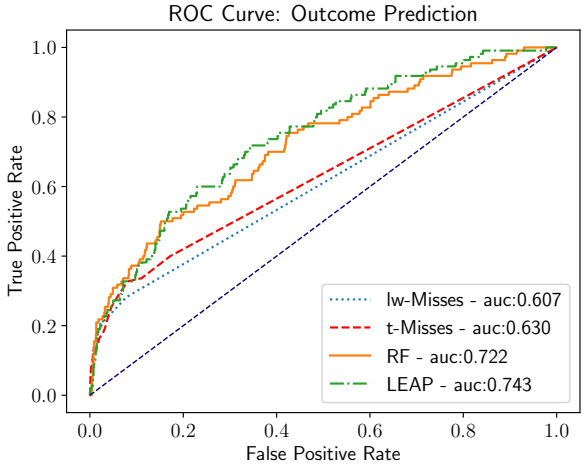


Figure 3.4: ROC curves for outcome prediction models.

6426 total patients. Recall that in Mumbai, the average health worker cares for about 25 patients. At a yearly starting salary of ₹216,864[28] (or $3026) our model would yield ₹37M in saved costs (or $525,000) per year.

3.7 Decision Focused Learning

We now explore a case study of how our LEAP model can be specialized to provide decision support for a particular intervention. We exploit end-to-end differentiability of the model to replace our earlier loss function (binary cross-entropy) with a performance metric tailored to the objective and constraints of specific decision problem. To accomplish this end-to-end training, we leverage recent advances in *decision-focused learning*, which embeds an optimization model in the loop of machine learning training[165,41].

We focus on a specific optimization problem that models the allocation of health workers to intervene with patients who are at risk in the near future. This prospective intervention is enabled by our real-time risk predictions and serves as an example of how our system can enable proactive, tar-

geted action by providers. However, we emphasize that our system can be easily modified to capture other intervention problems. Such flexibility is one benefit to our technical approach, which allows the ML model to *automatically* adapt to the problem specified by a domain expert.

Our optimization problem models a health worker who plans a series of interventions over the course of a week. The health worker is responsible for a population of patients across different locations, and may visit one location each day. We use location identifiers at the level of the TB Unit since this is the most granular identifier which is shared by the majority of patients in our dataset. Visiting a location allows the health worker to intervene with any of the patients at that location. The optimization problem is to select a set of locations to visit which maximizes the number of patients who receive an intervention *on or before the first day they would have missed a dose*. We refer to this quantity as the number of *successful interventions*, which we choose as our objective for two reasons. First, it measures the extent to which the health worker can proactively engage with patients before adherence suffers. Second, this objective only counts patients who start the week at MEDIUM attention and receive an intervention before they could have transitioned to HIGH, dovetailing with our earlier discussion on avoiding unobserved interventions in the data. This extends our earlier intervention proxy to handle day-by-day rewards.

We now show how this optimization problem can be formalized as a linear program. We have a set of locations $i = 1 \ldots L$ and patients $j = 1 \ldots N$ where patient j has location ℓ_j. Over days of the week $t = 1 \ldots 7$, the objective coefficient c_{jt} is 1 if an intervention on day t with patient j is successful and 0 otherwise. Our decision variable is x_{it}, and takes the value 1 if the health worker visit location i

on day t and 0 otherwise. With this notation, the final LP is as follows:

$$\max_x \sum_{t=1}^{7} \sum_{i=1}^{L} x_{it} \left(\sum_{j:\ell_j=i} c_{jt} \right)$$
$$\text{s.t.} \sum_{i=1}^{L} x_{it} \leq 1, t = 1...7$$
$$\sum_{t=1}^{7} x_{it} \leq 1, i = 1...L$$
$$0 \leq x_{it} \leq 1 \quad \forall i, t$$

where the second constraint prevents the objective from double-counting multiple visit to a location. We remark that the feasible region of the LP can be shown to be equivalent to a bipartite matching polytope, implying that the optimal solution is always integral.

The machine learning task is to predict the values of the c_{jt}, which are unknown at the start of the week. We compare three models. First, we extend the lw-Misses baseline to this setting by thresholding the number of doses patient j missed in the last week, setting $c_{jt} = 0$ for all t if this value falls below the threshold τ and $c_{jt} = 1$ otherwise. We used $\tau = 1$ since it performed best. Second, we trained our LEAP system directly on the true c_{jt} as a binary prediction task using cross-entropy loss. Third, we trained LEAP to predict c_{jt} using performance on the above optimization problem as the loss function (training via the differentiable surrogate given by Wilder et al. [165]). We refer to this model as LEAP-Decision.

We created instances of the decision problem by randomly partitioning patients into groups of 100, modeling a health worker under severe resource constraints (as they would benefit most from such a system). We included all patients, including those with no missed doses in the last week, since the overall resource allocation problem over locations must still account for them.

Figure 3.5 shows results for this task. In the top row, we see that LEAP and LEAP-Decision both

outperform lw-Misses, as expected. LEAP-Decision improves the number of successful interventions by approximately 15% compared to LEAP, demonstrating the value of tailoring the learned model to a given planning problem. LEAP-Decision actually has worse AUC than either LEAP or lw-Misses, indicating that typical measures of machine learning accuracy are not a perfect proxy for utility in decision making. To investigate what specifically distinguishes the predictions made by LEAP-Decision, the bottom row of Figure 3.5 shows scatter plots of the predicted utility at each location according to LEAP and LEAP-Decision versus the true values. Visually, LEAP-Decision appears better able to distinguish the high-utility outliers which are most important to making good decisions. Quantitatively, LEAP-Decision's predictions have worse correlation with the ground truth overall (0.463, versus 0.519 for LEAP), but better correlation on locations where the true utility is strictly more than 1 (0.504 versus 0.409). Hence, decision-focused training incentivizes the model to focus on making accurate predictions specifically for locations that are likely to be good candidates for an intervention. This demonstrates the benefit of our flexible machine learning modeling approach, which can use custom-defined loss functions to automatically adapt to particular decision problems.

3.8 Discussion

We present a framework for learning to make intervention recommendations from data generated by DAT systems applied to TB care. We develop a general approach for learning from medical adherence data that contains unobserved interventions and leverage this approach to build a model for predicting risk in multiple settings. In the real-time adherence setting, we show that our model would allow health workers to more accurately target interventions to high risk patients sooner – catching 21% more patients and 76% more missed doses than the current heuristic baseline. Next, we train our model for outcome prediction, showing how adherence data can more accurately de-

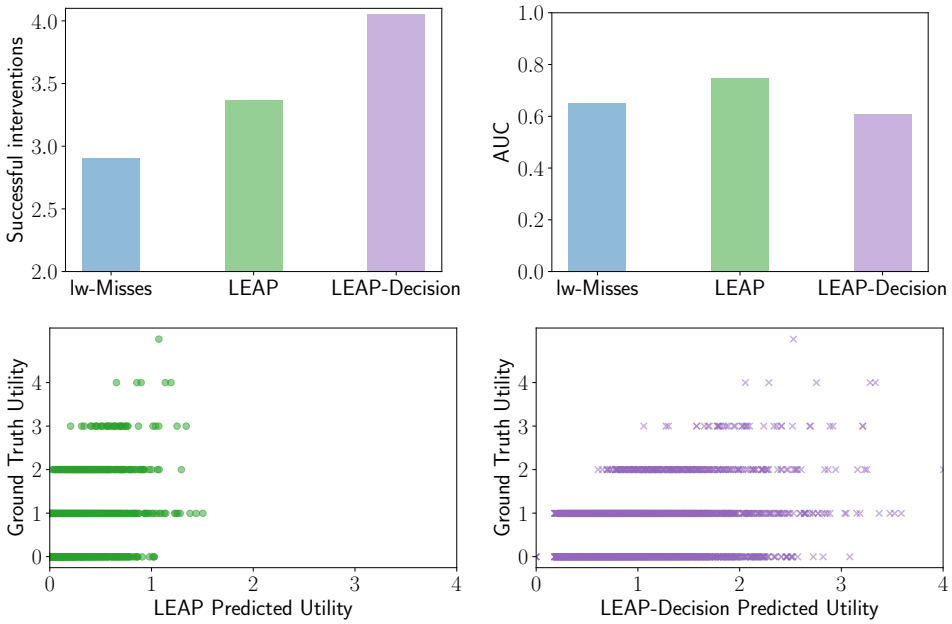


Figure 3.5: Results for decision focused learning problem. Top row: successful interventions and AUC for each method. Bottom row: visualizations of model predictions.

tect patients at risk of unfavorable treatment outcomes. We finally show that tailoring our model for a specific intervention via decision-focused learning can improve performance by a further 15%. The learning approaches we present here are general and could be leveraged to study data generated by DATs as applied to any medication regimen. With the growing popularity of DAT systems for TB, HIV, Diabetes, Heart Disease, and other medications, we hope to lay the groundwork for improved patient outcomes in healthcare settings around the world.

4

Collapsing Bandits and Their Application to Public Health Interventions

4.1 Introduction

Motivation. This chapter considers scheduling problems in which a planner must act on k out of N binary-state processes each round. The planner fully observes the state of the processes on which

she acts, then all processes undergo an action-dependent Markovian state transition; the state of the process is unobserved until it is acted upon again, resulting in uncertainty. The planner's goal is to maximize the number of processes that are in some "good" state over the course of T rounds. This class of problems is natural in the context of *monitoring tasks* which arise in many domains such as sensor/machine maintenance[68,54,1,155], anti-poaching patrols[130], and especially healthcare. For example, nurses or community health workers are employed to monitor and improve the adherence of patient cohorts to medications for diseases like diabetes[118], hypertension[24], tuberculosis[132,27] and HIV[77,76]. Their goal is to keep patients adherent (i.e., in the "good" state) but a health worker can only intervene on (visit) a limited number of patients each day. Health workers can play a similar role in monitoring and delivering interventions for patient mental health, e.g., in the context of depression[98,114] or Alzheimer's Disease[95].

We adopt the solution framework of *Restless Multi-Arm Bandits* (RMABs), a generalization of Multi-Arm Bandits (MABs) in which a planner may act on k out of N arms each round that each follow a Markov Decision Process (MDP). Solving an RMAB is PSPACE-hard in general[122]. Therefore, a common approach is to consider the Lagrangian relaxation of the problem in which the $\frac{k}{N}$ budget constraint is dualized. Solving the relaxed problem gives Lagrange multipliers which act as a greedy index heuristic, known as the Whittle index, for the original problem. Specifically, the *Whittle index policy* computes the Whittle index for each arm, then plays the top k arms with the largest indices. The Whittle index policy has been shown to be asymptotically optimal (i.e., $N \to \infty$ with fixed $\frac{k}{N}$) under a technical condition[161] and generally performs well empirically[9] making it a common solution technique for RMABs.

Critically, using the Whittle index policy requires two key components: (i) a fast method for computing the index and (ii) proving the problem satisfies a technical condition known as *indexability*. Without (i) the approach can be prohibitively slow, and without (ii) asymptotic performance guarantees are sacrificed[161]. Neither (i) nor (ii) are known for general RMABs. Therefore,

to capture the scheduling problems addressed in this work, we introduce a new subclass of RMABs, *Collapsing Bandits*, distinguished by the following feature: when an arm is played, the agent fully observes its state, "collapsing" any uncertainty, but when an arm is passive, no observation is made and uncertainty evolves. We show that this RMAB subclass is more general than previous models and leads to new theoretical results, including conditions under which the problem is indexable and under which optimal policies follow one of two simple threshold types. We use these results to develop algorithms for quickly computing the Whittle index. In experiments, we analyze the algorithms' performance on (i) data from a real-world healthcare scheduling task in which our approach ties state-of-the-art performance at a fraction the runtime and (ii) various synthetic distributions, some of which the algorithm achieves performance comparable to the state of the art even outside its optimality conditions.

To summarize, our contributions are as follows: (i) We introduce a new subclass of RMABs, Collapsing Bandits, (ii) Derive theoretical conditions for Whittle indexability and for the optimal policy to be threshold-type, and (iii) Develop an efficient solution that achieves a 3-order-of-magnitude speedup compared to more general state-of-the-art RMAB techniques, without sacrificing performance.

4.2 Restless Multi-Armed Bandits

An RMAB consists of a set of N arms, each associated with an MDP with two actions [129]. An MDP $\{\mathcal{S}, \mathcal{A}, r, P\}$ consists of a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a state-dependent reward function $r : \mathcal{S} \rightarrow \mathbb{R}$, and a transition function P, where $P^a_{s,s'}$ denotes the probability of transitioning from state s to s' when action a is taken. An MDP *policy* $\pi : \mathcal{S} \rightarrow \mathcal{A}$ represents a choice of action to take at each state. We consider both discounted and average reward criteria. The long-term *discounted reward* starting from state $s_0 = s$ is defined as $R^{\pi}_{\beta}(s) = E\left[\sum_{t=0}^{\infty} \beta^t r(s_{t+1} \sim T(s_t, \pi(s_t), s_{t+1}) | \pi, s_0 = s\right]$

where $\beta \in [0, 1)$ is the discount factor and actions are selected using π. To define average reward, let $f^{\pi}(s) : \mathcal{S} \rightarrow [0, 1]$ denote the *occupancy frequency* induced by policy π, i.e., the fraction of time spent in each state of the MDP. The *average reward* $\overline{R}^{\pi}$ of policy π be defined as the expected reward computed over the occupancy frequency: $\overline{R}^{\pi} = \sum_{s \in \mathcal{S}} f^{\pi}(s) r(s)$.

Each arm in an RMAB is an MDP with the action set $\mathcal{A} = \{0, 1\}$. Action 1 (0) is called the *active* (*passive*) action and denotes the arm being pulled (not pulled). The agent can pull at most k arms at each time step. The agent's goal is to maximize either her discounted or average reward across the arms over time. Some RMAB problems need to account for partial observability of states. It is sufficient to let the MDP state be the *belief state*: the probability of being in each latent state[73]. While intractable in general due to infinite number of reachable belief states, most partially observable RMABs studied (including our Collapsing Bandits) have polynomially many belief states due to a finite time horizon or other structures.

Related work. RMABs have been an attractive framework for studying various stochastic scheduling problems since Whittle indices were introduced[163]. Because RMABs are PSPACE-hard[122], studies usually consider restricted classes under which some performance guarantees can be derived. Collapsing Bandits form one such novel class that generalizes some existing results which we note in later sections. Liu & Zhao[96] develop an efficient Whittle index policy for a 2-state partially observable RMAB subclass in which the state transitions are unaffected by the actions taken and reward is accrued from the active arms only. Akbarzadeh & Mahajan[6] define a class of bandits with "controlled restarts," giving indexability results and a method for computing the Whittle index. However, "controlled restarts" define the active action as state independent, a stronger assumption than Collapsing Bandits which allow state-dependent action effects. Glazebrook et al.[54] give Whittle indexability results for three classes of restless bandits: (1) A machine maintenance regime with deterministic active action effect (we consider stochastic active action effect) (2) A switching regime in which the passive action freezes state transitions (in our setting, states always change re-

gardless of action) (3) A reward depletion/replenishment bandit which deterministically resets to a start state on passive action (we consider stochastic passive action effect). Hsu [64] and Sombabu et al. [145] augment the machine maintenance problem from Glazebrook et al. [54] to include either i.i.d. or Markovian evolving probabilities of an active action having no effect, a limited form of state-dependent action. Meshram et al. [109] introduce Hidden Markov Bandits which, similar to our approach, consider binary state transitions under partial observability, but do not allow for state dependent rewards on passive arms. In sum, our Collapsing Bandits introduce a new, more general RMAB formulation than special subclasses previously considered. Qian et al. [130] present a generic approach for any indexable RMAB based on solving the (partially observable) MDPs on arms directly. Because we derive a closed form for the Whittle index, our algorithm is orders of magnitude faster.

4.3 Collapsing Bandits

We introduce *Collapsing Bandits* (CoB) as a specially structured RMAB with partial observability. In CoB, each arm $n \in \{1, \ldots, N\}$ has binary latent states $\mathcal{S} = \{0, 1\}$, representing *bad* and *good* state, respectively. The agent acts during each of finite days $t \in 1, \ldots, T$. Let $a_t \in \{0, 1\}^N$ denote the vector of actions taken by the agent on day t. Arm n is said to be *active* at t if $a_t(n) = 1$ and *passive* otherwise. The agent acts on k arms per day, i.e., $\|a_t\| = k$, where $k \ll N$ because resources are limited. When acting on arm n, the true latent state of n is fully observed by the agent and thus its uncertainty "collapses" to a realization of the binary latent state. We denote this observation as $\omega \in \mathcal{S}$. States of passive arms are completely unobservable by the agent.

Active arms transition according to the *transition matrix* $P^{a,n}_{s,s'}$ and passive arms transition according to $P^{p,n}_{s,s'}$. We drop the superscript n when there is no ambiguity. Our scheduling problem, like many problems in analogous domains, exhibits the following natural structure: (i) processes are

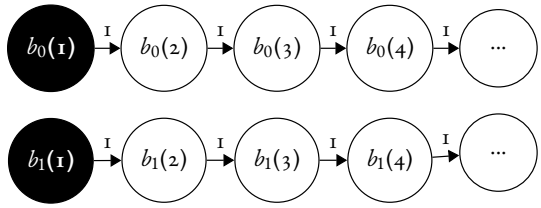


Figure 4.1: Belief-state MDP under the policy of always being passive. There is one chain for each observation $\omega \in \{0, 1\}$ with the head marked black. Belief states deterministically transition down the chains.

more likely to stay "good" than change from "bad" to "good"; (ii) when acted on, they tend to improve. These natural structures are respectively captured by imposing the following constraints on P^p and P^a for each arm: (i) $P^p_{0,1} < P^p_{1,1}$ and $P^a_{0,1} < P^a_{1,1}$; (ii) $P^p_{0,1} < P^a_{0,1}$ and $P^p_{1,1} < P^a_{1,1}$. To avoid unnecessary complication through edge cases, all transition probabilities are assumed to be nonzero. The agent receives reward $r_t = \sum_{n=1}^{N} s_t(n)$ at t, where $s_t(n)$ is the latent state of arm n at t. The agent's goal is to maximize the long term rewards, either discounted or average, defined in Sec. 4.2.

Belief-State MDP Representation In limited observability settings, belief-state MDPs have organized chain-like structures, which we will exploit. In particular, the only information that affects our belief of an arm being in state 1 is the number of days since that arm was last pulled and the state ω observed at that time. Therefore, we can arrange these belief states into two "chains" of length T, each for an observation ω. A sketch of the belief state chains under the passive action is shown in Fig. 4.1. Let $b_\omega(u)$ denote the belief state, *i.e., the probability that the state is* 1, if the agent received observation $\omega \in \{0, 1\}$ when it acted on the process u days ago. Note that $b_\omega(u)$ is also the expected reward associated with that belief state, and let $\mathcal{B}$ be the set of all belief states.

When the belief-state MDP is allowed to evolve under some policy, the following mechanism arises: first, after an action, the state ω is observed (uncertainty "collapses"), then one round passes causing the agent's belief to become $P^a_{\omega,1}$, representing the head of the chain determined by ω. Subsequent passive actions cause the process to transition deterministically down the same chain

(though, the transition in the latent state is still stochastic). Then when the process's arm is active, it transitions to the head of one of the chains with probability equal to the belief that the corresponding observation would be emitted (see Fig. 4.2a for an illustration).

The belief associated with a belief state can be calculated in closed form with the given transition probabilities. Formally,

$$b_\omega(u) = \tau_{u-1}(P^a_{\omega,1})\ \forall u \in [T] \text{ where } \tau_u(b) = \frac{P^p_{0,1} - (P^p_{1,1} - P^p_{0,1})^u (P^p_{0,1} - b(1 + P^p_{0,1} - P^p_{1,1}))}{(1 + P^p_{0,1} - P^p_{1,1})} \tag{4.1}$$

4.4 Collapsing Bandits: Threshold Policies and Whittle Indexability

Because of the well-known intractability of solving general RMABs, the widely adopted solution concept in the literature of RMABs is the Whittle index approach; for a comprehensive description, see Whittle [163]. Intuitively, the Whittle index captures the value of acting on an arm in a particular state by finding the minimum *subsidy* m the agent would accept to *not act*, where the subsidy is some exogenous "donation" of reward. Formally, the modified reward function becomes $r_m : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, where $r_m(s, 0) = r(s) + m$ and $r_m(s, 1) = r(s)$. Let $R^\pi_{\beta,m}(s) = E\left[\sum_{t=0}^\infty \beta^t r_m(s_t, \pi(s_t)) | \pi, s_0 = s\right]$ and $\overline{R}^\pi_m = \sum_{s \in \mathcal{S}} f^\pi(s) r_m(s, \pi(s))$ be the discounted and average reward criteria for this new subsidy setting, respectively. The former is maximized by the discounted value function (we give a value function for the average reward criterion in **Fast Whittle Index Computation**):

$$V_m(b) = \max \begin{cases} m + b + \beta V_m(\tau_1(b)) & \text{passive} \\ b + \beta(b V_m(P^a_{1,1}) + (1-b) V_m(P^a_{0,1})) & \text{active} \end{cases} \tag{4.2}$$

where τ is defined in Eq. 4.1 and b is shorthand for $b_\omega(u)$. In a CoB, the Whittle index of a belief state b is the smallest m s.t. it is equally optimal to be active or passive in the current state. Formally:

$$W(b) = \inf_m \{m : V_m(b; a = 0) \geq V_m(b; a = 1)\} \tag{4.3}$$

Critically, performance guarantees hold only if the problem satisfies *indexability*[161,163], a condition which says that for all states, the optimal action cannot switch to active as m increases. Let Π_m^* be the set of policies that maximize a given reward criterion under subsidy m.

Definition 4.4.1 (Indexability)**.** *An arm is indexable if $\mathcal{B}^*(m) = \{b : \forall \pi \in \Pi_m^*, \pi(b) = 0\}$ monotonically increases from $\emptyset$ to the entire state space as m increases from $-\infty$ to ∞. An RMAB is indexable if every arm is indexable.*

The following special type of MDP policy is central to our analysis.

Definition 4.4.2 (Threshold Policies)**.** *A policy is a* forward (reverse) threshold policy *if there exists a threshold b_{th} such that $\pi(b) = 0$ ($\pi(b) = 1$) if $b > b_{th}$ and $\pi(b) = 1$ ($\pi(b) = 0$) otherwise.*

Theorem 4.4.3. *If for each arm and any subsidy $m \in \mathbb{R}$, there exists an optimal policy that is a forward or reverse threshold policy, the Collapsing Bandit is indexable under discounted and average reward criteria.*

Proof Sketch. Using linearity of the value function in subsidy m for any fixed policy, we first argue that when forward (reverse) threshold policies are optimal, proving indexability reduces to showing that the threshold monotonically decreases (increases) with m. Unfortunately, establishing such a monotonic relationship between the threshold and m is a well-known challenging task in the literature that often involves problem-specific reasoning[96]. Our proof features a sophisticated induction argument exploiting the finite size of $\mathcal{B}$ and relies on tools from real analysis for limit arguments.

□

All formal proofs can be found in the appendix. We remark that Thm. 4.4.3 generalizes the result in the seminal work by Liu & Zhao [96] who proved the indexability for a special class of CoB. In particular, the RMAB in Liu & Zhao [96] can be viewed as a CoB setting with $P^a = P^p$, i.e., transitions are independent of actions.

Though the Whittle index is known to be challenging to compute in general [163], we are able to design an algorithm that computes the Whittle index efficiently assuming the optimality of threshold policies, which we now describe.

Fast Whittle Index Computation The main algorithmic idea we use is the Markov chain structure that arises from imposing a *forward* threshold policy on an MDP. A forward threshold policy can be defined by a tuple of the first belief state in each chain that is less than or equal to some belief threshold $b_{th} \in [0, 1]$. In the two-observation setting we consider, this is a tuple $(X_0^{b_{th}}, X_1^{b_{th}})$, where $X_\omega^{b_{th}} \in 1, \ldots, T$ is the index of the first belief state in each chain where it is optimal to act (i.e., the belief is less than or equal to b_{th}). We now drop the superscript b_{th} for ease of exposition. See Fig. 4.2a for a visualization of the transitions induced by such an example policy. For a forward threshold policy (X_0, X_1), the occupancy frequencies induced for each state $b_\omega(u)$ are:

$$f^{(X_0,X_1)}(b_\omega(u)) = \begin{cases} \alpha & \text{if } \omega = 0, u \leq X_0 \\ \beta & \text{if } \omega = 1, u \leq X_1 \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

$$\alpha = \left(\frac{(X_1 b_0(X_0))}{1 - b_1(X_1)} + X_0 \right)^{-1}, \beta = \left(\frac{X_1 b_0(X_0)}{1 - b_1(X_1)} + X_0 \right)^{-1} \frac{b_0(X_0)}{1 - b_1(X_1)} \tag{4.5}$$

These equations are derived from standard Markov chain theory. These occupancy frequencies do not depend on the subsidy. Let $f_m^{(X_0,X_1)}$ be the average reward of policy (X_0, X_1) under subsidy m.

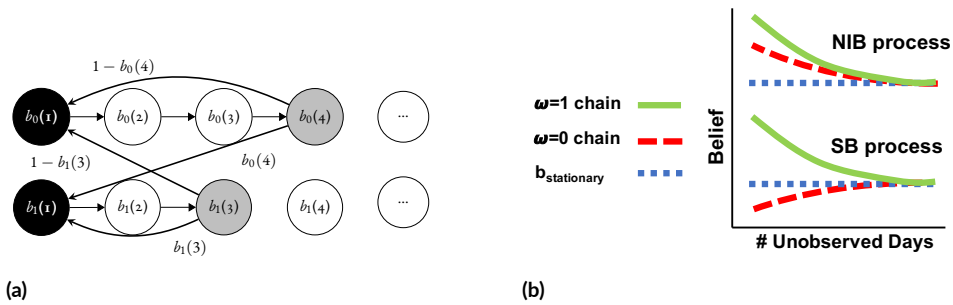


Figure 4.2: (a) Visualization of forward threshold policy ($X_0 = 4, X_1 = 3$). Black nodes are the head of each chain and grey nodes are the thresholds. (b) Non-increasing belief (NIB) process has non-increasing belief in both chains. A split belief process (SB) has non-increasing belief after being observed in state 1, but non-decreasing belief after being observed in state 0.

We decompose the average reward into the contribution of the state reward and the subsidy

$$J_m^{(X_0,X_1)} = \sum_{b \in \mathcal{B}} b f^{(X_0,X_1)}(b) + m(1 - f^{(X_0,X_1)}(b_1(X_1)) - f^{(X_0,X_1)}(b_0(X_0))) \tag{4.6}$$

Recall that for any belief state $b_\omega(u)$, the Whittle index is the smallest m for which the active and passive actions are both optimal. Given forward threshold optimality, this translates to two corresponding threshold policies being equally optimal. Such policies must have adjacent belief states as thresholds, as can be concluded from Lemma C.1.0.1 in Appendix C.1. Note that for a belief state $b_0(X_0)$ the only adjacent threshold policies with active and passive as optimal actions at $b_0(X_0)$ are (X_0, X_1) and $(X_0 + 1, X_1)$ respectively. Thus the subsidy which makes these two policies equal in value must thus be the Whittle Index for $b_0(X_0)$, which we obtain by solving: $J_m^{(X_0,X_1)} = J_m^{(X_0+1,X_1)}$ for m. We use this idea to construct two fast Whittle index algorithms.

Sequential index computation algorithm Alg. 4.4.1 precomputes the Whittle index of every belief state for each process, having time complexity $\mathcal{O}(|\mathcal{S}|^2 TN)$. Then, the per-round complexity to retrieve the top k indices is $\mathcal{O}(N \min\{k, log(N)\})$. This gives a great improvement

over the more general method given by Qian et al.[130] (our main competitor) which has per-round complexity of $\approx \mathcal{O}(N\log(\frac{1}{\varepsilon})(|\mathcal{S}|T)^{2+\frac{1}{18}})$, where $\log(\frac{1}{\varepsilon})$ is due to a bifurcation method for approximating the Whittle index to within error ε on each arm and $(|\mathcal{S}|T)^{2+\frac{1}{18}}$ is due to the best-known complexity of solving a linear program with $|\mathcal{S}|T$ variables[69].

Alg. 4.4.1 is optimized for settings in which the Whittle index can be precomputed. However, for online learning settings, we give an alternative method in Appendix C.6 that computes the Whittle index on-demand, in a closed form.

Algorithm 4.4.1: Sequential index computation algorithm

```
Initialize counters to heads of the chains: X_1 = 1, X_0 = 1
while X_1 < T or X_0 < T do
    Compute m_1 := m such that J_m^(X_0,X_1) = J_m^(X_0,X_1+1)
    Compute m_0 := m such that J_m^(X_0,X_1) = J_m^(X_0+1,X_1)
    Set i = arg min{m_0, m_1} and W(X_i) = min{m_0, m_1}
    Increment X_i
```

Our algorithm also requires that belief is decreasing in X_0 and X_1. Formally, we require:

Definition 4.4.4 (Non-increasing belief (NIB) processes)**.** *A process has* non-increasing belief *if, for any $u \in [T]$ and for any $\omega \in \mathcal{S}$, $b_\omega(u) \geq b_\omega(u+1)$.*

All possible CoB belief trends are shown in Fig. 4.2b. We make this distinction because the computation of the Whittle index in Alg. 4.4.1 is guaranteed to be exact for NIB processes that are also forward threshold optimal, though we show empirically that our approach works surprisingly well for most distributions. In the next section, we analyze the possible forms of optimal policies to find conditions under which threshold policies are optimal.

Types of Optimal Policies Analyzing Eq. 4.2 reveals that at most three types of optimal policies exist. This follows directly from the definition of $V_m(b)$, which is a max over the passive action value function and the active action value function. The former is convex in b, a well-known POMDP result [146], and the latter is linear in b. Thus, as shown in Fig. 4.3, there are three ways in which the value functions of each action may intersect; this defines three optimal policy forms of *forward*, *reverse* and *dual* threshold types, respectively. Forward and reverse threshold policies are defined in Def. 4.4.2; dual threshold policies are active between two separate threshold points and passive elsewhere. Not only do threshold policies greatly reduce the optimization search space, they often admit closed form expressions for the index as demonstrated earlier in this section. We now derive sufficient conditions on the state transition probabilities under which each type of policy is verifiably optimal.

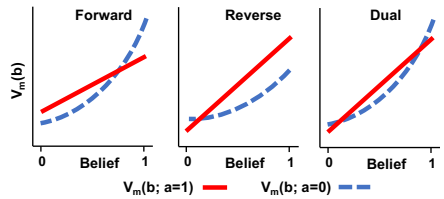


Figure 4.3: Components of $V_m(b)$ in Eq. 4.2. Since the passive action is convex in b, active action is linear in b, and value function is a max over these, at most three optimal policy types are possible.

Theorem 4.4.5. *Consider a belief-state MDP corresponding to an arm in a Collapsing Bandit. For any subsidy m, there is a* forward *threshold policy that is optimal under the condition:*

$$(P^p_{1,1} - P^p_{0,1})(1 + \beta(P^a_{1,1} - P^a_{0,1}))(1 - \beta) \geq P^a_{1,1} - P^a_{0,1} \tag{4.7}$$

Proof Sketch. Forward threshold optimality requires that if the optimal action at a belief b is passive, then it must be so for all $b' > b$. This can be established by requiring that the derivative of the passive action value function is greater than the derivative of the active action value function w.r.t. b. The main challenge is to distill this requirement down to measurable quantities so the final condition can be easily verified. We accomplish this by leveraging properties of $\tau(b)$ and using induction to derive both upper and lower bounds on $V_m(b_1) - V_m(b_2) \; \forall \, b_1, b_2$ as well as a lower bound on

$\frac{d(V_m(b))}{db}$. □

Intuitively, the condition requires that the intervention effect on processes in the "bad" state must be large, making $P^a_{1,1} - P^a_{0,1}$ small. Note that Liu & Zhao [96] consider the case where $P^a_{1,1} = P^p_{1,1}$ and $P^a_{0,1} = P^p_{0,1}$, which makes Eq. 4.7 always true. Thus we generalize their result for threshold optimality.

Theorem 4.4.6. *Consider a belief-state MDP corresponding to an arm in a Collapsing Bandit. For any subsidy m, there is a* reverse *threshold policy that is optimal under the condition:*

$$(P^p_{1,1} - P^p_{0,1})\Big(1 + \frac{\beta(P^a_{1,1} - P^a_{0,1})}{1-\beta}\Big) \leq P^a_{1,1} - P^a_{0,1} \tag{4.8}$$

Intuitively, the condition requires small intervention effect on processes in the "bad" state, the opposite of the forward threshold optimal requirement. Note that both Thm. 4.4.5 and Thm. 4.4.6 also serve as conditions for the average reward case as $\beta \to 1$ (a proof based on Dutta's Theorem [42] is given in Appendix C.4).

Conjecture 1. *Dual threshold policies are never optimal for Collapsing Bandits.*

This conjecture is supported by extensive numerical simulations over the random space of state transition probabilities, values of β, and values of subsidy m; its proof remains an open problem. Note that this would imply that all Collapsing Bandits are indexable.

4.5 Experimental Evaluation

We evaluate our algorithm on several domains using both real and synthetic data distributions. We test the following algorithms: **Threshold Whittle** is the algorithm developed in this chapter. **Qian et al.** [130], a slow, but precise general method for computing the Whittle index, is our main baseline

that we improve upon. **Random** selects k process to act on at random each round. **Myopic** acts on the k processes that maximize the expected reward at the immediate next time step. Formally, at time t, this policy picks the k processes with the largest values of $\Delta b_t = (b_{t+1}|a = 1) - (b_{t+1}|a = 0)$. **Oracle** fully observes all states and uses Qian et al. [130] to calculate Whittle indices. We measure performance in terms of *intervention benefit*, where 0% corresponds to the reward of a policy that is always passive and 100% corresponds to Oracle. All results are averaged over 50 independent trials.

4.5.1 Real Data: Monitoring Tuberculosis Medication Adherence

We first test on tuberculosis medication adherence monitoring data, which contains daily adherence information recorded for each real patient in the system, as obtained from Killian et al. [82]. The "good" and "bad" states of the arm (patient) correspond to "Adhering" and "Not Adhering" to medication, respectively. State transition probabilities are estimated from the data. Because this data is noisy and contains only the adherence records and not the intervention (action) information (as the authors state), we perturb the computed average transition matrix by reducing (increasing) $P_{\omega,1}$ by a uniform random number between 0 and δ_1, δ_2 (δ_3, δ_4) then renormalizing to obtain $P^{p}_{\omega,1}$ ($P^{a}_{\omega,1}$) for the simulation. Reward is measured as the undiscounted sum of patients (arms) in the adherent state over all rounds, where each trial lasts $T = 180$ days (matching the length of first-line TB treatment) with N patients and a budget of k calls per day. All experiments in this section set all δ to 0.05.

In Fig. 4.4a, we plot the runtime in seconds vs the number of patients N. Fig. 4.4b compares the intervention benefit for $N = 100, 200, 300, 500$ patients and $k = 10\%$ of N. In the $N = 200$ case, the runtimes of a single trial of Qian et al. and Threshold Whittle index policy are 3708 seconds and 3 seconds, respectively, while attaining near-identical intervention benefit. Our algorithm is thus 3 orders of magnitude faster than the previous state of the art without sacrificing performance.

We next test Threshold Whittle as the resource level k is varied. Fig. 4.4c shows the performance

in the $k = 5\%N$, $k = 10\%N$ and $k = 15\%N$ regimes ($N = 200$). Threshold Whittle outperforms Myopic and Random by a large margin in these low resource settings. We also affirm the robustness of our algorithm to δ, the perturbation parameter used to approximate real-world $P^{p}_{\omega,1}$ and $P^{a}_{\omega,1}$ from the data, and present the extensive sensitivity analysis in Appendix C.7. Finally, in Appendix C.6 we couple our algorithm to a Thompson Sampling-based learning approach and show it performs well in the real-world case where transition probabilities would need to be learned online, supporting the deployability of our work.

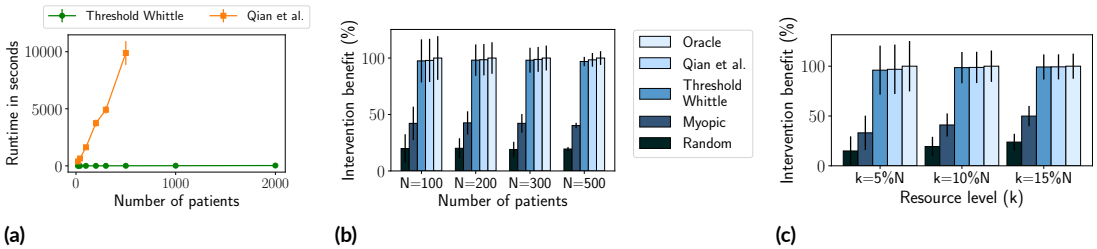


Figure 4.4: (a) Threshold Whittle is several orders of magnitude faster than Qian et al. and scales to thousands of patients without sacrificing performance on realistic data (b). (c) Intervention benefit of Threshold Whittle is far larger than naive baselines and nearly as large as Oracle.

4.5.2 Synthetic Domains

We test our algorithm on four synthetic domains, that potentially characterize other healthcare or relevant domains, and highlight different phenomena. Specifically, we: (i) identify situations when Myopic fails completely while Whittle remains close to optimal, (ii) analyze the effect of latent state entropy on policy performance, (iii) identify limitations of Threshold Whittle by constructing processes for which Threshold Whittle shows separation from Oracle, and (iv) test robustness of our algorithm outside of the theoretically guaranteed conditions. To facilitate comparison with the real data distribution, we simulate trials for $T = 180$ rounds where reward is the undiscounted sum of arms in state 1 over all rounds. We consider the space of transition probabilities satisfying the

assumed natural constraints, as outlined in Sec. 4.3.

Fig. 4.5a demonstrates a domain characterized by processes that are either self-correcting or non-recoverable. Self-correcting processes have a high probability of transitioning from state 0 to 1 regardless of the action taken, while non-recoverable processes have a low chance of doing so. We show that when the immediate reward is larger for the former than the latter, Myopic can perform even worse than Random. That is because a myopic policy always prefers to act on the self-correcting processes per their larger immediate reward, while Threshold Whittle, capable of long-term planning, looks to avoid spending resources on these processes. In this regime, the best long-term plan is to always act on the non-recoverable processes to keep them from failing. Analytical explanation of this phenomenon is presented in Appendix C.5. We set the resource level, $k = 10\% N$ in our simulation for Fig. 4.5a. Note that performance of Myopic drops as the fraction of self-correcting processes becomes larger and reaches a minimum at $x = 100\% - k = 90\%$. Beyond this point, Threshold Whittle can no longer completely avoid self-correcting processes and the gap subsequently starts to decrease.

Fig. 4.5b explores the effect of uncertainty in the latent state on long-term planning. For each point on the x-axis, we draw all transition probabilities according to $P^p_{\omega,1}, P^a_{\omega,1} \sim [x, x + 0.1]$. The entropy of the state of a process is maximum near 0.5 making long term planning most uncertain and as a result, this point shows the biggest gap with Oracle, which can observe all the states in each round. Note that Myopic and Whittle policies perform similarly, as expected for (nearly) stochastically identical arms.

Fig. 4.5c studies processes that have a large propensity to transition to state 0 when passive and a corresponding low active action impact, but a significantly larger active action impact in state 1. This makes it attractive to exclusively act on processes in the 1 state. This simulates healthcare domains where a fraction of patients degrade rapidly, but can recover, and indeed respond very well to interventions if already in a good state. To simulate these, we draw transition matrices with

$P^p_{0,1}, P^p_{1,1}, P^a_{0,1} \sim [0.3, 0.32]$ and $P^a_{1,1} \sim [0.7, 0.72]$ in varying proportions and sample the rest from the real TB adherence data. Because the best plan is to act on processes in state 1, both Myopic and Whittle act on the processes with the largest belief giving Oracle a significant advantage as it has perfect knowledge of states.

Although we provide theoretical guarantees on our algorithm for forward threshold optimal processes with non-increasing belief, Fig. 4.5d reveals that Alg. 4.4.1 performs well empirically even with these conditions relaxed. Here, we sample processes uniformly at random from the state transition probability space, and use rejection sampling to vary the proportion of threshold optimal processes. Threshold Whittle performs well even when as few as 20% of the processes are forward threshold optimal; we briefly analyze this phenomenon in Appendix C.8.

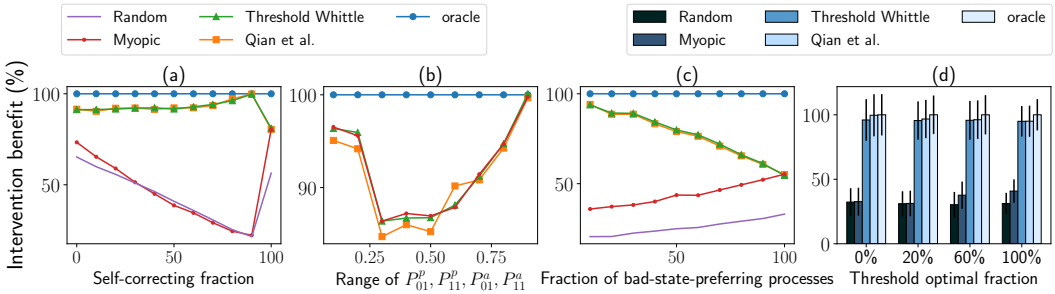


Figure 4.5: (a) Myopic can be trapped into performing even worse than Random while Threshold Whittle remains close to optimal. (b) Long-term planning is least effective when entropy of states is maximum. (c) Myopic and Whittle planning become similar when more processes are prone to failures. (d) Threshold Whittle is surprisingly robust to processes even outside of theoretically guaranteed conditions.

4.6 Conclusion

We open a new subspace of Restless Bandits, *Collapsing Bandits*, which applies to a broad range of real-world problems, especially in healthcare delivery. We give new theoretical results that cover a large portion of real-world data as well as an algorithm that runs thousands of times faster than the state of the art without sacrificing performance. We simultaneously also recognize limitations of our

theoretical results, which become narrow in the average reward case. We envision several interesting avenues for future work, including techniques to incorporate the user/health worker inputs for planning, generalizing our inherently 2-state approach to allow for a multi-state model, and allowing multiple actions and/or more general reward functions.

Broader Impact

Our work is largely motivated by resource constrained health intervention delivery. This setting is common across low, middle, and high-income countries, in which community health workers (CHWs) are recruited to deliver basic care to a cohort of patients or benefactors. In fact, CHWs have been critical in achieving global health initiatives for over five decades, and evidence shows that CHWs have had a positive impact in myriad domains including maternal and newborn health [31,43], (non-)communicable diseases [31,143], and sexual/reproductive health [169] in low-resource communities across the world [40,43,143,162]. Our modeling has the potential to improve the delivery of care in these highly resource-constrained settings.

Figure 4.6: CHW delivering vaccine. Credit: Pippa Ranger.

However, a deployment of our system to any setting must be done responsibly. For instance, we designed our system with the intention of *assisting* human CHWs plan resource-limited interventions. That said, we present results that highlight our algorithm's ability to plan for thousands of processes at a time, far more than for which a human could independently plan. Just making this capability available could encourage the automation of applicable interventions via automated calls or texts, potentially displacing CHW jobs, reducing human contact with patients, and unfairly limiting care for patients with limited access to technology.

Additionally, users of the system must be dutifully aware that its recommendations will be based solely on the data entered in the system. In the context of medication adherence monitoring, if the worker enters incorrect data, e.g., the patient was adhering ("good" state) but they instead mark the patient as not adhering ("bad" state), then the algorithm could make the wrong recommendation about the patient the next day, since its belief of the patient's adherence would also be wrong.

Finally, our AI system is inherently a blackbox which would likely be replacing an interpretable scheduling heuristic. This would limit any user or administrator's ability to audit decisions around why certain patients were recommended for intervention. As with any potential deployment of a blackbox system to a domain that affects the allocation of resources to humans, system designers should be acutely aware of the balance between their needs to be able to perform audits vs. their need for optimization.

5

Restless and Uncertain: Robust Policies for Restless Bandits via Deep Multi-Agent Reinforcement Learning

5.1 Introduction

Restless multi-armed bandits (RMABs), a model for constrained resource allocation among N independent stochastic processes (arms), are widely studied. Traditionally a *binary-action* problem, in which a planner decides whether or not to act on each of N arms, here we consider the *multi-action* generalization [81,52] which more accurately captures challenging real-world planning problems. Salient examples of RMABs include scheduling [13,172], machine replacement [54,137], aerial vehicle routing [91], anti-poaching patrol planning [130], and healthcare [92,103]. While these works have established important theoretical foundations, they share one key limitation: assuming stochastic dynamics are precisely known. Having exact knowledge of dynamics is impossible in many real-world problems. For example, in healthcare intervention planning, the probability that a patient will adhere to treatment after receiving an intervention is not perfectly known *a priori*; in anti-poaching patrol planning, the probability of finding a poacher's snare at some location is not known with certainty.

Accordingly, methods have been developed to learn RMAB policies *online*, assuming no *a priori* knowledge [71,157]. However, these methods require tens of thousands of samples to converge to good

policies which is prohibitive for many real-world problems, e.g., in finite-length treatment settings such as tuberculosis [103] with only a few dozen rounds. Instead, real-world planners must make the most of noisy data at hand, estimating dynamics from historical data or consulting experts, inducing significant *uncertainty*. RMAB techniques can be used to plan with point estimates, but we show that ignoring uncertainty can lead to arbitrarily bad policies.

To address these shortcomings and push RMABs toward wider real-world applicability, we introduce *Robust RMABs*, a generalization of RMABs which allows stochastic dynamics to be specified as uncertainty intervals, rather than point estimates. This new problem is very computationally demanding, adding a combinatorial layer of complexity onto an already PSPACE-hard problem [122]. Addressing this complexity gives rise to a rich set of challenges that necessitates the design of new techniques that not only help solve the robust objective we analyze, but also are of general interest to RMAB research.

Concretely, we plan under a *minimax regret* objective, using a double oracle (DO) framework [106] that has seen success in problems involving a *single* Markov decision process (MDP) [170]. The DO approach casts the robust planning problem as a zero-sum game between an *agent* oracle and adversarial *nature* oracle. However, existing techniques fail for any non-trivially sized RMABs since the state and action spaces grow combinatorially in the number of arms N and resource constraint B, respectively. Specifically, given S-sized state spaces for each arm, the full combinatorial problem has state space of size S^N and action space–and thus policy-network output–of size $\binom{N}{B}$ (for binary-action RMAB; action space is larger with multi-action). At this size, we found that directly applying Xu et al. [170] to solve the full combinatorial problem as a single process fails to learn good policies for RMABs as small as $N = 5$ arms, with $B = 3$ and $S = 2$. Moreover, under the minimax regret objective, the nature oracle is a particularly difficult challenge as it requires jointly searching the RMAB policy space and the continuous, uncertain space of transition probabilities. Previously, this objective has been posed as a non-stationary RL problem and solved heuristically with a single pol-

icy network [170]. We improve the nature oracle by formulating it as a multi-agent RL problem and develop a novel solution method for RMABs. In summary, our contributions are:

1. We introduce the Robust RMAB problem with interval uncertainty over arm dynamics and develop techniques to solve a minimax regret objective via double oracle.
2. To enable the DO approach, we introduce DDLPO, a novel deep RL algorithm for RMABs, of general interest. DDLPO tackles the combinatorial complexity of RMABs by learning an auxiliary "λ-network" in tandem with individual arm policy networks, which greatly reduces training sample complexity. The procedure implements the reward-maximizing agent oracle, has convergence guarantees, and solves RMABs with multiple action types [81,52], the first deep RL procedure to do so. DDLPO also easily extends to more general weakly-coupled MDPs [4,60] and enables computing continuous-action policies, a previously unstudied RMAB direction.
3. We formulate the non-stationary regret-maximizing nature oracle as a multi-agent RL (MARL) problem, a framework of potential general interest in robust planning. We solve this problem in the combinatorially hard RMAB setting by extending DDLPO to include a shared critic and a continuous-action policy network for nature's selection of the uncertain transition dynamics.

5.2 Related Work

RMABs The reward-maximizing, binary-action RMAB problem was introduced by Whittle [163]. His widely used Whittle index policy [103,54,13] is asymptotically optimal under *indexability* [161]. Glazebrook et al. [52] and Hodge & Glazebrook [63] extended the Whittle index to multi-action RMABs with special monotonic structure, while Killian et al. [81] gave a more general Lagrange-based method. Hawkins [60] studied methods for weakly coupled Markov decision processes (WCMDP), which generalize multi-action RMABs to have multiple constraints, and propose Lagrangian solutions for small problems. Adelman & Mersereau [4] and Gocgun & Ghate [55] followed by providing better so-

lutions to WCMDPs but sacrifice scalability. All these works assumed precise knowledge of stochastic dynamics. Some recent works have studied online RMABs with unknown dynamics but all have prohibitively large sample complexity [47,72,18,80]. None consider robust planning under environment uncertainty, which we address.

Our work also relates to learning algorithms for *stochastic* multi-armed bandit (MAB) problems [111,21,88]. However, since stochastic MABs follow a stateless reward process, learning algorithms utilize the fact that the true optimal policy simply selects the top B reward-producing arms each round. Conversely, the arms in restless MABs have reward processes that follow MDPs, so the top B arms to play each round is state- and action-dependent and constantly evolving, making both the learning and the planning problems much more challenging, and which our algorithms address.

RL for RMABs A few recent works learn Whittle indices for indexable binary-action RMABs using (i) deep reinforcement learning (DRL) [117] and (ii) tabular Q-learning [18,46,12]. Killian et al. [80] take tabular Q-learning to the multi-action setting. In contrast, our DRL approach provides a more general solution to binary and multi-action RMAB domains, not requiring indexability or problem structure, and is far more scalable than tabular methods. We are also the first to handle continuous-action RMABs, key to the nature oracle. Also related is the space of combinatorial RL. However, most existing algorithms consider single-shot problems, e.g., traveling salesman [87,79], which lack a notion of future state that is critical to solving any version of RMAB, and none accommodate the general cost/budget structure of multi-action RMAB [147]; our methods address these limitations.

Robust planning Work on robust planning in RL mainly focuses on maximin reward via robust adversarial RL [126] or multi-agent RL (MARL) [90,94], but maximin reward leads to overly conservative policies [120]. The minimax regret criterion [23] avoids this pitfall, but this objective is challenging with very large or continuous strategy spaces. This can be addressed with the DO approach

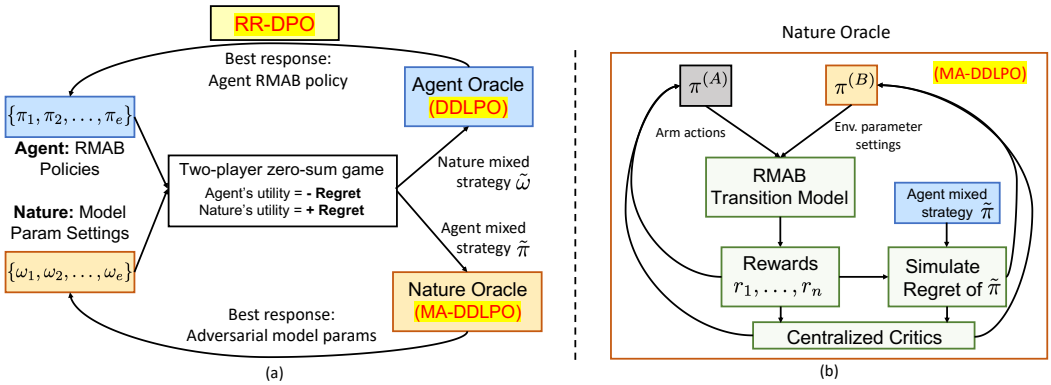


Figure 5.1: (a) Proposed framework for solving the Robust RMAB problem. The main loop follows a DO approach to iteratively compute a minimax regret optimal RMAB policy where each oracle is a novel DRL algorithm for RMABs. **(b)** The nature oracle: a novel multi-agent RL formulation of RMAB, that tackles non-stationarity with a centralized critic.

proposed by McMahan et al. [106] which explores a small subset of strategies while still guaranteeing optimal convergence [51]. Subsequently, DO has been extended to optimize MARL problems with multiple selfish agents [90]. Recently, Xu et al. [170] used DO to solve a single Markov decision process (MDP) minimax-regret planning problem and used RL to implement the oracles. However, when applied to RMABs, the number of outputs in their policy network grows exponentially, as does the size of the state space being learned, both of which require prohibitively long training times beyond trivially sized RMABs. Accordingly, we found that their RL algorithms failed to scale past $N = 5$ arms and $S = 2$ states, whereas we show in Sec. 5.5 that our algorithms solve problems that are orders of magnitude larger. Additionally, their approach is designed only for continuous state/action spaces, whereas our approach can find robust policies for any combination of discrete *or* continuous state/action spaces. We accomplish this via our novel formulation of the nature oracle as a MARL problem, which decomposes the causes of non-stationarity, i.e., agent and nature, and learn them with separate networks.

5.3 Preliminaries

We consider the multi-action RMAB setting with N arms[81,52], which generalizes classical binary-action RMABs[163].* Each arm $n \in [N]$ follows an MDP $(\mathcal{S}_n, \mathcal{A}_n, \mathcal{C}_n, T_n, R_n, \beta)$, where $\mathcal{S}_n$ is a set of finite, discrete states; $\mathcal{A}_n$ is a set of finite, discrete actions; $\mathcal{C}_n : \mathcal{A}_n \to \mathbb{R}$ defines action costs, where $\mathcal{C}_n[0] = 0$ encodes a no-cost "passive action" for all arms; $T_n : \mathcal{S}_n \times \mathcal{A}_n \times \mathcal{S}_n \to [0,1]$ gives the probability of transitioning from one state to another given an action; $R_n : \mathcal{S}_n \to \mathbb{R}$ is a reward function; and $\beta \in [0,1)$ is the discount factor. For ease of exposition, let $\mathcal{S}_n, \mathcal{A}_n, \mathcal{C}_n$, and R_n be the same for all $n \in [N]$, and thus drop the subscript n, though all methods apply to the general case. Let $\boldsymbol{s}$ be an N-length vector of states over all arms and let $\boldsymbol{A} \in \{0,1\}^{N \times |\mathcal{A}|}$ be a decision matrix that one-hot-encodes the action taken on each arm. The planner computes policies π which map states $\boldsymbol{s}$ to actions $\boldsymbol{A}$ with the constraint that the sum cost of actions is less than a budget B in every round $t \in [H]$.

We extend multi-action RMABs to the robust setting in which the exact transition probabilities are unknown. Instead, the transition dynamics T_n of each arm $n \in [N]$ are determined by a set of parameters $\omega_n \in \Omega_n$, each within a given interval uncertainty $\overline{\underline{\omega}}_n := [\underline{\omega}_n, \overline{\omega}_n]$. Let ω be a given parameter setting such that $\omega_n \in \overline{\underline{\omega}}_n$ for all $n \in [N]$. Let $G(\pi, \omega) = \mathbb{E}[\sum_{t=1}^{H} \beta^t \sum_{n \in [N]} R(\boldsymbol{s}_t^n) \mid \pi, \omega]$ be the planner's expected discounted reward under π and ω, where $\boldsymbol{s}_n^t$ is the state of arm n at time t. Then, *regret* is defined:

$$L(\pi, \omega) = G(\pi_\omega^\star, \omega) - G(\pi, \omega) \,, \tag{5.1}$$

where $\pi_\omega^\star$ is the optimal reward-maximizing policy under ω. In our robust setting, our objective is to

*Our approaches also easily extend to weakly-coupled MDPs, which allow multiple budget constraints[60], as well as to continuous-action RMABs, previously unstudied.

compute a policy $\pi^\dagger$ that minimizes the maximum regret L possible for any realization of ω, i.e.:

$$\pi^\dagger = \min_{\pi} \max_{\omega} L(\pi, \omega) \,. \tag{5.2}$$

This problem is computationally expensive to solve since simply computing a policy π that maximizes the reward $G(\pi, \omega)$ is PSPACE-hard[122] even when the T_n are known, i.e., ω is given.

A more tractable approach for computing multi-action RMAB policies π is to utilize the Lagrangian relaxation[60,81], reproduced below. For a given ω, the optimal policy $\pi^\star_\omega$ maximizes the constrained Bellman equation:

$$J(\boldsymbol{s}) = \max_{\boldsymbol{A}^c} \left\{ \sum_{n=1}^{N} R(\boldsymbol{s}_n) + \beta \mathop{\mathbb{E}}_{\omega}[J(\boldsymbol{s}') \mid \boldsymbol{s}, \boldsymbol{A}^c] \right\} \tag{5.3}$$

$$\text{where } \boldsymbol{A}^c \subseteq \boldsymbol{A}$$

$$\text{s.t.} \sum_{n=1}^{N} \sum_{j=1}^{|\mathcal{A}|} \boldsymbol{A}_{nj} c_j \leq B \qquad \sum_{j=1}^{|\mathcal{A}|} \boldsymbol{A}_{nj} = 1 \;\; \forall n \in [N]$$

where $\boldsymbol{A}_{nj} = 1$ if the jth action is taken on arm n (else 0) and $c_j \in \mathcal{C}$ is the jth action cost. We then take the Lagrangian relaxation of the budget constraint[60], giving:

$$J(\boldsymbol{s}, \lambda^\star) = \min_{\lambda} \left(\frac{\lambda B}{1-\beta} + \sum_{n=1}^{N} \max_{j \in |\mathcal{A}|} \{ Q_n(\boldsymbol{s}_n, a_{nj}, \lambda) \} \right) \tag{5.4}$$

$$\text{where } \; Q_n(\boldsymbol{s}_n, a_{nj}, \lambda) = R(\boldsymbol{s}_n) - \lambda c_j + \beta \mathbb{E}_\omega \left[Q_n(\boldsymbol{s}'_n, a_{nj}, \lambda) \mid \pi^{La}_\omega(\lambda) \right] \,. \tag{5.5}$$

Here, a_{nj} is the jth action of arm n, Q is the state-action value function, and $\pi^{La}_\omega(\lambda)$ is the optimal policy for a given λ. The key insight is that this relaxation decouples the value functions of the arms, except for the shared λ, i.e., for a given value of λ, all Q_n could be solved via N individual value iter-

ations. However, finding and setting $\lambda := \lambda^\star$ is critical to finding good policies for multi-action RMABs[81,52], where $\pi_\omega^{La}(\lambda^\star)$ is used to recover a policy that respects the original budget constraint by solving a knapsack with $Q_n(\boldsymbol{s}_n, a_{nj}, \lambda^\star)$ as values, $\mathcal{C}$ as weights, and the constraints of Eq. 5.3, then taking the actions according to the Q_n in the solved knapsack. The knapsack solution finds the combination of actions with the largest sum of learned $Q_n(\boldsymbol{s}_n, a_{nj}, \lambda^\star)$ values which still respects the budget. The integer program for the knapsack is given in Appendix D.2 and has time complexity $\mathcal{O}(N|\mathcal{A}|B)$[81].

5.4 Solving Robust RMABs

We now build our approach for finding robust RMAB policies, visualized in Fig. 5.1(a). We use an iterative DO approach which achieves the minimax regret objective of Eq. 5.2 by casting the optimization problem as a zero-sum game between two players: an *agent* which learns policies π to minimize regret, and an adversarial *nature* which selects environment parameters ω to maximize regret of the agent. In this two-player game, the *pure strategy* space for the agent is the set of all feasible RMAB policies π that respect the budget constraint. The pure strategy space for nature is a continuous, closed set of parameters ω within the given uncertainty intervals. The algorithm maintains a pure strategy set for the agent and nature (Fig. 5.1(a) left boxes); each iteration, these strategy sets are used to compute a *mixed strategy*—i.e., a probability distribution over pure strategies—Nash equilibrium in a regret game (Fig. 5.1(a) center). Each oracle then learns a best response against the opponent's mixed strategy to add to its strategy set (Fig. 5.1(a) right boxes).

The agent oracle's goal is to find an RMAB policy π, or pure strategy, that minimizes regret (Eq. 5.1) given a nature mixed strategy $\tilde{\omega}$. That is, the agent minimizes $L(\pi, \tilde{\omega})$ w.r.t. π, while $\tilde{\omega}$ is constant. Recall from Eq. 5.1 that $L(\pi, \tilde{\omega}) = G(\pi^\star_{\tilde{\omega}}, \tilde{\omega}) - G(\pi, \tilde{\omega})$. Since $\tilde{\omega}$ and $\pi^\star_{\tilde{\omega}}$ are constant, then the first term $G(\pi^\star_{\tilde{\omega}}, \tilde{\omega})$ is also constant. Thus minimizing $L(\pi, \tilde{\omega})$ is equivalent to maximizing

the second term $G(\pi, \tilde{\omega})$, which is maximal at $\pi = \pi^\star_{\tilde{\omega}}$. In other words, the agent oracle must compute an optimal reward-maximizing policy w.r.t. $\tilde{\omega}$. Such a reward-maximizing objective aligns with existing RL techniques, but still requires that we address the challenge of learning in the combinatorial state and action spaces of the RMAB. To address this challenge, *we propose a new RL method which decomposes the RMAB into N per-arm learning problems and a complementary λ-network learning problem*, which together learn to spend limited budget where it will give the best return, detailed in Sec. 5.4.1.

Conversely, the nature oracle seeks to find a parameter setting ω, or pure strategy, that maximizes the agent's regret given a mixed strategy $\tilde{\pi}$, i.e., maximize $L(\tilde{\pi}, \omega)$ with respect to ω, while $\tilde{\pi}$ is fixed. This objective is even more challenging because both $G(\pi^\star_\omega, \omega)$ and $G(\tilde{\pi}, \omega)$ are functions of ω. Most critically, computing $G(\pi^\star_\omega, \omega)$ requires obtaining an optimal policy $\pi^\star_\omega$ as ω changes in the optimization—this amounts to a planning problem in which an agent must learn an optimal policy while the environment changes, controlled by ω, making the nature oracle difficult to solve. Moreover, in the interval uncertainty setting we consider, ω is defined by a space of continuous values; thus nature's pure strategy space is infinite, making the problem even more complex, since it cannot be exhaustively searched.

To tackle this complexity we propose a novel method for implementing the regret-maximizing nature oracle by casting it as an MARL problem. The approach, visualized in Fig. 5.1(b), trains one auxiliary agent to solve for a policy $\pi^\star_\omega$ (π^A in Fig. 5.1(b)), needed to compute $G(\pi^\star_\omega, \omega)$ in the regret term, and simultaneously trains a second agent to learn worst-case parameters ω (π^B in Fig. 5.1(b)) that minimize $G(\tilde{\pi}, \omega)$—together, these will maximize the regret $L(\tilde{\pi}, \omega)$. With this MARL setup, we mitigate nonstationarity through centralized critic networks which allow each agent to include the other's actions in their learned state space. Solving a MARL problem requires an RL algorithm to optimize the underlying policy, so we first introduce our novel RL approach, DDLPO, to solve RMABs (Sec. 5.4.1) as a part of our agent oracle and then use the algorithm as the backbone of our

Algorithm 5.4.1 DDLPO

Input: Initial state $\boldsymbol{s}_0$, nature mixed strategy $\tilde{\omega}$,
`n_epochs`, `n_subepochs`, `n_steps`

Init. policy networks θ_n for each arm $n \in [N]$
Init. critic networks φ_n for each arm $n \in [N]$
Init. λ-network Λ
Init. `buff` = [] and $\boldsymbol{s} = \boldsymbol{s}_0$
for *epoch* $= 1, 2, \ldots,$ `n_epochs` **do**
  Sample $\lambda = \Lambda(\boldsymbol{s})$
  Sample $\omega \sim \tilde{\omega}$
  for *subepoch* $= 1, \ldots,$ `n_subepochs` **do**
    for timestep $t = 1, \ldots,$ `n_steps` **do**
      Sample actions $a_n \sim \theta_n(s_n, \lambda) \quad \forall n \in [N]$
      $\boldsymbol{s}', \boldsymbol{r} =$ `Simulate`$(\boldsymbol{s}, \boldsymbol{a}, \omega)$
      Add tuple $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{s}', \lambda)$ to `buff`
      $\boldsymbol{s} = \boldsymbol{s}'$
    Update each (θ_n, φ_n) pair via PPO, using trajectories in `buff`
  Update Λ via Prop. 5.4.1 with costs of final subepoch
return $\theta_1, \ldots, \theta_N, \varphi_1, \ldots, \varphi_N$ and Λ

nature oracle (Sec. 5.4.2).

5.4.1 Agent Oracle: Deep RL for RMAB

Existing DRL approaches can be applied to the objective in Eq. 5.3, but, as detailed in Sec. 5.2, they fail to scale past trivially sized RMAB problems since the action and state spaces grow exponentially in N. To overcome this, we develop a novel DRL algorithm that instead solves the decoupled problem (Eq. 5.4). The key benefit of decoupling is to render policies and Q values of each arm independent, allowing us to learn N independent networks with *linearly sized state and action spaces, relieving the combinatorial burden of the learning problem*. However, this decoupling approach introduces a new technical challenge in solving the dual objective which maximizes over policies but

Algorithm 5.4.2 DDLPO-Act

Input: State $\boldsymbol{s}$, costs $\mathcal{C}$, budget B, agent actor, critic, and λ networks $\theta_1, \ldots, \theta_N$, $\varphi_1, \ldots, \varphi_N$, Λ, selection method α

$\lambda = \Lambda(\boldsymbol{s})$
if α == 'GreedyProba' **then**
  $p_n = \theta_n(s_n, \lambda) \quad \forall n \in [N]$ {Action distr. of arm n}
  $\boldsymbol{a}$ = `GreedyProba`$(\boldsymbol{p}, \mathcal{C}, B)$ {Greedily select highest probability actions until budget B is reached}
else if α == 'QKnapsack' **then**
  $q_{nj} = \varphi_n(s_n, a_{nj}, \lambda) \quad \forall n \in [N], \forall j \in [|\mathcal{A}|]$
  $\boldsymbol{a}$ = `QKnapsack`$(\boldsymbol{q}, \mathcal{C}, B)$ {Solve knapsack in Appendix D.2}
else if α == 'Whittle' **then** {Binary action only}
  $\boldsymbol{a} = \textsc{BinaSearch}(\boldsymbol{s}, B, \varphi_1, \ldots, \varphi_N)$ {Appendix D.2}
return $\boldsymbol{a}$

minimizes over λ, as discussed in Sec. 5.3.

To solve this, we derive a dual gradient update procedure that iteratively optimizes each objective as follows: (1) holding λ constant, learn N independent policy networks via policy gradient, augmenting the state space to include λ as input, as in Eq. 5.4; (2) use sampled trajectories from those learned policies as an estimate to update λ towards its minimizing value via a novel gradient update rule. Another challenge is that $\lambda^\star$ of Eq. 5.4 depends on the current state of each arm—therefore, a key element of our approach is to learn this function $\lambda^\star(\boldsymbol{s})$ concurrently with our iterative optimization, using a neural network we call the λ-network that is parameterized by Λ. To train the λ-network, we use the following gradient update rule.

Proposition 5.4.1. *To learn the value λ that minimizes Eq. 5.4 given a state $\boldsymbol{s}$, the λ-network, parameterized by Λ, should be updated with the following gradient rule:*

$$\Lambda_t = \Lambda_{t-1} - \alpha \left(\frac{B}{1-\beta} + \sum_{n=1}^{N} D_n(s_n, \lambda_{t-1}(\boldsymbol{s})) \right) \tag{5.6}$$

where α is the learning rate and $D_n(s_n, \lambda)$ is the negative of the expected β-discounted sum of action costs for arm n starting at state s_n under the optimal policy for arm n for a given value of λ.

As D_n lacks a closed form, the key insight we make is that it can be estimated by sampling multiple rollouts of the policy networks of all arms during training. As long as arm policies are trained for adequate time on the given value of λ, the gradient estimate will be accurate, i.e., $D_n(s_n, \lambda_{t-1}(\boldsymbol{s})) \approx -\sum_{k=0}^{K-1} \beta^k c_n^k$ where K is the number of samples collected in an epoch and c_n^k is the action cost of arm n in round k. Moreover, this procedure will converge to the optimal parameters $\Lambda^\star$ if the arm policies are optimal.

Proposition 5.4.2. *Given arm policies corresponding to optimal Q-functions, Prop. 5.4.1 will lead Λ to converge to the optimal as the number of training epochs and $K \to \infty$.*

Proofs are given in Appendix D.1. One interesting feature of this update rule is that to collect samples that reflect the proper gradient, the RMAB budget must not be imposed *at training time*—rather, the policy networks and λ-network must be allowed to learn to play the Lagrange policy of Eq. 5.4, which learns to spend the correct budget in expectation, via our iterative update procedure. Therefore, at training time, we sample actions randomly according to the actor network distributions, without imposing the budget constraint. However, *at test time, we always take actions in a way that respects the budget constraint* as described in Alg. 5.4.2. Alg. 5.4.2 chooses actions either by (1) selecting greedily by the probabilities of the arm actor networks (2) using the learned $Q(\lambda)$-functions of the arm critic networks to follow the Q-value-maximizing knapsack procedure (Appendix D.2), or (3) in binary-action settings, using the $Q(\lambda)$-functions to follow a binary search procedure such that selected actions are equivalent to the Whittle index policy (Appendix D.2).

In theory, the policy networks could be trained via any DRL procedure that ensures the above characteristics for training the λ-network. In practice, we train with proximal policy optimization

(PPO)[141], a state-of-the-art policy gradient approach. Importantly, PPO is also flexible enough to handle both discrete and continuous actions which is necessary for the nature oracle.

Finally, to enable our iterative, dual-update procedure in practice, we need a mechanism to both (1) explore new arm policy actions after an update to Λ, then (2) exploit learned policy actions to develop good gradient estimates for Λ. We navigate this important trade-off by adding an entropy regularization term to the policy networks losses, controlled via a cyclical temperature parameter. We call our algorithm Deep Distributed Lagrange Policy Optimization (DDLPO), provide pseudocode in Algorithm 5.4.1, and include more implementation details in Appendix D.4.

5.4.2 Nature Oracle: Multi-Agent RL

Armed with a DRL procedure for learning RMAB policies, we now develop the MARL procedure, which we call MA-DDLPO, to implement the nature oracle. Recall that the challenge of the nature oracle is to jointly optimize a policy $\pi^\star_\omega$ and environment parameters ω. We propose to solve this optimization using MARL, designed to handle this form of non-stationarity[99] via centralized critics. In our MARL setup, each of two "players" (i.e., the "multiple agents") will aim to compute $\pi^\star_\omega$ and ω, respectively, with separate objectives. The procedure is visualized in Fig. 5.1(b).

To implement the MARL nature oracle, we introduce two new players A and B. Player A is an *auxiliary player* whose goal is to optimize the RMAB policy $\pi^\star_\omega$ given a changing ω, i.e., the first term of regret (Eq. 5.1). We call A auxiliary because its learned policy will never be used outside the nature oracle; A is only used to assist the nature oracle in computing the regret associated with a given ω. Alternatively player B is an adversarial player whose goal is the same as that of the nature oracle itself, i.e., to find parameters ω that maximize regret of the current agent mixed strategy $\tilde{\pi}$. We define a shared transition function for the environment in which the players act $T : \mathcal{S} \times \mathcal{A}_A \times \mathcal{A}_B \to \mathcal{S}$. Here, $\mathcal{A}_A$ is the action space of the underlying multi-action RMAB. At a given state $\boldsymbol{s}$, the action space $\mathcal{A}_B$ defines for player B actions ω which, in general, depend on $\boldsymbol{s}$. That is, at each

Algorithm 5.4.3 MA-DDLPO

Input: Agent mixed strategy $\tilde{\pi}$, `n_epochs`, `n_subepochs`, `n_steps`, `n_sims`

Init. player A: arm policy networks $\theta_n^{(A)}$ and arm critic networks $\varphi_n^{(A)}$ $\forall n \in [N]$, and λ-network Λ
Init. player B: environment parameter policy network $\theta^{(B)}$, critic network $\varphi^{(B)}$
Init. `buff` = []
for *epoch* $= 1, 2, \ldots,$ `n_epochs` **do**
  Sample $\boldsymbol{s}$ at random
  Sample $\lambda = \Lambda(\boldsymbol{s})$
  for *subepoch* $= 1, \ldots,$ `n_subepochs` **do**
    for $t = 1, \ldots,$ `n_steps` **do**
      Sample $a_n^{(A)} \sim \theta_n^{(A)}(s_n, \lambda)$ for each $n \in [N]$
      Sample $\omega^{(B)} \sim \theta^{(B)}(\boldsymbol{s})$
      $\boldsymbol{r}^{(A)}, \boldsymbol{s}' = \textsc{simulate}(\boldsymbol{s}, \boldsymbol{a}^{(A)}, \omega^{(B)})$
      $\tilde{r} = \textsc{simulate}(\boldsymbol{s}, \tilde{\pi}(\boldsymbol{s}), \omega^{(B)}, \texttt{n\_sims})$ {(mean of `n_sims` 1-step rollouts of $\tilde{\pi}$)}
      $r^{(B)} = \left(\sum_{n \in [N]} r_n^{(A)}\right) - \tilde{r}$ {(regret of $\tilde{\pi}$)}
      Add $(\boldsymbol{s}, \boldsymbol{a}^{(A)}, \omega^{(B)}, \boldsymbol{r}^{(A)}, r^{(B)}, \boldsymbol{s}', \lambda)$ to `buff`
      $\boldsymbol{s} = \boldsymbol{s}'$
    Update each $(\theta_n^{(A)}, \varphi_n^{(A)})$ pair using trajectories in `buff`. $\varphi_n^{(A)}$ get $\omega^{(B)}$ as part of state
  Update Λ via Prop. 5.4.1 with costs of final subepoch
  Update $\theta^{(B)}, \varphi^{(B)}$ using trajectories in `buff`. $\varphi^{(B)}$ gets $\boldsymbol{a}^{(A)}$ as part of state
return $\theta^{(B)}$

step, player B selects environment parameters ω, and thus transition probabilities that will influence the outcome of player A's actions. We adopt the centralized critic idea from multi-agent PPO [174] to our RMAB setting to create MA-DDLPO. A notable strength of our MARL approach is that it allows the discrete-space policy of player A and the continuous-space policy of player B to be learned by separate networks, simplifying training compared to an alternative combined-network approach. Moreover, our choice to use PPO offers a convenient way to learn both types of policies as separate networks, while utilizing a single framework of update rules.

A critical step is then to define the rewards for players A and B to match their objectives. Since player A's objective is to find $\pi^\star_\omega$, it adopts the reward defined by the underlying RMAB, that is, $R^{(A)}(\boldsymbol{s}) = \sum_{n=1}^{N} R_n(\boldsymbol{s})$. However, player B's objective is to learn the regret-maximizing parameters ω. This objective is challenging because it requires computing and optimizing over the returns of the fixed input policy $\tilde{\pi}$ with respect to all possible ω, which is in general non-convex. In practice, to estimate the returns of $\tilde{\pi}_\omega$, we execute a series of roll-outs against player B's current action. That is, given $\boldsymbol{s}$ at a given round, we sample an action from $\tilde{\pi}_\omega$ and the next state $\boldsymbol{s'}$, and define the *regret-based* reward of player B, as $R^{(B)} = \sum_{n=1}^{N} R_n(s_n) - \frac{1}{Y}\sum_{y=1}^{Y} r_y^{\tilde{\pi},\omega}$, where $r_y^{\tilde{\pi},\omega}$ is the reward from each of Y one-step Monte Carlo simulations of the mixed strategy $\tilde{\pi}$ in ω.

To train the policies, player A has the same policy network architecture as DDLPO, i.e., N discrete policy networks and one λ-network, and the player B actor network is a single continuous-action policy network. Since players A and B have separate reward functions, they have their own critic networks, but these critics are *centralized* in that they both take the actions of the other as input. Other than the centralized critic, player A is trained the same way as DDLPO, and player B is trained in a standard PPO fashion. In practice, to ensure good gradient estimates for player A's λ-network in MA-DDLPO, we keep player B's network—and thus the environment—constant between Λ updates, updating B's network with the same frequency as the λ-network updates. Pseudocode for MA-DDLPO is given in Alg. 5.4.3 and further details of its implementation are given in Appendix D.4.

5.4.3 Minimax Regret RMAB Double Oracle

We now have all the pieces needed to run our robust algorithm, Robust RMABs via Deep Policy Oracles (RR-DPO), visualized in Fig. 5.1(a), with pseudocode presented in Algorithm 5.4.4, adapted from the MIRROR framework [170]. We use DDLPO to instantiate the agent oracle, MA-DDLPO for the nature oracle, and run RR-DPO until the improvement in value for each player is

Algorithm 5.4.4 RR-DPO

Input: Environment simulator and parameter uncertainty intervals $\overline{\underline{\omega}}_n$ for all $n \in [N]$
Parameters: Convergence threshold ε
Output: Agent mixed strategy $\tilde{\pi}$

```
Ω_0 = {ω_0}, with ω_0 selected at random
Π_0 = {π_{B_1}, π_{B_2}, ...}, where π_{B_i} are baseline and heuristic strategies
for epoch e = 1, 2, ... do
    Solve for (π̃_e, ω̃_e), mixed Nash equilibrium of regret game with strategy sets Ω_{e−1}
    and Π_{e−1}
    π_e = DDLPO(ω̃_e)
    ω_e = MA-DDLPO(π̃_e)
    Ω_e = Ω_{e−1} ∪ {ω_e}, Π_e = Π_{e−1} ∪ {π_e}
    if L(π̃_e, ω_e) − L(π̃_{e−1}, ω̃_{e−1}) ≤ ε and L(π_e, ω̃_e) − L(π̃_{e−1}, ω̃_{e−1}) ≤ ε then
        break
return π̃_e
```

within a tolerance ε or until a set number of iterations.

We now establish conditions under which RR-DPO converges to the minimax regret–optimal policy in finite iterations. In the binary-action setting, assuming each oracle returns true best responses, and under an analytical condition that is straightforward to achieve, i.e., finite pure strategy sets:[†]

Proposition 5.4.3. *RR-DPO converges in a finite number of steps to the minimax regret-optimal policy.*

In addition, we empirically verify that good policies are found outside of these conditions, and that RR-DPO converges using our continuous-strategy-space nature oracle. Further, we show that a policy that maximizes reward assuming a fixed parameter set can incur arbitrarily large regret when the parameters are changed (proofs in Appendix D.1).

[†]Straightforward to achieve for nature oracle via discretization.

Proposition 5.4.4. *In the Robust RMAB problem with interval uncertainty, the max regret of a reward-maximizing policy can be arbitrarily large compared to a minimax regret-optimal policy.*

5.5 Experimental Evaluation

We first experimentally demonstrate the importance of robust planning in the presence of uncertainty using a hand-crafted synthetic domain (inspired by Prop. 5.4.4). We then evaluate our algorithm on two challenging real-world-inspired public health planning scenarios which demonstrate the capability of our robust RMAB framework. All experiments use selection method $\alpha =$ 'GreedyProba' for DDLPO-Act (Alg. 5.4.2), which we found had the best performance.

We compare RR-DPO against five baselines. These baselines include three variations of the reward-maximizing approach from Hawkins [60], which, given fixed environment parameters ω, at each step computes a Lagrange policy, then chooses actions following the knapsack procedure described in Sec. 5.3. The three variations are pessimistic (**HP**), mean (**HM**), and optimistic (**HO**), which assume the environment parameters are set at the lower bound, mean, and upper bound of the given intervals for each arm. We also implement **RLvMid**, which *learns* (rather than computes) a policy via DDLPO assuming *mean* parameters, and **Rand**, which acts randomly to fill the budget. All results are averaged over 50 random seeds and 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 3.7.10. Our DDLPO implementation builds on OpenAI Spinning Up [3] and RR-DPO builds on the MIRROR implementation [170], computing Nash equilibria using Nashpy 0.0.21 [86]. Code is available at `https://github.com/killian-34/RobustRMAB` and hyperparameter settings are in Appendix D.4.

5.5.1 Experimental Domains

Synthetic demonstrates that reward-maximizing policies (RLvMid, HP, HM, HO) may incur large regret in the presence of uncertainty. There are three binary-action arm types $\{U, V, W\}$, each with $\mathcal{C} = \{0, 1\}, \mathcal{S} = \{0, 1\}, R(s) = s$, and the following transition matrix, with rows and columns corresponding to actions and next states, respectively:

$$T^n_{s=0} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}, \quad T^n_{s=1} = \begin{bmatrix} 1.0 & 0.0 \\ 1 - p_n & p_n \end{bmatrix}$$

$$p_U \in [0.00, 1.00], p_V \in [0.05, 0.90], p_W \in [0.10, 0.95]$$

When an arm is at $s = 0$, each action has equal impact on the state transition. When the arms are at $s = 1$, selecting arms with high p_n is optimal. This implies that policies can be specified by the order in which arms would be acted on, when they are in state $s = 1$. Accordingly, $\pi_{HP} = [W, V, U]$, $\pi_{HM} = [W, U, V]$, and $\pi_{HO} = [U, W, V]$. However, observe that there exist values of p_n that can make each of the reward-maximizing policies incur large regret, e.g., for π_{HO} $p_U = 0.0, p_V = 0.9, p_W = 0.1$ would induce an optimal policy $[V, W, U]$, which is the reverse of π_{HO}.

ARMMAN is a real-world *maternal healthcare intervention problem* modeled as a binary-action RMAB [18]. The goal is to select a subset of mothers each week to intervene on to encourage engagement with automated maternal health messaging. The behavior of enrolled women is modeled by an MDP with three states: Self-motivated, Persuadable, and Lost Cause. We use the summary statistics given in their paper and assume uncertainty intervals of 0.5 centered around the transition parameters, resulting in 6 uncertain parameters per arm (details in Appendix D.3.1). Similar to the setup by Biswas et al. [18], we assume 1:1:3 split of arms with high, medium, and low probability of increasing their engagement upon intervention. In our experiments, we scale the value of N in mul-

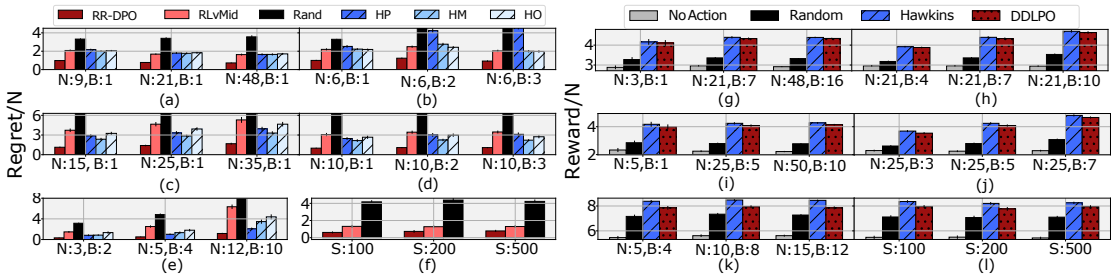


Figure 5.2: (a–f) Maximum policy regret of RR-DPO in robust setting for Synthetic (a,b), ARMMAN (c,d) and SIS (e,f) domains. Lower is better. Synthetic is scaled by 3 and ARMMAN by 5 to maintain the distributions of arm types specified in Sec. 5.5. (e) uses $S = 50$ and (f) uses $N = 5, B = 4$. RR-DPO beats all baselines by a large margin across various settings. **(g–l)** Returns of DDLPO for reward-maximizing setting (agent oracle) for synthetic (g,h), ARMMAN (i,j), and SIS (k,l) domains. Higher is better. (k) uses $S = 50$ and (l) uses $N = 5, B = 4$. DDLPO is competitive across parameter settings.

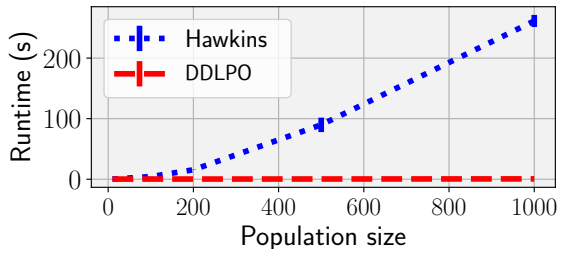


Figure 5.3: The poor scaling of query time of the Hawkins baseline compared to DDLPO, as discussed in Sec. 5.5, even for relatively small problem sizes ($N = 10, B = 2$).

tiples of 5 to keep the same split of arm categories of 1:1:3.

SIS Epidemic Model is a discrete-state model in which arms represent distinct geographic regions and each member of an arm's population of size N_p is either (**S**)usceptible to or (**I**)nfected with an infectious disease. Such models have been the subject of increased interest following the COVID-19 pandemic[62,78], and will represent a large-state and multi-action experimental domain. In our model, the count of **S** members of the population is the state of each arm. Each arm's SIS model is defined by parameters κ, the average number of contacts per round, and r_{infect}, the probability of infection given contact with an **I** member. Details on computing discrete state transition probabilities from these parameters are derived from Yaesoubi & Cohen [171] and given in Appendix D.3.2. We introduce three intervention actions $\{a_0, a_1, a_2\}$ with costs $c = \{0, 1, 2\}$. Action a_0

represents no action, a_1 represents messaging about physical distancing (divides κ by a_1^{eff}), and a_2 represents distributing face masks (divides r_{infect} by a_2^{eff}). We impose the following uncertainty intervals: $\kappa \in [1, 10]$, $r_{infect} \in [0.5, 0.99]$, $a_{\{1,2\}}^{eff} \in [1, 10]$.

5.5.2 Performance of RR-DPO

First, we evaluate the performance of the algorithms in uncertain environments. We compute the regret of an agent's pure strategy π against a nature pure strategy ω as the difference in the average reward obtained by π against ω and the average reward of the best strategy in the experiment against ω. The average reward is the discounted sum of rewards over all arms for a horizon of length 10, over 25 simulations. In each setting, DO runs for 6 epochs, using 100 rollout steps and 100 training epochs for each oracle. After completion, each baseline strategy is evaluated by querying the nature oracle for the best response against that strategy, then computing max regret against all ω. The regret of RR-DPO is computed as the utility of the agent mixed strategy returned by the DO over the two-player regret game.

Fig. 5.2(a–f) shows RR-DPO incurs the lowest regret, beating the baselines in all domains. (a,b) shows results on the synthetic domain, demonstrating our approach can reduce regret by ~50% against the benchmarks, across various values of N and B. Moreover, as B increases, the regret incurred may increase, since higher budget implies better reward potential for the optimal policy; however, the regret for RR-DPO remains small even as B grows. Similarly, for the ARMMAN domain (c,d), a challenging domain adapted from a real-world problem, our algorithm performs consistently better than the baselines, achieving regret that is around 50% lower than the best baselines. In the SIS domain (e–f), another real-world planning setting with a larger state space and multiple actions, our results are robust across parameter settings. Importantly, this holds even as we increase the state space from $S = 100$ to 500 (f), in which running the Hawkins baseline becomes prohibitively expensive.

Finally, we run sensitivity analyses of the algorithms against H and the size of the uncertainty sets (Appendix Fig. D.1). When H varies from 10 to 100, RR-DPO maintains very low regret, while competitor regret as much as doubles, increasing RR-DPO's relative improvement as high as ~60%. Similar results are obtained when varying the uncertainty intervals between 0.25, 0.5 and 1.0 times their widths from the experiments in Fig. 5.2, with RR-DPO always dominating.

5.5.3 Performance of DDLPO

We also evaluate the performance of DDLPO, our novel DRL approach to find reward-maximizing policies for multi-action RMABs, which implements our agent oracle. We compare against **No Action** and **Random** baselines as well as the computationally intensive solution by Hawkins which computes the Lagrange policy, but which requires exact environment parameters and discrete states/actions. Hawkins upper bounds DDLPO for small discrete problems since it is exact whereas DDLPO learns the Lagrange policy from samples. Each experiment is a traditional reward-maximizing RMAB instantiated with a random sample of valid parameter settings for each seed.

Fig. 5.2(g–l) shows DDLPO achieves reward comparable to the Hawkins algorithm and significantly better than random, providing insight into the success of our RR-DPO approach which DDLPO enables, and showing promise for DDLPO as an algorithm of general interest. In the synthetic domain (g,h), DDLPO learns to act on the 33% of arms who belong to category W. The mean reward of DDLPO almost matches that of Hawkins algorithm as N scales with a commensurate budget (g). As we fix N and vary the budget (h), the optimal policy accumulates more reward, and DDLPO almost equals the optimal. We observe similar results on the ARMMAN domain (i,j), a challenging real-world health problem. On the SIS domain (k,l), the strong performance of DDLPO holds in a multi-action setting even as we increase the number of states from 50 to 500 (l).

Moreover, DDLPO beats Hawkins computationally: in Fig. 5.3, a single rollout (10 rounds) of Hawkins takes ~100 seconds when there are 500 states, scaling quadratically in general. This

demonstrates that it would be prohibitive to run Hawkins in the loop of RR-DPO, since agent policies are evaluated thousands of times to compute the regret matrices. For just 25 simulations, computation would take ~42 minutes to evaluate a single cell in the regret matrix, which has $|\Pi| \times |\Omega|$ total cells.

5.6 Conclusion

We address a key limitation blocking RMABs from many real-world settings: that arm dynamics are not known precisely. To plan safe, effective policies, robust approaches accounting for uncertainty are essential, which we give in RR-DPO, enabled by DDLPO, a novel deep-RL algorithm for RMABs of general interest. We hope our contributions bring us closer to deploying RMABs for real-world impact.

6

Robust Planning over Restless Groups: Engagement Interventions for a Large-Scale Maternal Telehealth Program

6.1 Introduction

Maternal mortality, the death of a mother* during pregnancy or within 42 days after childbirth, is an ongoing global health crisis. In India, the maternal mortality rate is particularly stark, estimated between 99 and 130 deaths per 100K births in 2020 [107,50], significantly higher than Sustainable Development Goal 3.1 target of 70 per 100K births [153]. Tragically, most maternal deaths are preventable [164], but lack of finances and awareness prevent mothers from seeking care, particularly in low-income communities [25].

To improve maternal health outcomes, we work with ARMMAN, an India-based non-profit that provides free preventive care to millions of mothers by sending automated health voice messages, specifically targeted towards low-income communities (similar to MAMA [70]). Mothers enrolled in the program receive weekly automated voice messages during pregnancy and up to one year after childbirth. Randomized control trials showed that ARMMAN's messaging program significantly improves key indicators including treatment-seeking during complications, infant breast-

*We recognize that the term "mother" is imperfect, most notably by not reflecting transgender and non-binary identities. We highlight alternative language with discussion in Appendix E.1.

Figure 6.1: Mothers enrolled with ARMMAN receive life-saving preventative care information via voice messages throughout their pregnancy, childbirth, and neonatal period. Photo courtesy of ARMMAN.

feeding, and post-infancy weight[115]. However, ARMMAN found that nearly 38% of mothers disengage, missing critical health information. To improve engagement, ARMMAN employs health workers to provide service calls, but there are only tens of health workers compared to hundreds of thousands of mothers in a given service area — so interventions must be carefully targeted to maximize engagement.

Working with ARMMAN, we model this resource-limited intervention planning problem as a restless multi-armed bandit (RMAB), where each mother (arm) changes their weekly engagement (state) according to a stochastic Markov decision process. RMABs are PSPACE-hard to solve exactly[122] and even the more tractable, asymptotically optimal "Whittle index policy"[163] is challenging to compute at scale.

To improve the scalability of real-world RMAB planning, Mate et al.[104] proposed to organize arms into a small number of *groups*, infer transition dynamics from each group's data, then compute the Whittle index policy per group. While the scalability of their method is desirable for ARMMAN's problem setting, it ignores a key reality of *model uncertainty*: learning transition probabilities from historical data leads to imprecise and imperfect estimates which must be accounted for in planning. Computing RMAB policies that are robust to model uncertainty has only recently been studied. Existing methods achieve robustness to *interval uncertainty* over model dynamics by planning against a model-controlling "nature" adversary to yield policies that minimize max regret[83,170].

Robustness is desirable for ARMMAN's setting, but these methods require training deep reinforcement learning (RL) agents for each arm, so unfortunately do not scale past hundreds of arms.

To enable large-scale, robust intervention planning for ARMMAN, we bridge the gaps in previous works by introducing *robust grouped RMAB*. Our model achieves scalability by considering a grouped-arm paradigm and optimizing for minimax regret over the uncertain model dynamics per group. Unfortunately, the grouping abstraction breaks key assumptions used in previous robust RMAB work: that (1) policies improve by collecting samples of regret by evolving a joint state of all arms, and (2) the nature adversary controls the transitions of each arm individually. We overcome (1) by *decomposing regret per arm*, freeing the planner from relying on a cumbersome joint state to enable efficient group-abstracted planning. For (2), we prove that *restricting the adversary to control dynamics only over groups does not change the equilibrium strategy*, allowing us to leverage the scalable robust grouped model to find policies over hundreds of thousands of arms without sacrificing quality.

Our contributions are as follows. *First,* we introduce *robust grouped RMABs* with a minimax regret objective and propose a solution that employs the double oracle framework [106]. The approach we propose is **GROUPS**: Group RMAB Oracles for Uncertainty-robust Planning at Scale. *Second,* we develop novel methods designed for robust grouped RMABs to implement the two oracles, the planner and adversary. Planning over groups of arms allows large scale-up but presents several new algorithmic challenges as we detail above. *Third,* we prove that the minimax regret–optimal strategy is the same whether the planner and adversary play at the individual or group level. *Our proof enables massive scale-up as it is now sufficient to compute robust strategies over groups*, instead of over individual arms. *Finally,* we demonstrate empirically on real data that **GROUPS reduces worst-case regret up to 50% compared to baselines, representing potentially thousands of additional engagements with life-saving information.** We are working with ARMMAN to deploy GROUPS to positively impact maternal health.

6.2 Related Work

Mobile-based maternal health services are effective and affordable in low- and middle-income communities [160,150]. Successful programs include MatHealth in Uganda [116], Aponjon in Bangladesh [8], ARMMAN in India [115], and text4baby in the United States [44]. Our work is designed to support such programs.

Whittle [163] introduced RMABs and proposed the *Whittle index policy*, which computes indices estimating each arm's "return on investment" then acts on arms with the top K. Weber & Weiss [161] showed this policy is asymptotically optimal under a technical condition. Many RMAB studies assume known transition dynamics, although some recent works design methods to learn policies online [157,117,18,80,156]. However, these online approaches require collecting a prohibitively large number of samples, limiting their real-world applicability in scenarios where the time horizon is short.

Most robust planning papers consider single-MDP (one arm) settings [126,90,94], rather than the budget-coupled N-MDP setting of RMAB. Even for single MDPs, optimizing criteria such as minimax regret [23] requires searching massive strategy spaces; double oracle [106] is one approach to do so efficiently. Recent work combines double oracle with deep RL to solve for minimax regret–optimal robust policies for single MDPs [170]. Killian et al. [83] extended the idea to solve larger RMABs. Both Xu et al. [170] and Killian et al. [83] use deep RL which, if applied to a group setting, would need to explicitly account for the size of each group and state of each arm within each group, limiting their methods' ability to scale beyond hundreds of arms. For the large problem size that ARMMAN faces, our methods must scale to hundreds of thousands of arms.

Finally, robust planning for *stochastic* bandits is well studied [102,65] However, stochastic bandits are stateless and lack passive rewards, and so are not expressive enough to model ARMMAN's setting.

6.3 Model

We consider *grouped RMABs* where N arms (enrolled mothers) comprise M groups. Each arm $n \in [N]$ follows an MDP $\langle \mathcal{S}, \mathcal{A}, P^n, r, \gamma \rangle$ where $s \in \mathcal{S} := \{0, 1\}$ is the state space indicating whether a mother is engaging ($s_n = 1$) or not engaging ($s_n = 0$) with automated voice messages; $r(s) = s$ is the reward function; $a \in \mathcal{A} := \{0, 1\}$ is the action space, i.e., {not intervene, intervene}; $P^n(s, a, s')$ is the probability that arm n transitions from state s to s' given action a; $\gamma \in [0, 1]$ is the discount factor. Let $\boldsymbol{s} \in \mathcal{S}^N$ and $\boldsymbol{a} \in \mathcal{A}^N$ be the combined state and action vectors of all arms. At each timestep t, the task is to choose K mothers to intervene on (deliver service calls to) given the state $\boldsymbol{s}_t$ at time t.

Formally, we compute RMAB policies $\pi : \mathcal{S}^N \to \mathcal{A}^N$ that respect a budget constraint $\|\pi(\boldsymbol{s}_t)\|_1 = K$ for all t. For a given policy π and a fixed environment $P := \{P^n\}_{n \in [N]}$ representing a matrix of transition probabilities of all arms, the average discounted reward is $G(\pi, P) := \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s}_t) \mid \pi, P]$. Given P, the optimal policy which maximizes reward is $\pi_P^\star := \max_\pi G(\pi, P)$. An asymptotically optimal RMAB policy is the Whittle index policy (WIP), which computes the Whittle index $W_P^n(s)$ for each arm n and state s, then intervenes on the arms with the greatest K indices. The Whittle index represents "return on investment," interpreted as a charge for acting that makes *no intervention* equally valuable as *intervention* in the long term. Let $Q_P^n(s, a, \lambda) = r(s) - \lambda a + \gamma \mathbb{E}_{s' \in \mathcal{S}}[\max_{a' \in \mathcal{A}} Q_P^n(s', a', \lambda)]$ be the long-term expected value of action a on arm n in state s. Then, for a given P, the Whittle index for arm n at state s is $W_P^n(s) = \min\{\lambda : Q_P^n(s, 1, \lambda) = Q_P^n(s, 0, \lambda)\}$.

Grouped RMAB For scalability, we organize arms into groups, extending the concept from Mate et al. [104] to our more challenging *robust* setting, e.g., by clustering based on historical engagement patterns. We then estimate uncertainty intervals over transition probabilities per group. However, note that our robust policy computation steps in Section 6.4 are agnostic to the par-

ticular grouping and interval estimation methods. Let $\varphi : [N] \to [M]$ be a surjective mapping of arms to groups and $\varphi^{-1}(m)$ be the set of arms in group m. The uncertainty intervals are $\overline{P}^m_{s,a,s'} := [\underline{P}^m_{s,a,s'}, \overline{P}^m_{s,a,s'}]$ for all m, s, a, s'. Then let $\overline{P}^m := \{\overline{P}^m_{s,a,s'}\}_{s,a,s'}$ be the interval uncertainty matrix for group m across all states and actions. Importantly, though arms in the same group have the same uncertainty intervals, they may not have the same instantiated probabilities within those intervals.

Minimax regret We define regret for grouped RMAB as:

$$R(\pi, P) := G(\pi^\star_P, P) - G(\pi, P) \,, \tag{6.1}$$

where P instantiates $P^m \in \overline{P}^m$ for all groups $m \in [M]$. Our objective is to learn a policy π that minimizes max regret:

$$\min_\pi \max_P R(\pi, P) \,. \tag{6.2}$$

We choose minimax regret as our robust objective since it does not require probability distributions over the uncertainty intervals[23]. Such distributional information is scarce in our setting where $K \ll N$, giving us few samples of transitions for action $a = 1$.

6.4 Methodology

We introduce GROUPS (Group RMAB Oracles for Uncertainty-robust Planning at Scale), a four-step approach visualized end-to-end in Fig. 6.2. Step (3) is our key algorithmic contribution. In step (1), similar arms (mothers) are mapped into groups. In step (2), we combine data from arms in each group with historical engagement data, using bootstrapping to estimate uncertainty intervals $\overline{P}^m$ for each group[140]. In step (3), we compute a minimax regret–optimal policy over groups, where arms in a given group are treated as having the same transition probabilities, greatly improving com-

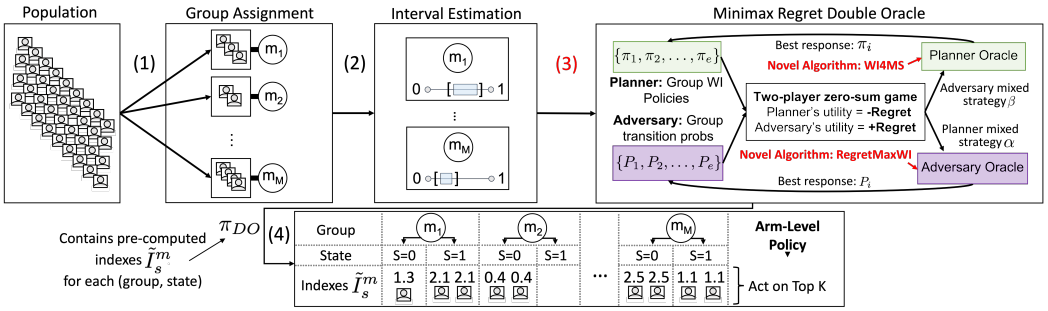


Figure 6.2: GROUPS pipeline for robust grouped RMABs. (1) Assign enrolled mothers (arms) to groups. (2) Estimate uncertainty intervals over transition probabilities. *(3) Novelty of this work:* Compute robust minimax regret–optimal policy via double oracle, where each oracle efficiently searches the large-scale strategy spaces by using the group abstraction. (4) To execute policies, translate group-level indices $\tilde{I}_s^m$ to arm-level intervention policy.

putational efficiency. Critically, we show in Section 6.5 that this group-level planning is lossless — i.e., the policies we compute are the same minimax regret–optimal policies as would be computed if grouped arms were allowed *different* transition probabilities (within the same uncertainty intervals). In step (4), we map group-level policies back to individual-level policies by computing Whittle indices for each group $m \in [M]$, then assigning an index to each arm n within that group based on its current state s_n. Our policy is to intervene on mothers with the top K indices.

Double oracle In step (3), we adopt a double oracle (DO) framework [106], solving Eq. 6.2 by formulating the problem as a two-player zero-sum game between the RMAB *planner* and nature *adversary*, where the players aim to minimize and maximize regret respectively. The planner's *pure strategy* space is the finite set of all feasible RMAB policies π; the adversary has the continuous space of transition probabilities P within the uncertainty intervals $\overline{\underline{P}}^m$ for all $m \in [M]$. The algorithm maintains a finite pure strategy set for each player. For each iteration, we compute a mixed strategy Nash equilibrium (MSNE) on the game over the finite strategy sets. A *mixed strategy* is a probability distribution over pure strategies. In each iteration, the planner oracle computes a best response pure strategy π against the adversary's mixed strategy; π is added to the planner's finite strategy set. We

follow a symmetric approach to compute a best response P for the adversary. Upon termination, we return the final planner mixed strategy, which is guaranteed, under mild conditions, to be an ε-optimal minimax solution [170]. In practice, we terminate after T iterations [90]. The key technical challenge of using the double oracle approach is designing *planner* and *adversary* oracles for *group* RMABs.

6.4.1 Planner Oracle: WI for Mixed Strategy

An adversary mixed strategy β contains tuples (P_i, β_i) where β_i is the probability of playing pure strategy P_i. Similarly, a planner mixed strategy α contains tuples (π_i, α_i) where α_i is the probability of playing pure strategy π_i.

The planner oracle must compute an intervention policy π that minimizes regret with respect to a given adversary mixed strategy β over environment settings P_i. Since β and thus all P_i are fixed, and only the second term of regret in Eq. 6.1 depends on π, minimizing regret is equivalent to maximizing reward, to ensure that mothers engage with as many voice messages as possible. However, existing reward-maximizing RMAB algorithms assume a *single* environment P_i, versus a mixed strategy β over multiple P_i. To address this combinatorially hard problem, we develop a new heuristic approach that computes well-performing policies π based on strategically weighted combinations of Whittle indices.

Unfortunately, optimizing *exact* regret is at least PSPACE-hard [122]. Previous work optimized regret of the Lagrange relaxation [83], but relied on joint arm states which does not scale. *We introduce a decomposed notion of regret, allowing us to optimize regret of the full RMAB in a far more scalable way.* We call this *Whittle index regret*: the sum of Whittle indices played by a policy π compared to the optimal WIP. The key is that the Whittle index is a measure of "reward if played" — so agents who play arms with low Whittle indexes in lieu of arms with high Whittle indexes will incur large regret. As a further advantage, this regret notion naturally extends to groups — since the Whittle

index is a function only of transition probabilities and rewards, all of which are shared in a group under P_i — improving scaling.

Given states $\boldsymbol{s}$, denote the set of arms pulled by policy π as $\Phi^\pi(\boldsymbol{s}) = \{n \in [N] : \pi_n(\boldsymbol{s}) = 1\}$ where $\pi_n(\boldsymbol{s})$ is the action on arm n. The planner's Whittle index regret $R_W^{planner}(\boldsymbol{s})$ is:

$$\sum_{(P_i,\beta_i)} \beta_i \left[\max_{\substack{\kappa \subseteq [N] \\ |\kappa|=K}} \left\{ \sum_{n \in \kappa} (W^n_{P_i}(\boldsymbol{s}^n)) \right\} - \sum_{n \in \Phi^\pi(\boldsymbol{s})} W^n_{P_i}(\boldsymbol{s}^n) \right] . \tag{6.3}$$

The first term in Eq. 6.3 corresponds to a planner's optimal mixed strategy which plays the WIP corresponding to each setting of transition probabilities P_i in β. To minimize regret $R_W^{planner}$, we seek a policy π that plays Whittle indices as close as possible to the WIPs in the first term, which equivalently maximizes the second term. How to produce a *pure* strategy π that closely follows the *mixed* WIP policies of the first term is the key challenge. *We start by making the first term more closely computable as a pure strategy* with a relaxation that leads to relaxed regret, by moving the expectation over β_i inside the max over indices:

$$\max_{\substack{\kappa \subseteq [N] \\ |\kappa|=K}} \left\{ \sum_{n \in \kappa} \sum_{(P_i,\beta_i) \in \beta} \beta_i W^n_{P_i}(\boldsymbol{s}^n) \right\} . \tag{6.4}$$

We replace the first term of $R_W^{planner}(\boldsymbol{s})$ (from Eq. 6.3) with Eq. 6.4 to get $\tilde{R}_W^{planner}(\boldsymbol{s})$. This illuminates a heuristic for the planner oracle. Specifically, Eq. 6.4 can be computed exactly by a single policy π, meaning we can make $\tilde{R}_W^{planner}(\boldsymbol{s}) = 0$ by finding a π equivalent to Eq. 6.4. To do so, we compute Whittle indices for each pure strategy P_i, compute the β_i–weighted average index $\tilde{I}_s^m$ for each group m and state s, then follow the greedy strategy of a WIP. Since the expectation over β_i is pushed through the max (Eq. 6.4) we have $\tilde{R}_W^{planner}(\boldsymbol{s}) \leq R_W^{planner}$, but we show in appendix Fig. E.1 that this weighted index policy performs well, despite this relaxation. We call this approach Whittle Index for Mixed Strategy (WI4MS), given in Alg. 6.4.1. Whittle indices are computed via

Algorithm 6.4.1 WI4MS (Planner Oracle)

Input Adversary mixed strategy β

1: **for** $(P_i, \beta_i) \in \beta$ **do** {environment and probability i}
2: **for** $\{m = 1$ to $M\}$ and $\{s \in \mathcal{S}\}$ **do**
3: $\tilde{I}[m, s] \mathrel{+}= \beta_i \times \textsc{ComputeWI}(m, s, P_i^m)$
4: $\pi = \text{WIP}(\tilde{I})$ {implements Whittle index policy}
5: **return** π {planner pure strategy}

ComputeWI described in Alg. E.7.2 in the appendix.

6.4.2 Adversary Oracle: RegretMax Whittle Index

The adversary oracle must find one environment P that maximizes regret for the planner's current mixed strategy α over policies π_i to maximize the number of missed calls. To guide the search, we must address challenges both in maximizing regret of RMAB policies and in searching over a continuous strategy space $\overline{\underline{P}}^m$. Our insight is to maximize regret by manipulating the optimal RMAB policy (a Whittle index policy) to simultaneously *minimize* the values of Whittle indices acted on by the planner and *maximize* indices that are not.

We utilize again the notion of Whittle index regret, re-defined for the adversary oracle:

$$R_W^{adversary} = \mathop{\mathbb{E}}_{\boldsymbol{s}} \left[\sum_{n \in \Phi^{\pi_P^\star}(\boldsymbol{s})} W_P^n(\boldsymbol{s}^n)) \;\middle|\; \pi_P^\star, P \right]$$
$$- \sum_{(\pi_i, \alpha_i) \in \alpha} \alpha_i \left(\mathop{\mathbb{E}}_{\boldsymbol{s}} \left[\sum_{n \in \Phi^{\pi_i}(\boldsymbol{s})} W_P^n(\boldsymbol{s}^n)) \;\middle|\; \pi_i, P \right] \right). \tag{6.5}$$

Given an environment, P_i, Eq. 6.5 captures the difference in the Whittle indices collected by the optimal policy $\pi_{P_i}^\star$ versus the Whittle indices collected by the policies of the agent mixed strategies π_i. The WIP is a proxy for finding the most effective arms on which to intervene; intuitively, this means the adversary oracle should find P_i which maximizes the Whittle indices of arms played by

Algorithm 6.4.2 RegretMaxWI (Adversary Oracle)

Input: Mixed strategies (α,β), intervals $\overline{\underline{P}}^m$, group-mean budget K_M, $P = []$

1: $\{L_s^m\}_{s\in\mathcal{S}}^{m\in[M]} = \textsc{MonteCarlo}(\alpha,\beta)$ {simulation}
2: $K_{\text{TH}} = \textsc{FindThresh}(L, K_M)$ {returns action count of $\lceil K_M \rceil^{\text{th}}$ group-state}
3: **for** $\{m = 1$ to $M\}$ and $\{s \in \mathcal{S}\}$ **do**
4: $\quad obj[m,s] = \min$ if $(L_s^m \geq K_{\text{TH}})$ else max
5: **for** $m = 1$ to M **do**
6: $\quad P^m = \textsc{MinMaxWhittleBQP}(obj[m], \overline{\underline{P}}^m)$
7: **return** P {Adversary pure strategy}

the optimal policy *but not* played by the planner, and simultaneously minimizes the Whittle indices of arms played *only* by the planner policies.

The first challenge is to determine which arms the planner will act on in expectation. We propose a simple but effective solution which counts the number of times the arm-state pairs are acted on during Monte Carlo simulation of the planner's mixed strategy. Since the adversary operates at the group level, we then aggregate arm-state counts into group-state counts, denoted L_s^m for each group m and state s. The next question is which group-state indices to minimize or maximize. Intuitively, if we reduced all indices an equal amount, we would reduce *reward* but not *regret* since the optimal policy, i.e., the first term of Eq. 6.5, would reduce the same as the second. Thus, we need to strategically minimize some indices, but *maximize* others to induce an optimal policy that plays different arms. Specifically, we choose to minimize the indices of the top $K_M = \frac{K}{N/M}$ — i.e., the budget normalized by average group size — entries of L_s^m, approximating the top K choices of the agent mixed strategy in expectation. Then we maximize the Whittle indices of all group-state pairs below that threshold.

The second challenge is to find transition probabilities P that minimize or maximize the Whittle indices of a group over its transition probability intervals. This problem has general implications, e.g., for optimistic or pessimistic search over uncertainty sets in online learning. We derive a novel

binary-quadratic program that, given a group and objective for each state (min, max, or null), computes a P^m that optimizes the indices for all states simultaneously, detailed in the appendix as MinMaxWhittleBQP (Eq. E.13). We give the full adversary oracle algorithm, RegretMaxWI, in Alg. 6.4.2 and empirically demonstrate its good performance in the appendix Fig. E.2.

6.5 Theoretical Regret Guarantee

In Section 6.4, we proposed an approach to compute a minimax regret–optimal strategy against an adversary choosing the same transition probabilities for all arms in the same group from their corresponding intervals. However, arms within the same group may not have identical transition probabilities. Also, it is not intuitive that a minimax regret–optimal policy, when the adversary chooses the same transition probabilities for all the arms in a group, also minimizes max regret when the adversary chooses different transition probabilities for the arms in a group from their corresponding intervals. In this section, we show this is true under mild assumptions. In particular, the minimax regret–optimal strategy of the planner is the same against an adversary choosing transition probabilities at the group level as against an adversary choosing transition probabilities at the individual level.

Let Π be the planner's pure strategy space of all individual-level policies, i.e., all choices of subsets of arms with cardinality K. Then we define mixed strategy sets for the planner at *individual-level*, $\Delta_I(\Pi)$, and *group-level*, $\Delta_M(\Pi)$, where $\Delta_M(\Pi) \subseteq \Delta_I(\Pi)$ is a restricted set of mixed strategies in which the planner is indifferent between arms in the same group and state (see Appendix E.4.2 for definition). Next, let $\mathscr{P}$ be the adversary's pure strategy space, containing all individual-level policies, i.e., choices of transition probabilities $\{P^n\}_{n\in[N]}$ respecting the given uncertainty intervals $\overline{\underline{P}}^{\varphi(n)}$. Similarly, we define mixed strategy sets for the adversary at individual-level, $\Delta_I(\mathscr{P})$, and group-level, $\Delta_M(\mathscr{P})$, where $\Delta_M(\mathscr{P}) \subseteq \Delta_I(\mathscr{P})$ is a restricted space that assigns same transition

probabilities to all arms within a group.

For $X, Y \in \{I(\text{individual}), M(\text{group})\}$, the regret game with X-level planner and Y-level adversary is noted as X/Y. The X/Y regret of a planner's mixed strategy $\alpha \in \Delta_X(\Pi)$ against an adversary's mixed strategy $\beta \in \Delta_Y(\mathscr{P})$ is:

$$R(\alpha, \beta) := \sum_{i \in [|\Delta_X(\Pi)|]} \sum_{j \in [|\Delta_Y(\mathscr{P})|]} \alpha_i \beta_j R(\pi_i, P_j) \ ,$$

where α_i is the ith pure strategy of the X-level planner and β_j is the jth pure strategy of the Y-level adversary. Let $\alpha^\star_{X,Y}$ be the planner's mixed strategy of a X/Y game, defined:

$$\min_{\alpha \in \Delta_X(\Pi)} \max_{\beta \in \Delta_Y(\mathscr{P})} R(\alpha, \beta) = \max_{\beta \in \Delta_Y(\mathscr{P})} R(\alpha^\star_{X,Y}, \beta)$$

which holds since the regret game is a two-player zero sum game, making minimax regret equal to maximin reward. We call this the worst-case regret for $\alpha^\star_{X,Y}$.

We first show in Theorem 6.5.1[†] that, when all arms within the same group have the same transition intervals, the minimax I/I regret is equal to the minimax M/I regret.

Theorem 6.5.1. *The worst-case regrets of $\alpha^\star_{I,I}$ and $\alpha^\star_{M,I}$ against an adversary operating at the individual level is equal:*

$$\max_{\beta \in \Delta_I(\mathscr{P})} R(\alpha^\star_{I,I}, \beta) = \max_{\beta \in \Delta_I(\mathscr{P})} R(\alpha^\star_{M,I}, \beta) \ .$$

Similarly, in Theorem 6.5.2, we show that, when all arms within the same group have the same transition intervals, the minimax I/M regret is equal to the minimax M/M regret.

Theorem 6.5.2. *The worst-case regrets of $\alpha^\star_{I,M}$ and $\alpha^\star_{M,M}$ against an adversary operating at the group*

[†]Proofs of Theorem 6.5.1, 6.5.2, and 6.5.3 are given in Appendix E.5.

level are equal:

$$\max_{\beta \in \Delta_M(\mathscr{P})} R(\alpha^\star_{I,M}, \beta) = \max_{\beta \in \Delta_M(\mathscr{P})} R(\alpha^\star_{M,M}, \beta) \,.$$

Finally, we use these results to establish our main result in Theorem 6.5.3 that the worst-case regret of $\alpha^\star_{M,M}$ is equal to the worst-case regret of $\alpha^\star_{I,I}$ when (1) all arms in the same group have the same intervals and (2) there exists a surjective function ψ that maps $\Delta_I(\mathscr{P})$ to $\Delta_M(\mathscr{P})$ that preserves the regret ordering of planner and adversary strategies (formal definition and example ψ given in Appendix E.5.1).

Theorem 6.5.3. *If there exists an order-preserving map, then the worst-case regret of $\alpha^\star_{M,M}$ is equal to that of $\alpha^\star_{I,I}$, against an individual-level adversary, that is,*

$$\max_{\beta \in \Delta_I(\mathscr{P})} R(\alpha^\star_{M,M}, \beta) = \max_{\beta \in \Delta_I(\mathscr{P})} R(\alpha^\star_{I,I}, \beta) \,.$$

Theorems 6.5.1, 6.5.2, and 6.5.3 together establish that the minimax regret–optimal strategy is the same whether the planner and adversary play at individual or group level. In particular, this result ensures that, under some conditions, the minimax regret–optimal strategy obtained by our algorithm GROUPS, which implements group-level planner and adversary, is also minimax regret–optimal against an individual-level adversary.

6.6 Experiments

6.6.1 Experiment Setup

ARMMAN maternal health domain Every week, ARMMAN's automated system delivers prerecorded health messages to each enrolled mother with information tailored to the mother's gestational age. If mothers stop listening to the messages, healthcare workers can deliver interventions to try to improve mothers' engagement. We evaluate *the increase in number of health messages*

mothers listen to using GROUPS to target interventions compared to existing baselines. To construct a simulation environment, we use a real anonymized dataset from ARMMAN's records of weekly program engagement data for 15,336 mothers (though we note that ARMMAN's larger service areas operate on the scale of hundreds of thousands). A mother is "engaged" if they listen to at least 30 seconds of a message that week. Thus, states are {`not engaged`, `engaged`} with rewards 0 and 1, respectively. To create an arm–group mapping, we run K-means clustering on the engagement data and compute uncertainty intervals via bootstrapping followed by multiple imputation to compute standard deviations of the means[140]. Statistics on the uncertainty intervals and group sizes are shown in appendix Figs. E.6 and E.7. For details on the dataset and consent for collection, see appendix E.11.

In the experiments, the default parameters match the intervention setup used by ARMMAN, i.e., budget $K = 100$, $N = 15{,}320$ mothers, and $M = 40$ groups. For sensitivity analysis, we vary the budget, horizon, and number of mothers. Additional analysis varying uncertainty interval width, number of groups, and distribution of group sizes are included in appendix Fig. E.4.

Additional domains To demonstrate wider applicability, we include results from two additional domains. The **TB** domain is constructed from an anonymized dataset of daily adherence to tuberculosis medication[82]. States, rewards, and groups were derived analogously to the maternal health setting; complete details are in appendix E.12, including group statistics in Figs. E.8 and E.9. In our experiments, the default setting has $N = 8{,}350$ arms, $M = 60$ groups, budget $K = N/10$, and $A_\sigma = 3$, i.e., interval width of 3 standard deviations. We vary the budget, number of groups, and A_σ. Finally, we use the **Synthetic** benchmark domain from recent robust RMAB work[83]. This domain considers three "arm types" $[U, V, W]$ with different intervals, designed so that non-robust policies incur greater regret than robust ones. We augment the domain to allow homogeneous groups of each arm type, where the size and proportion of groups of each type may vary.

In our experiments, the default setting has $N = 18{,}000$ arms, $M = 36$ groups, where $1/3$ of groups are composed of each of the arm types, and budget $K = 100$. We run sensitivity analysis on K, the proportion of groups made up of each arm type, and a "block group" setting which joins all arms of a given type into a single group.

Evaluation To evaluate performance, we plan at the group level but simulate individuals within groups independently, where each individual undergoes state transitions based on their own state, action, and transition probabilities. All experiments use horizon $H = 10$ and report the average of 30 seeds. We measure total reward with discount factor $\gamma = 0.9$. In Fig. 6.3, we evaluate each approach in terms of regret (Eq. 6.1), computed by simulating each planner strategy against the full set of adversary pure strategies and selecting one that maximizes regret. Note, there is no actual deployment of the proposed algorithm; all results are simulated.

Baselines First, we compare against the state-of-the-art robust RMAB method, *DDLPO*, for small settings in which DDLPO can complete[83]. For larger-scale experiments with tens of thousands of arms, no other robust methods are tractable, so we compare against several scalable non-robust baselines. The non-robust baseline from Mate et al. [104] assumes all environment parameters take the median of their uncertainty intervals then computes a reward-maximizing WIP; this strategy was employed in a recent real-world pilot[104]. We consider two additional non-robust variants which assume that all parameters take the lower bound of the uncertainty interval (*pessimist*) or the upper bound (*optimist*), then compute a WIP strategy. Finally, *random* plans a WIP strategy against an environment that is uniformly randomly sampled from the uncertainty intervals.

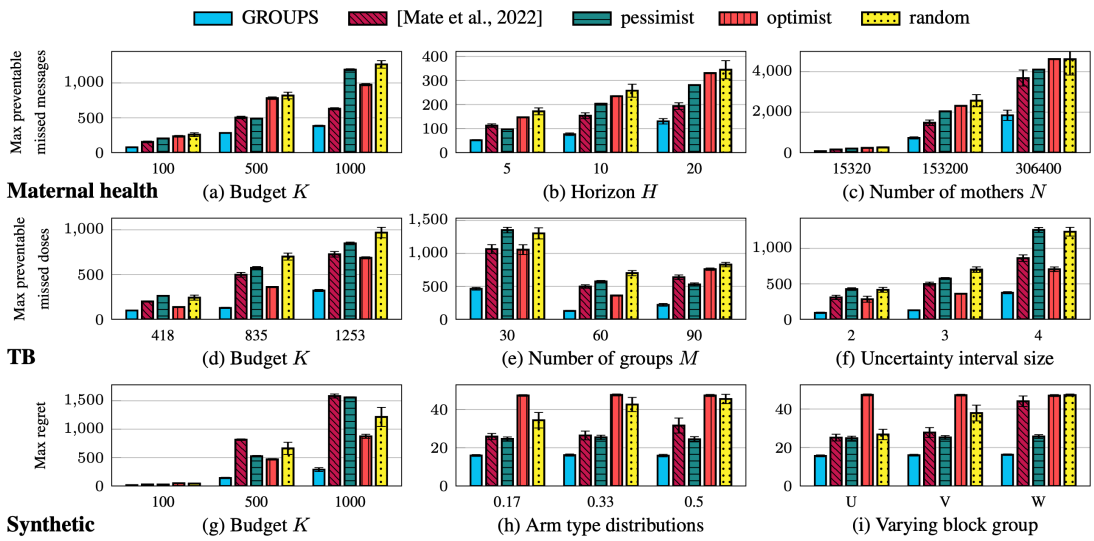


Figure 6.3: Max regret (lower is better) incurred by GROUPS, our robust solution approach, compared to non-robust baselines across various settings. *(a–c) Maternal health.* For (c), the number of arms is increased by multiplying each group size by a constant factor, i.e., 1, 10, and 20, but M is constant. *(d–f) TB.* For (d), budgets are 5%, 10%, and 15% of N. *(g–i) Synthetic.* For (h), the x-axis is the fraction of groups of arm type U — the fraction of type V is always 0.33, and the remaining fraction are type W. For (i) the x-axis denotes the arm type that has been combined into a single group of 6000 arms, where the other two types are split across 12 groups each of size 500. In the maternal health and TB settings, regret can be interpreted, in real-world terms, as the maximum preventable missed health messages and doses, respectively, across the uncertainty space.

6.6.2 Results

Fig. 6.3 shows GROUPS outperforms baselines in terms of max regret across several settings. Fig. 6.3(a–c) shows results for the **maternal health** setting of ARMMAN. In particular, Fig. 6.3(c) shows that GROUPS scales past 300,000 arms, representing more than a 1000× increase over the robust state-of-the-art to meet a key need of real-world deployment settings. Moreover, across experiments, the max regret of GROUPS is nearly half that of the non-robust strategy used in Mate et al. [104]. In other words, our simulations demonstrate that compared to the best non-robust strategy **GROUPS could prevent mothers from missing thousands of pregnancy-related health messages, each containing potentially life-saving care information.**

On the **TB** domain (Fig. 6.3(d–f)), we see again that GROUPS performs well across various

strategies for grouping and computing uncertainty, even with very imbalanced group sizes. On the **synthetic** domain (Fig. 6.3(g–i)), across various budgets and grouping strategies, the non-robust baselines vary in performance and are sometimes worse than random, demonstrating the need for reliable robust policies. Moreover, Table 6.1 shows that GROUPS even outperforms the state-of-the-art DDLPO in terms of regret on the synthetic benchmark dataset for problems sizes small enough for DDLPO to complete (i.e., $N < 100$). The superior performance of GROUPS is due to our Whittle-based policies which specialize to two-action settings, in contrast to the more general but highly stochastic deep learning–based policies of DDLPO.

	GROUPS	DDLPO
$N = 6$	0.64 ± 0.05	1.00 ± 0.06
$N = 9$	0.47 ± 0.06	0.98 ± 0.05
$N = 12$	0.45 ± 0.06	0.88 ± 0.05

Table 6.1: Regret of GROUPS vs. robust method DDLPO on Synthetic. We set $M = N$ and $K = 1$ to match the evaluation in Killian et al. [83]. GROUPS incurs less regret.

Supported by Theorem 6.5.3, GROUPS scales significantly without incurring additional regret. In Appendix E.9, we demonstrate the significant runtime improvement of GROUPS as M decreases, holding N constant. The scalability of our approach is critical for robust RMAB solutions to actually be deployed in real-world, low-resource settings.

6.7 Conclusion

The GROUPS algorithm we introduce presents several key advances to make RMABs more useful in practice, enabling simultaneous *scaleup* and *robustness to uncertainty*. We are working with ARMMAN to deploy GROUPS to positively impact maternal health, demonstrating the real-world capabilities this work enables. Most notably, **our simulation experiments demonstrate that our robust planning method could help ARMMAN prevent mothers from missing thousands of**

health messages, a promising result that we hope to translate into practice to help deliver life-saving health information to otherwise under-served mothers.

7

Conclusion

To conclude, this thesis makes key advances in AI to help enable effective resource allocation in public health. In working closely with collaborators, I identified key thematic challenges of resource heterogeneity and planning under model and observational uncertainty. I developed state of the art algorithms for addressing these challenges using multi-agent planning, optimization, machine learning, and deep reinforcement learning. Moving forward, I aim to continue to collaborate with interdisciplinary teams to research and develop effective AI tools with the goal of improving health

equitably for all. From a technical viewpoint, there are many areas in which I would like to continue to advance this agenda.

Learning from real-world data. When learning from historical trajectories, some decisions may be generated based on variables that are impartially recorded (e.g., ambiguous active medication lists or adherence rates) or not recorded (e.g., patient preferences or decision-maker bias.) Training naively on such data can lead to spurious policies with ambiguous support. To address this, I will work with healthcare partners to develop latent variable models that can account for specific structures of unobserved confounders, to maximize our ability to learn from existing historical data. Also of key interest is the complementary problem of designing closed-loop learning systems whose decisions, when deployed, will impact the data distributions that the system will later (1) learn from and (2) be evaluated on by decision makers. A key unanswered question in AI planning is how to design policies that are jointly useful for learning (e.g., requiring *some* exploration) and allow for statistically rigorous evaluation (e.g., requiring *minimal* exploration to allow for post hoc construction of synthetic cohorts).

Safety and Reliability. Especially in high-stakes real-time settings (like the intensive care unit), there are certain policy choices that are unacceptable e.g., unintentionally forcing vital signs into an unsafe range. I will build on my experience in constrained RL to develop methods for giving probabilistic guarantees that policies will avoid pre-specified unsafe state space regions. Moreover, I will develop methods for statistically evaluating these guarantees on real-world offline data. Of course, sometimes guarantees may be challenging to satisfy, especially in more volatile regions of the state space. So it will also be important to build on the constrained policy methods to incorporate "learning to defer" to human experts when they enter regions of the state space where safety guarantees may be violated.

In sum, I believe that AI will have a positive impact on the world, especially in the domain of health. Developed as complementary components of holistic systems, conceived, developed and

analyzed by diverse interdisciplinary teams, AI may achieve its promise.

A

Appendix to Chapter 1

A.1 Constructing upper and lower bounds

The algorithm for constructing $\mathcal{U}$ and $\mathcal{L}$, the bounded representations of $V^i(s^i, \lambda)$, is given in Alg. A.1.1.

A.2 Proof of Theorems 1.5.1 and 1.5.2

Theorem 1.5.1. *λ^i are $\frac{\sigma^2}{n}$-sub-Gaussian where $\sigma^2 = \frac{1}{4}\left(\frac{r_{max}}{c_{min}(1-\beta)}\right)^2$*

Proof. By Hoeffding's Lemma, if the values of a random variable can be bounded almost surely by a lower bound a and upper bound b, and its expected value is 0, then the random variable is $\frac{(b-a)^2}{4}$-sub-Gaussian. Note that shifting the expected value to be centered around the true mean introduces the factor n in the denominator, the number of samples used to estimate the mean. It remains to show that λ^i is bounded. The lower bound of λ^i is 0 due to the nature of the budget constraint, i.e., the budget constraint is $\leq$. To upper bound λ^i, we consider the Lagrange LP (i.e., Eq. 1.7) for a single arm. We assume there always exists a 0 cost action (bandit assumption) which achieves at least 0 zero reward. Then if any single action charge λc_{min} is greater than any possible long term future reward $\frac{r_{max}}{1-\beta}$, the optimal policy will always choose the no-cost action, since the difference of the future reward and the charge will always be negative otherwise. Thus lambda cannot be increased further in the objective of the LP than $\frac{r_{max}}{c_{min}(1-\beta)}$, and is thus the upper bound on λ^i. □

Theorem 1.5.2. *The number of samples n needed to estimate the sample mean of λ^i within an error ε and with confidence $1-\delta$ is lower bounded as:*

$$n \geq \frac{1}{2\varepsilon^2}\left(\frac{r_{max}}{c_{min}(1-\beta)}\right)^2 \log\left(\frac{1}{\delta}\right) \tag{A.1}$$

Proof.

$$P(\hat{\lambda^i} \geq \lambda^i + \varepsilon) \leq \exp\left(-\frac{n\varepsilon^2}{2\sigma^2}\right) \leq \delta \quad \text{Hoeffding bound} \tag{A.2}$$

$$n \geq \frac{2\sigma^2}{\varepsilon^2}\log\left(\frac{1}{\delta}\right) \tag{A.3}$$

$$n \geq \frac{1}{2\varepsilon^2}\left(\frac{r_{max}}{c_{min}(1-\beta)}\right)^2 \log\left(\frac{1}{\delta}\right) \tag{A.4}$$

□

A.3 Modified Knapsack

The integer program used to compute policies in section 1.6 is as follows:

$$\max_{X} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} x_{i,j} Q^i(s^i, a_j, \lambda_{min}) \tag{A.5}$$

$$\text{s.t.} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} x_{i,j} c_j \leq B \tag{A.6}$$

$$\sum_{j=0}^{M-1} x_{i,j} = 1 \quad \forall i \in 0 \ldots N-1 \tag{A.7}$$

$$x_{i,j} \in \{0, 1\} \tag{A.8}$$

where $Q^i(s^i, a_j, \lambda_{min})$ is the action value function associated with arm i. Note that $Q^i(s^i, a_j, \lambda_{min})$ can be readily computed using the value functions returned by BLam and SampleLam via algorithm A.3.1.

Note that this integer program is a generalization of the {0,1} knapsack problem, in which multiple "actions" can be taken on each item to achieve different values at the corresponding action costs. In practice we use Gurobi to solve the integer program, but we also give a dynamic program solution in Alg. A.3.2 to prove the worst-case computational complexity. The idea is to compute a table $z[i, j]$ where z is the maximum achievable value considering to the first i items and respecting a budget j. The table entries are computed iteratively using using a standard dynamic programming approach. Once the table is computed, the optimal value can be retrieved from the entry $z[N-1, B]$, and the corresponding optimal actions can be recovered via a backtrace. Both computing the table and running the backtrace take $\mathcal{O}(N|\mathcal{A}|B)$ time, as well as $\mathcal{O}(NB)$ and $\mathcal{O}(N)$ memory respectively.

Note that all algorithms in this chapter employ this same knapsack procedure to select policies using their respective estimates of Q.

A.4 Additional Experimental Results

BLam's superior runtime emerges when there is existing problem structure of which its bounding technique can take advantage. In the medication adherence experiment, the primary structure it identifies is that both high and low adherence patients have small responses to interventions. In terms of $V(s, \lambda)$, the term in the LP that BLam "bounds" with a cheaper representation, such small-response patients have a $V(s, \lambda)$ curve whose derivative quickly goes to zero. Such curves can be closely approximated with "bounded" curves made up of only a few test points (see Fig. 1.1) — and the closer the approximation of the bounds, the faster BLam will converge.

In Fig. A.1, we investigate a second mode by which BLam can achieve speedups, as the budget B is varied. As the budget increases, resources are less constrained which intuitively makes planning easier, since there are more resources available to take each arm's individually optimal action (i.e., the action one would take on an arm if planning for that arm in isolation). This allows all methods to tend toward the same high reward as the budget is increased. However, BLam's adaptivity allows it to recognize when the problem is less constrained to automatically converge even more quickly to the optimal solution, as demonstrated by the increasing runtime gap between Hawkins and BLam as the budget is increased. These gains will be most drastic when the true λ_{min} is 0, since (1) this can be quickly verified in a single step of BLam if its upper bound returns 0 and (2) the larger the budget B, the more likely BLam will return an upper bound of 0, since the larger the slope of the term $\frac{\lambda B}{1-\beta}$ will become. Figs. A.2 and A.3 replicate the results from Figures 1.5 and 1.4, respectively, with log-scale runtimes included.

Finally, in Fig. A.4, we give a sensitivity analysis of SampleLam that demonstrates the change in

its performance as the underlying distribution of arms shifts. We simulate a shifting distribution of arms by sampling an α fraction of arms from a *fully random* distribution, which has uniform random transition probabilities and rewards (with $r(s) \in [0, 1]$) and a $1 - \alpha$ fraction of arms from the adherence simulation data. The arms from the adherence simulation data have $d = 4$ and the arms from the fully random distribution have a corresponding 40 states. Arms from both distributions have four actions with costs in $[0, 1, 2, B]$, as in the adherence simulation in section 1.7. We compare the discounted sum of rewards, using discount factor 0.95, averaged over all arms N and 20 simulations, over L = 40 rounds. We also subtract the mean reward collected from the **nobody** policy to make the differences between **SampleLam** and the best policy **Hawkins** more clear. Results are shown for a budget of $B = 0.1$ and $N = 250$ arms.

Note that Fig. 1.2a was generated using arms from the fully random distribution – this suggests that SampleLam should be good at estimating λ_{min} as $\alpha \to 1$ and should correspondingly achieve performance comparable to Hawkins. Also note that SampleLam performed suboptimally on the adherence simulation data in section 1.7, suggesting that λ_{min} was not equal to the sample mean of λ_is in that distribution in general. We see that when α is near 0, SampleLam's poor performance on the adherence simulation data is shown as expected, but as α increases, and more arms are sampled from the fully random distribution, SampleLam's estimates of λ_{min} become progressively closer to the true value, making the policies it computes progressively closer to the well-performing Lagrange policy.

A.5 Lagrange Policy vs. Index Policy

Here we expand on the relationship between the Lagrange policy and the Whittle index policy for binary-action RMABs, by giving the following Lemma:

Lemma A.5. *The Lagrange policy always produces equal or greater rewards than the Whittle index*

policy in binary-action RMABs as $N, B \to \infty$ with B/N fixed.

Proof. This can be seen via the results from Weber & Weiss [161] which prove the asymptotic optimality of the Whittle index policy under indexability (see Whittle's definition of indexability [163]). To prove their result, they first give Theorem 1 which shows that, in the binary-action case, as $N, B \to \infty$ with B/N fixed, the optimal return from the constrained problem (Eq. 1.3) converges to the optimal return of the Lagrangian relaxation of the problem (Eq. 1.7). This implies that if one could compute and follow the policy that minimizes the Lagrange bound (i.e., the Lagrange policy), one could achieve the asymptotically optimal reward. They then go on to show that under indexability, index policies replicate the Lagrange policy. However, since we are computing the Lagrange policy directly, we bypass the step of computing indices, but we maintain the asymptotic performance guarantees that they enjoy. This holds true regardless of whether the problem is indexable. □

Algorithm A.1.1: BuildBounds

```
Data: D, G, N
1  U = [] ;                          // list of dicts for upper bound pieces
2  L = [] ;                          // list of dicts for lower bound pieces
3  for i ∈ 0, ..., N − 1 do
4      /* Computing upper bounds, start from back                          */
5      j = |G[i]| − 1;
6      U[i,j]['m'] = 0;                          // last slope always 0 for UB
7      U[i,j]['b'] = 0;                          // last intercept is arbitrary
8      λ_test = G[i,j];
9      /* set up the next line: y=mx+b                                     */
10     y = U[i,j]['m'] ∗ λ_test + U[i,j]['b'];
11     for j ∈ |G[i]| − 2, ..., 0 do
12         U[i,j]['m'] = D[i,j + 1];
13         /* b=y-mx                                                       */
14         U[i,j]['b'] = y − U[i,j]['m'] ∗ λ_test;
15         λ_test = G[i,j];
16         /* set up the next line: y=mx+b                                 */
17         y = U[i,j]['m'] ∗ λ_test + U[i,j]['b'];
18     /* Computing lower bounds, start from front                         */
19     y = 0;
20     for j ∈ 0, ..., |G[i]| − 1 do
21         λ_test = G[i,j];
22         L[i,j]['m'] = D[i,j];
23         /* b=y-mx                                                       */
24         L[i,j]['b'] = y − L[i,j]['m'] ∗ λ_test;
25         /* set up the next line: y=mx+b                                 */
26         y = L[i,j]['m'] ∗ G[i,j + 1] + L[i,j]['b'];
27 return U,L
```

Algorithm A.3.1: Compute Action Value Function

```
Data: V, T, R, C, λ, β
Q = [] ;                                   // hold the action value function
for s ∈ S do
    for a ∈ A do
        Q[s, a] = R[s] − λ ∗ C[a] + β ∗ Σ_{s′∈S} V[s′] ∗ T[s, a, s′]
return Q
```

Algorithm A.3.2: ActionKnapsackDP

```
Data: Q, C, B
;                                  // Q filtered to current state of each arm i
z = [] ;                                          // N × B Table for solving DP
for j ∈ 0, ..., B do
    /* Initialize top row to 0                                              */
    z[0, j] = 0;
for i ∈ 0, ..., N − 1 do
    for j ∈ 0, ..., B do
        z[i, j] = z[i − 1, j] + Q[i, 0];
        for k ∈ 1, ..., |A| − 1 do
            if C[k] ≤ j then
                z[i, j] = max{z[i, j], z[i − 1, j − C[k]] + Q[i, k]};
return z
```

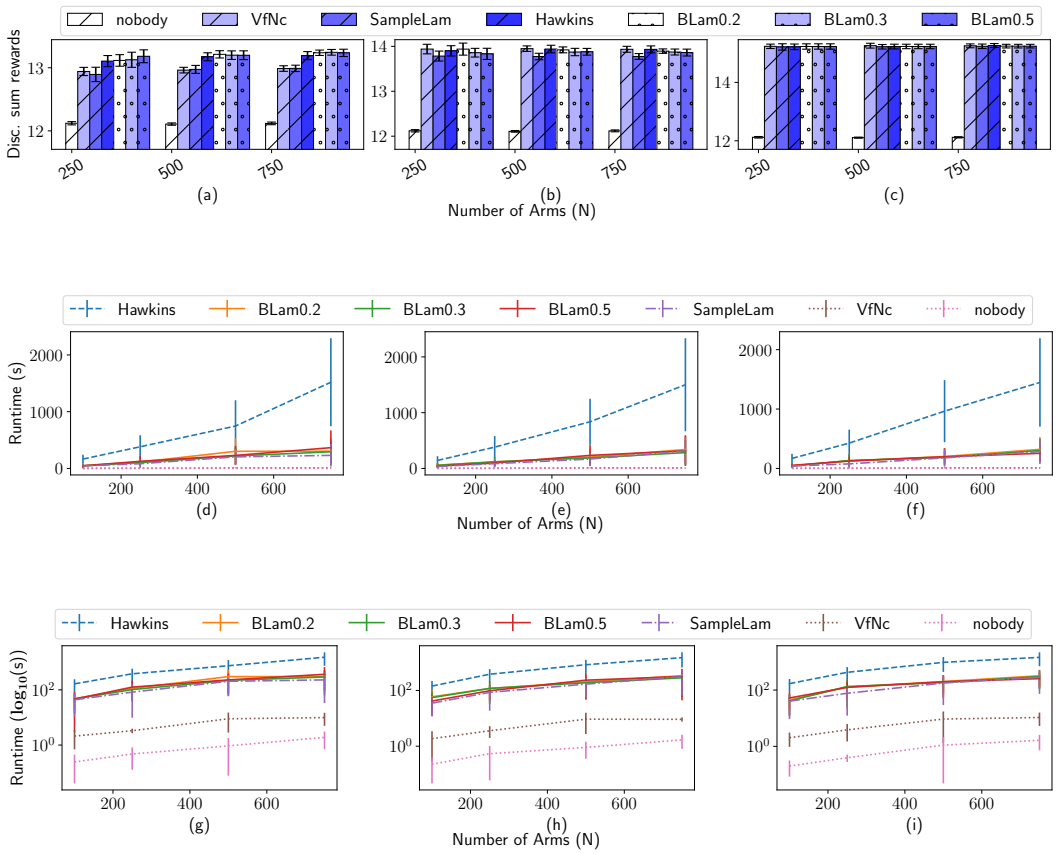


Figure A.1: Rewards (top row), linear-scale runtimes (middle row) and log-scale runtimes (bottom row) on the health care dataset with $d = 4$ adherence levels. Columns represent a budget of $0.1N$, $0.2N$, and $0.5N$, respectively. At all values of ε, BLam significantly outperforms VfNc when the budget is small and the tradeoff between individual actions is important. BLam also scales much better than Hawkins, achieving a 5 times speedup in the 0.1 budget case and 6 times speedup in the 0.2 and 0.5 budget cases. BLam, at all values of ε, scales similarly to SampleLam on this dataset.

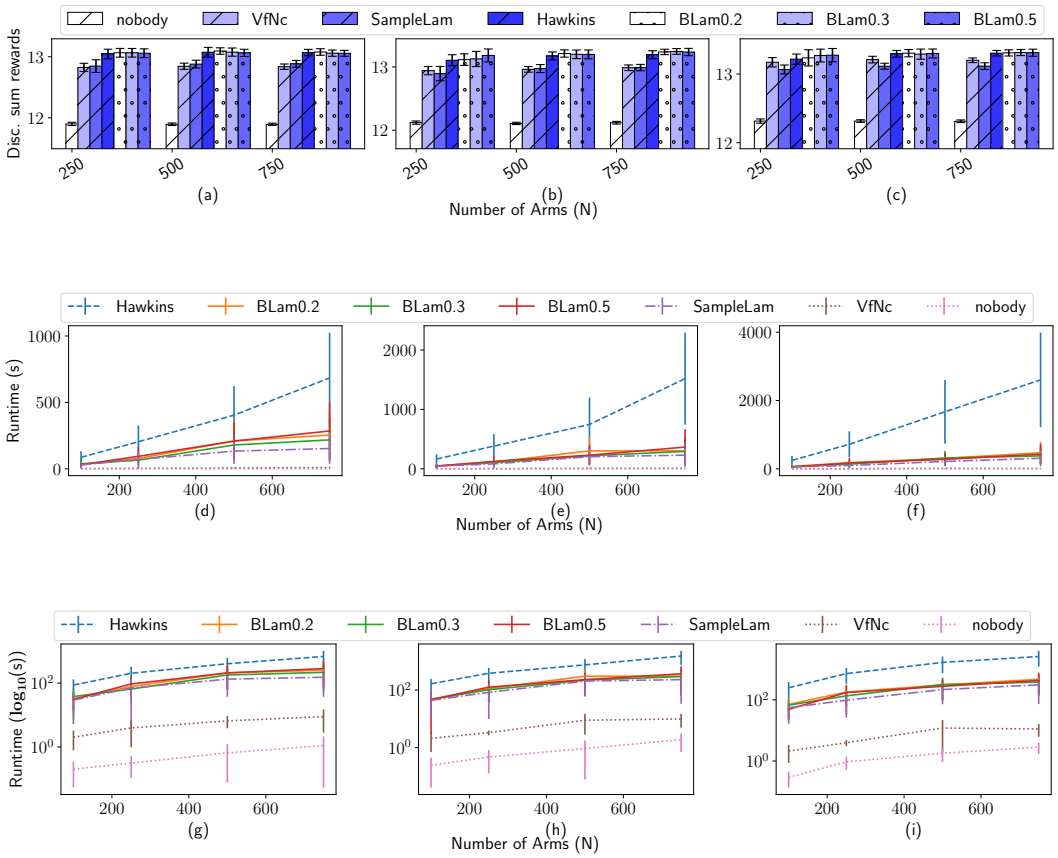


Figure A.2: Rewards (top row), linear-scale runtimes (middle row) and log-scale runtimes (bottom row) on the health care dataset with budget $0.1N$. Columns represent $d = 3, 4, 5$ adherence levels, respectively. At all values of ε, BLam significantly outperforms VfNc. Further, the Hawkins LP scales quadratically in the number of states on each arm, while BLam identifies problem structure that keep the underlying LPs small, making speedups more dramatic as the problem size increases. BLam, at all values of ε, scales similarly to SampleLam on this dataset. (Same as Fig. 1.5, but with log-scale runtime included.)

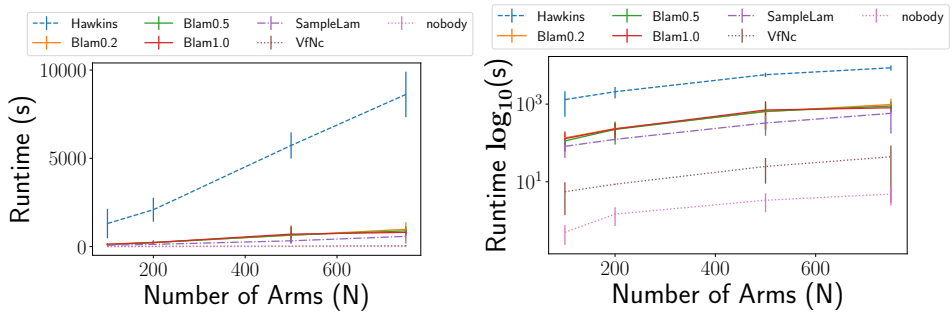


Figure A.3: Linear-(left) and log-scale (right) runtimes on the Greedy/Reliable/Easy simulation domain. BLam again scales extremely well, but SampleLam's scaling advantage is more evident on this dataset. (Left plot is the same as Fig. 1.4.)

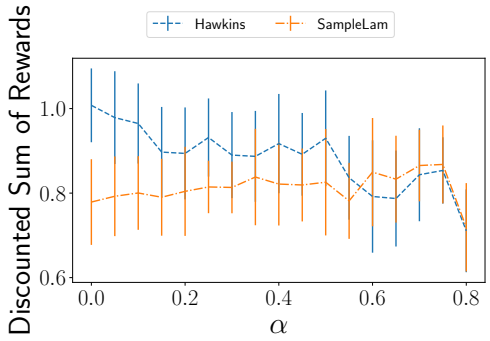


Figure A.4: Sensitivity analysis of SampleLam, where α fraction of arms are sampled from a fully random distribution and $1 - \alpha$ arms are sampled from the adherence simulation data. As more arms are sampled from the fully random distribution, SampleLam's estimates become closer to λ_{min}, making the policies it computes progressively closer to the well-performing Lagrange policy.

B

Appendix to Chapter 2

B.1 Proof of convergence for MAIQL

In this section, we provide a detailed proof of the convergence for MAIQL. We begin by stating 2 standard assumptions for establishing the convergence guarantee of Q-learning in the average-reward setting, and then add a third that's required for two time-scale convergence.

Assumption 1 (Uni-chain Property)**.** *There exists a state s_0 that is reachable from any other state*

$s \in S$ with a positive probability under any policy.

This property formalizes the notion that there aren't any 'forks' in the MDP, in each of which very different outcomes could occur. This is important because, if there were a fork, the notion of 'average' reward would be ill-defined as it would depend on which 'fork' gets taken.

Assumption 2 (Asynchronous Update Step-Size)**.** *The sequence of step-sizes $\{\alpha(t)\}$ satisfy the following properties for any $x \in (0,1)$:*

$$\sup_t \frac{\alpha(\lfloor xt \rfloor)}{\alpha(t)} < \infty$$

$$\sup_{y \in [x,1]} \left| \frac{\sum_{m=0}^{\lfloor yt \rfloor} \alpha(m)}{\sum_{m=0}^{t} \alpha(m)} - 1 \right| \to 0$$

This is a condition that is required to show that updating $Q(s, a_j)$ values one at a time with an ε-greedy policy is equivalent to updating all the $Q(s, a_j)$ values together, in expectation.

Assumption 3 (Relative Step-Size)**.** *The two sequences of step-sizes, $\{\alpha(t)\}$ and $\{\gamma(t)\}$, satisfy the following properties:*

(A) *Fast Time-Scale:* $\displaystyle\sum_{t=0}^{\infty} \alpha(t) \to \infty, \qquad \sum_{t=0}^{\infty} \alpha^2(t) < \infty$

(B) *Slow Time-Scale:* $\displaystyle\sum_{t=0}^{\infty} \gamma(t) \to \infty, \qquad \sum_{t=0}^{\infty} \gamma^2(t) < \infty$

$$\text{(C)} \quad \lim_{t \to \infty} \frac{\gamma(t)}{\alpha(t)} \to 0$$

An example of possible step sizes for which this condition is true is $\alpha(t) = \frac{1}{t}$ and $\gamma(t) = \frac{1}{t \log t}$. In our experiments we use $\alpha(t) = \frac{C}{\lceil \frac{t}{D} \rceil}$, and $\gamma(t) = \frac{C'}{1 + \lceil \frac{t \log(t)}{D} \rceil}$.

We then detail the proof for Theorem 2.4.3 below. This proof involves mapping the discrete Q and λ updates from the MAIQL algorithm (Section 2.4) to updates in an equivalent continuous-

time Ordinary Differential Equation (ODE). This conversion then allows us to use the analysis tools created to analyse the evolution of two-timescale ODEs to show that our coupled updates converge. The proof detailed below broadly follows along the lines of Avrachenkov & Borkar [12], but where they discuss convergence in the binary action case, we generalize their proof to the multi-action scenario by using the notion of multi-action indexability from Glazebrook et al. [52].

Theorem 2.4.3. *MAIQL converges to the optimal multi-action index $\lambda^*_{s,a}$ for a given state s and action a under Assumptions 1, 2, 3, and the problem being multi-action indexable.*

Proof. To convert these discrete updates to ODEs, we map a given time-step t to a point $\tau = T(t)$ in a continuous time, such that any time $T(t) = \sum_{m=0}^{t} \alpha(t)$. Because we're parameterising the time with α (rather than γ) we call τ the fast time-scale. To make this more concrete, we define $Q(\tau)$ as a function of the Q-value with time, and set $Q(T(t)) = Q^t$ to the value of the Q-function after t updates . Then, for values of $T(t) < \tau < T(t+1)$, $Q(\tau)$ is assumed to be linearly interpolated between Q^t and Q^{t+1}, creating a continuous function of τ. Similarly, we define $\lambda(\tau)$ such that $\lambda(T(t)) = \lambda^t$

We can then re-arrange the terms in Equation 2.9 to create an ODE that characterizes the value of $Q(\tau)$:

$$
\begin{aligned}
Q^{t+1}(s, a_j) = Q^t(s, a) + \alpha(t)\big[[r(s) - \lambda^t_{s,a_j} c_j - f(Q^t) \\
+ \max_{a'_j \in \{0,1\}} Q^t(s', a'_j)] - Q^t(s, a_j)\big] \\
\Rightarrow \underbrace{\frac{Q^{t+1}(s, a_j) - Q^t(s, a_j)}{\alpha(t)}}_{\dot{Q}(\tau)} = [r(s) - \lambda^t_{s,a_j} c_j - f(Q^t) \\
+ \max_{a'_j \in \{0,1\}} Q^t(s', a'_j)] - Q^t(s, a_j)
\end{aligned}
$$

where $\dot{Q}(\tau)$ is the derivative of $Q(\tau)$ and corresponds to the slope of the interpolated function in the range $(T(t), T(t+1))$.

Similarly, we can re-arrange Equation 2.10 to get the ODE for $\lambda(\tau)$:

$$\begin{aligned} \lambda^{t+1}_{s,a_j} &= \lambda^{t}_{s,a_j} + \alpha(t)\left(\frac{\gamma(t)}{\alpha(t)}\right)\left(Q^t(s,a_j) - Q^t(s,a_{j-1})\right) \\ \Rightarrow \underbrace{\frac{\lambda^{t+1}_{s,a_j} - \lambda^{t}_{s,a_j}}{\alpha(t)}}_{\dot{\lambda}(\tau)} &= \left(\frac{\gamma(t)}{\alpha(t)}\right)\left(Q^t(s,a_j) - Q^t(s,a_{j-1})\right) \end{aligned} \tag{B.1}$$

then we see $\lim_{\tau\to\infty}\dot{\lambda}(\tau) \to 0$ because, by Assumption 3 (c), $\lim_{t\to\infty}\frac{\gamma(t)}{\alpha(t)} \to 0$ and, by Assumption 3 (A), $T(\infty) = \sum_{t=0}^{\infty}\alpha(t) \to \infty$. Therefore, $\lambda(\tau)$ can be seen as quasi-static w.r.t. $Q(\tau)$ at the fast time-scale. As a result, the updates in this time-scale correspond to standard Q-Learning for a fixed MDP defined by the value of $\lambda(\tau)$. Given Assumptions 1, 3 (A), and 2, this is known to converge to the optimal Q-values Q^*_λ for the given value of $\lambda(\tau)$[2].

Now, at the slow time-scale τ', we can repeat this continuous-time re-parameterization, except with $T'(t) = \sum_{m=0}^{t}\gamma(t)$. Then, re-arranging Equation 2.9 in a similar way as above, we get:

$$\begin{aligned} \underbrace{\frac{Q^{t+1}(s,a_j) - Q^t(s,a_j)}{\gamma(t)}}_{\dot{Q}(\tau')} = \left(\frac{\alpha(t)}{\gamma(t)}\right)&\left[r(s) - \lambda^t_{s,a_j}c_j - f(Q^t)\right. \\ &\left. + \max_{a'_j\in\{0,1\}} Q^t(s',a'_j)\right] - Q^t(s,a_j) \end{aligned}$$

Now, given that $\lim_{t\to\infty}\frac{\alpha(t)}{\gamma(t)} \to \infty$, and from the argument above about the Q-values converging in the fast time-scale, we can see the interpolated $\lambda(\tau')$ value as tracking the converged Q-values $Q^*_{\lambda(\tau')}$ (for that value of $\lambda(\tau')$). Then, we can write the ODE for $\lambda(\tau')$ as:

$$\dot{\lambda}(\tau') = Q^*_{\lambda(\tau')}(s,a_j) - Q^*_{\lambda(\tau')}(s,a_{j-1})$$

where $Q^*_{\lambda(\tau')}$ corresponds to the optimal Q-values corresponding to the given value of $\lambda(\tau')$.

Now, if $\lambda(\tau') < \lambda^*_{s,a_j}$ (the multi-action index for state s and action a_j), by the definition 2.4.2

of the multi-action index, we know that an action of weight c_j or higher is preferred. As a result, we see that $\dot{\lambda}(\tau') > 0$ in that case. If $\lambda(\tau') > \lambda^*_{s,a_j}$, the opposite is true and so $\dot{\lambda}(\tau') < 0$. Then, because $\lambda(0) = 0$ is bounded and given the step-sizes in Assumption 3 (B), $\lambda(\tau')$ converges to an equilibrium in which $Q^*_\lambda(s, a_j) - Q^*_\lambda(s, a) \to 0$.

Given that, by definition, $\lambda^*(s, a_j)$ is the value at which $Q^*_\lambda(s, a_j) = Q^*_\lambda(s, a_{j-1})$, $\lambda(\tau')$ converges to the multi-action index. □

This is a high-level proof, but the specific conditions for convergence can be seen in Lakshminarayanan & Bhatnagar [89]. They require 5 conditions: (1) Lipschitzness, (2) Bounded 'noise', (3) Properties about the relative step-sizes, (4) Convergence of fast time-scale, and (5) Convergence of slow time-scale.

Of these, (1)-(4) proceed in much the same way as in Avrachenkov & Borkar [12] because they do not depend on the multi-action extension of indexability. In addition, it is easy to show that the proof of (5) from Avrachenkov & Borkar [12] extends to the multi-action case which considers the limiting value of $Q(s, a_j) - Q(s, a_{j-1})$ rather than $Q(s, 1) - Q(s, 0)$. As a result, we refer the reader to Avrachenkov & Borkar [12] for the complete proof.

B.2 Reproducibility

Code is available at `https://github.com/killian-34/MAIQL_and_LPQL`. All the Q and λ values are initiated to zero in all the experiments. The parameter settings used for the two process type, random, and medication adherence data experiments are included in Tables B.1, B.2, and B.3, respectively. C is the multiplier for the size of the Q-value updates. C' is the multiplier for the size of the index value updates. "Rp/dream" is the number of replays per dream. "Rp T" is the replay period (replay every T steps). λ-bound is the upper bound (and negative of the lower bound) imposed on values of the indices for WIBQL and MAIQL during learning – placing these bounds sometimes

	WIBQL	QL-λ=0	MAIQL	M-Aprx	LPQL
C	0.1	0.2	0.1	0.4	0.4
C'	0.2	-	0.2	-	-
Rp/dream	NA	1000	1000	1000	NA
Rp T	1E+06	100	10	100	1E+06
λ-bound	3	-	3	3	3
D	500	500	500	500	500
ε_0	0.99	0.99	0.99	0.99	0.99
n_{lam}	-	-	-	3000	3000

Table B.1: Parameter settings for two process experiment.

helps prevent divergent behavior in early rounds when updates are large – $\lambda_{\max}$ is the upper bound value that an index could take, as defined by the problem parameters, i.e., $\frac{\max\{r\}}{\min\{C\}(1-\beta)}$ [81]. D is the divisor of the decaying ε-greedy function as well as the divisor of $\alpha(t)$ and $\gamma(t)$, the decaying functions defining the size of the updates of Q-values and index values, defined in the previous section. ε_0 is the multiplier for the ε-greedy function. n_{lam} is the number of points in λ-space used to approximate the Q(s, a, λ)-functions in LQPL and MAIQL-Aprx. All values were determined via manual tuning – empirically we found that most parameter settings led to similar long-term performance between algorithms, as long as the settings did not cause the algorithms to diverge. In the tables, M-Aprx stands for MAIQL-Aprx.

B.3 Algorithm Pseudocodes

See Algorithms B.3.1 and B.3.2 for the update and action selection steps of MAIQL and Algorithms B.3.3 and B.3.4 for the update and action selection steps of LPQL. The linear program for

	WIBQL	QL-λ=0	MAIQL	M-Aprx	LPQL
C	-	-	0.2	0.8	0.8
C'	-	-	0.4	-	-
Rp/dream	-	-	1000	NA	NA
Rp T	-	-	100	1E+06	1E+06
λ-bound	-	-	$\lambda_{\max}$	$\lambda_{\max}$	$\lambda_{\max}$
D	-	-	500	500	500
ε_0	-	-	0.99	0.99	0.99
n_{lam}	-	-	-	2000	2000

Table B.2: Parameter settings for random data experiment.

Oracle-LP-Index for a given current state $\boldsymbol{s}_{cur}$ and action a_k is given below:

$$
\begin{aligned}
&\min_{V^i(s^i,\lambda^i),\lambda^i} \sum_{i=0}^{N-1} \frac{\lambda^i B}{1-\beta} + \sum_{i=0}^{N-1} \mu^i(s^i) V^i(s^i,\lambda^i) \\
&\text{s.t. } V^i(s^i,\lambda^i) \geq r^i(s^i) - \lambda^i c_j + \beta \sum_{s^{i\prime}} T(s^i, a^i_j, s^{i\prime}) V^i(s^{i\prime},\lambda^i) \\
&\qquad\qquad \forall i \in \{0,...,N-1\}, \;\; \forall s^i \in \mathcal{S}, \;\; \forall a_j \in \mathcal{A} \\
&r^i(s^i_{cur}) - \lambda^i c_k + \beta \sum_{s^{i\prime}} T(s^i_{cur}, a^i_k, s^{i\prime}) V^i(s^{i\prime},\lambda^i) = \\
&\quad r^i(s^i_{cur}) - \lambda^i c_{k-1} + \beta \sum_{s^{i\prime}} T(s^i_{cur}, a^i_{k-1}, s^{i\prime}) V^i(s^{i\prime},\lambda^i) \\
&\qquad\qquad \forall i \in \{0,...,N-1\} \\
&\lambda^i \geq 0 \;\; \forall i \in \{0,...,N-1\}
\end{aligned}
\tag{B.2}
$$

The LP is similar to Eq. 2.11, but differs in two ways. First, instead of having a single λ value across all arms, each arm has its own independent λ^i value. Second, the second group of constraints is new, and forces the λ^i values to be set such that the planner would be indifferent between taking the ac-

	WIBQL	QL-λ=0	MAIQL	M-Aprx	LPQL
C	-	0.8	0.05	0.8	0.8
C'	-	-	0.1	-	-
Rp/dream	-	1000	1000	1000	1000
Rp T	-	10	5	5	5
λ-bound	-	-	$\lambda_{\max}$	$\lambda_{\max}$	$\lambda_{\max}$
D	-	1000	2000	1000	1000
ε_0	-	0.99	0.99	0.99	0.99
n_{lam}	-	-	-	2000	2000

Table B.3: Parameter settings for adherence data experiment.

tion in question a_k or the action that is one step cheaper a_{k-1}, which follows exactly the definition of the multi-action indexes. Note that although the indexes can each be computed independently, for convenience, we compute the index for a given a_k for each arm simultaneously to reduce overhead, as given in the above LP.

The ActionKnapsackILP referenced in Algorithm B.3.4 is the same as the modified knapsack given in Eq. A.8, reproduced below:

$$\begin{aligned}
&\max_{\boldsymbol{A}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \boldsymbol{A}_{ij} Q_{\boldsymbol{s},\lambda_{ind}}(i, a_j) \\
&\text{s.t.} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \boldsymbol{A}_{ij} c_j \leq B \\
&\sum_{j=0}^{M-1} \boldsymbol{A}_{ij} = 1 \quad \forall i \in 0, \ldots, N-1 \\
&\boldsymbol{A}_{ij} \in \{0, 1\}
\end{aligned} \tag{B.3}$$

where $Q_{\boldsymbol{s},\lambda_{ind}}(i, a_j)$ is the Q-function for each arm filtered to the current state of the arms, $\boldsymbol{s}$, and

minimizing value λ_{ind}, as given by the penultimate line of Algorithm B.3.4.

RandomAction, referenced in Algorithms B.3.2 and B.3.4, chooses random actions through the following iterative procedure: (1) randomly choose an arm with uniform probability, (2) randomly choose an action with probability inversely proportional to one plus its cost (must add one to avoid dividing by 0 for no-action). The procedure iterates until the budget is exhausted.

EpsilonGreedy(t), also referenced in Algorithms B.3.2 and B.3.4, draws a uniform random number between 0 and 1 and returns true if it is less than $\varepsilon_0 / \lceil \frac{t}{D} \rceil$ and false otherwise.

B.4 Medication Adherence Setting Details

We used the following procedure to estimate transition probabilities from the medication adherence data from Killian et al. [82]. First, we specify a history length of L. This gives a state space of size

Algorithm B.3.1: MAIQL Update

Data: $Q \in \mathbb{R}^{N \times |\mathcal{S}| \times (|\mathcal{A}|-1) \times |\mathcal{S}| \times |\mathcal{A}|}$, `// Need one copy of` $Q[s,a]$ `for each index on each arm`
$\lambda \in \mathbb{R}^{N \times |\mathcal{S}| \times (|\mathcal{A}|-1)}$, `// multi-action index estimates`
Batch, $\mathcal{C}$, `// Experience tuples, action costs`
$t, \nu(\cdot)$, `// iteration, state-action counter`
$\mathcal{S}, \mathcal{A}, N$ `// state space, action space, # of arms`
Hyperparameters: β, C, C', D **for** $(n, s, a, r, s') \in Batch$ **do**
  $\alpha = \frac{C}{\lceil \frac{\nu(s,a,n)}{D} \rceil}$
  for $i \in 0, \ldots, |\mathcal{S}|$ **do**
    for $j \in 1, \ldots, |\mathcal{A}|$ **do**
      $Q[n,i,j,s,a] \mathrel{+}=$
      $\alpha(r - \mathcal{C}[a] * \lambda[i,j] + \beta * \max\{Q[n,i,j,s']\} - Q[n,i,j,s,a])$
  if $a \neq 0$ & $t \pmod N == 0$ **then**
    $\gamma = \frac{C'}{1 + \lceil \frac{\nu(s,a,n) \log \nu(s,a,n)}{D} \rceil}$
    $\lambda[s,a] \mathrel{+}= \frac{\gamma(Q[n,s,a,s,a]) - Q[n,s,a,s,a-1])}{\mathcal{C}[a] - \mathcal{C}[a-1]}$
return Q, λ

Algorithm B.3.2: MAIQL Action Select

```
Data: λ ∈ ℝ^{N×|S|×(|A|−1)},                    // multi-action index estimates
 s ∈ ℝ^N                                         // current state of all arms
 t, N, B                                         // current iteration, # of arms, budget
if EpsilonGreedy(t) then
    return RandomAction()
else
    a = [0 for _ in range(N)]
    λ_f = FilterCurrentState(λ, s)                // λ_f ∈ ℝ^{N×(|A|−1)}
    for i ∈ 0 ... B do
        i = arg max(λ_f[a + 1] − λ_f[a])          // a is a vector index, arg max
          ignores out of bounds indexes
        a[i] += 1
return a
```

2^L for each arm. Then, for each patient in the data, we count all of the occurrences of each state transition across a treatment regimen of 6 months (168 days). If L was small (e.g., 1 or 2), we could take a frequentist approach and simply normalize these counts appropriately to get valid transition probabilities to sample for experiments. However, as the history length L gets larger, the number of non-zero entries in the count data for state transitions become large. We take two steps to account for this sparsity. (1) We run K-means clustering over all patients, using the count data as features, then combine the counts for all patients within a cluster. Intuitively, the larger the K, the more "peaks" of the distribution of patient adherence modes we will try to approximate, but the fewer data points are available to estimate the distribution in each cluster — however, it may be desirable to have more clusters to allow for some samples to come from uncommon but "diverse" modes that may be challenging to plan for. We set K to 10. (2) We then take a Bayesian approach, rather than a frequentist approach for sampling patients/processes from the clustered counts data. That is, we treat the counts as priors of a beta distribution, then sample transition probabilities from those distributions according to the priors. Finally, to simulate action effects, since actions were not recorded

Algorithm B.3.3: LPQL Update

Data: $Q \in \mathbb{R}^{N \times n_{lam} |\mathcal{S}| \times |\mathcal{A}|}$, `// Need one copy of` $Q[s,a]$ `for each of the` n_{lam} `test points on each arm`
Batch, $\mathcal{C}$, `// Experience tuples, action costs`
$\lambda_{\max}$, `// Max` λ `at which to estimate` Q
n_{lam}, `// # of` λ `points at which to estimate` Q
$\nu(\cdot)$ `// state-action counter`
Hyperparameters: β, C, D **for** $(n, s, a, r, s') \in Batch$ **do**
  $\alpha = \frac{C}{\lceil \frac{\nu(s,a,n)}{D} \rceil}$
  for $i \in 0, \ldots, n_{lam}$ **do**
    $\lambda_p = \frac{i * \lambda_{\max}}{n_{lam}}$
    $Q[n, i, s, a] \mathrel{+}= \alpha(r - \mathcal{C}[a] * \lambda_p + \beta * \max\{Q[n, i, s']\} - Q[n, i, s, a])$
return Q

in the available adherence data, we scale the priors multiplicatively according to the index of the action, i.e., larger/more expensive actions increase the priors associated with moving to the adhering state.

In summary, to get a transition function for a single simulated arm in the medication adherence experimental setting, we do the following. First, randomly choose a cluster, with probability weighted by the number of patients in the cluster. Then, build up a transition matrix by sampling each row according to its own beta distribution with priors given by the counts data (i.e., actual observations of $s \to s'$ transitions), scaled by the action effects.

This process was desirable for producing simulated arms with transition functions tailored to resemble that of a real world dataset, while allowing for some randomness via the sampling procedure, as well as a straightforward way to impose simulated action effects. However, one downside of this approach is that, since each row of the transition matrix is sampled independently, this may produce simulated arms whose probability of adherence changes in a non-smooth manner as a function of history. For example, in the real-world, we would expect that $P(0111 \to 1111)$ is correlated with $P(1011 \to 0111)$ and that $P(0000 \to 0000)$ is correlated with $P(1000 \to 0000)$, but our procedure

would not necessarily enforce these relationships if there were not sufficient occurrences of each transition in the counts data.

The python code used to execute this procedure is included in the repository at https://github.com/killian-34/MAIQL_and_LPQL.

Algorithm B.3.4: LPQL Action Select

Data: $Q \in \mathbb{R}^{N \times n_{lam} |\mathcal{S}| \times |\mathcal{A}|}$, `// Q-functions for each of the` n_{lam} `test points on each arm`

$\boldsymbol{s} \in \mathbb{R}^N$ `// current state of all arms`

$\lambda_{\max}$, `// Max` λ `at which` Q `is estimated`

n_{lam}, `// # of` λ `points at which` Q `is estimated`

t, β `// iteration, discount factor`

$N, \mathcal{C}, B$ `// # of arms, action costs, budget`

if $\textit{EpsilonGreedy}(t)$ **then**

  return $\textsc{RandomAction}()$

$\boldsymbol{a} = [0 \text{ for \_ in range } (N)]$

$Q_f = \textsc{FilterCurrentState}(Q, \boldsymbol{s})$ `//` $Q_f \in \mathbb{R}^{N \times n_{lam} \times |\mathcal{A}|}$

$\lambda_{ind} = -1$

`/* The min of Eq. 2.11 occurs at the point where the negative sum of slopes of all` $V^i = \max\{Q^i_\lambda\}$ `is` $\leq B/(1-\beta)$`, so we will iterate through our estimates of` Q^i_λ `and stop our search at the first point where that is true. */`

for $i \in 0, \ldots, n_{lam}$ **do**

  $\lambda^0_p = \frac{i * \lambda_{\max}}{n_{lam}}$

  $\lambda^1_p = \frac{(i+1) * \lambda_{\max}}{n_{lam}}$

  $m_V = \frac{\max_a\{Q_f[:, i+1]\} - \max_a\{Q_f[:, i]\}}{\lambda^1_p - \lambda^0_p}$ `//` $m_V \in \mathbb{R}^N$

  if $\sum_n \{m_V\} \geq -\frac{B}{1-\beta}$ **then**

    $\lambda_{ind} = i$

    break

$\boldsymbol{a} = \textsc{ActionKnapsackILP}(Q_f[:, \lambda_{ind}, :], \mathcal{C}, B)$

return $\boldsymbol{a}$

C

Appendix to Chapter 4

C.1 Proof of Indexability

We give the proof assuming forward threshold policies are optimal, and note where relevant how the proof also works for reverse threshold optimal policies.

Fact 1. *For two non-concurrent, increasing, linear functions $f_1(m)$ and $f_2(m)$ and two points m_1, m_2, such that $m_1 \leq m_2$, if $f_1(m_1) \leq f_2(m_1)$ and $f_1(m_2) \geq f_2(m_2)$, then $\frac{df_1}{dm} \geq \frac{df_2}{dm}$. Additionally, if*

$f_1(m_1) < f_2(m_1)$ *and* $f_1(m_2) \geq f_2(m_2)$, *then* $\frac{df_1}{dm} > \frac{df_2}{dm}$.

Proof. We now start proving the theorem by assuming that forward belief threshold policies are optimal. Let $b^*_{th}(m)$ denote the threshold corresponding to the optimal threshold policy for a given m. To show indexability, we must show that if a belief state b is passive, i.e., $b > b^*_{th}(m_1)$, for some m_1, then it is also passive, i.e., $b > b^*_{th}(m_2)$, for all $m_2 \geq m_1$.

In our problem, we have $2T$ belief states which, for a forward threshold policy, can be arranged in a descending order of their belief values: $\mathcal{B} := \{b_{2T}, b_{2T-1}, \ldots, b_i, \ldots, b_1\}$.* A forward threshold policy is then any real value b_{th} which splits $\mathcal{B}$ into a passive set $\mathcal{P} = \{b_i : b_i > b_{th} \, \forall b_i \in \mathcal{B}\}$ and active set $\mathcal{C} = \{b_i : b_{th} \geq b_i \, \forall b_i \in \mathcal{B}\}$. Note that all values of b_{th} such that $b_{i+1} \geq b_{th} > b_i \, \forall i \in 1, \ldots, 2T$ correspond to the same threshold policy. Thus there are only $2T + 1$ unique threshold policies possible corresponding to the $2T + 1$ such belief regions marked by points in $\mathcal{B}$. Let $\Pi = \{\pi_{2T+1}, \pi_{2T}, \ldots, \pi_1\}$ denote these unique possible threshold policies arranged in a decreasing order, where $\pi_i \geq \pi_j$ implies $b^*_{th}(\pi_i) \geq b^*_{th}(\pi_j)$ where $b^*_{th}(\pi_i)$ is the optimal belief threshold associated with π_i.† Thus the threshold policy π_i would follow: $b_i > b^*_{th}(\pi_i) \geq b_{i-1}$ $\forall i \in 1, \ldots, 2T+1$, where $b_0 := -\infty$ and $b_{2T+1} := \infty$. Note that in a policy π_i, if for a belief state b, the optimal action is passive, then under a policy π_j, the optimal action at b is also passive $\forall j \leq i$ because $b^*_{th}(\pi_i) \geq b^*_{th}(\pi_j)$. Thus to prove indexability, it is sufficient to show that:

$$\begin{aligned} &\forall m_1, m_2 \text{ such that } m_1 \leq m_2, \\ &\text{if } \pi^*(m_1) = \pi_i \text{ and } \pi^*(m_2) = \pi_j, \text{ then} \\ &\implies i \geq j \end{aligned} \tag{C.1}$$

*For simplicity, this assumes the starting belief is equal to the belief at the head of one of the chains, i.e., $P^a_{1,1}$ or $P^a_{0,1}$. However, we could add to the set $\mathcal{B}$ another T belief states corresponding to a chain that starts from any arbitrary belief and evolves for T passive actions. These new states could be ordered appropriately within $\mathcal{B}$ and the rest of the proof would follow unchanged.

†For reverse threshold optimal processes, simply arrange $\mathcal{B}$ and Π in ascending order of belief. The rest of the proof follows similarly.

where $\pi^*(m)$ denotes the optimal threshold policy at subsidy m.

Lemma C.1.0.1. Let m_i^* be the *infimum* among all m's for which $\pi^*(m) = \pi_i$. Then, the infimum is achievable (i.e., $\pi^*(m_i^*) = \pi_i$) and moreover $m_{2T+1}^* < m_{2T}^* < \ldots < m_1^*$.

Proof. We prove this using induction. Consider the base case: $m_{2T+1}^* < m_i^* \; \forall \, i < 2T+1$. When $m \to -\infty$, the optimal action would clearly always be to act to avoid accruing large negative reward. So π_{2T+1} would be the optimal policy for $m \to -\infty$ and clearly the base case is true.

For the inductive case, assume the hypothesis, $m_{2T+1}^* < \ldots < m_{t+1}^* < m_i^* \; \forall i < t+1$. Let m_t^* be the *infimum* among all m's for which $\pi^*(m) = \pi_t$. We must show: (1) $m_t^* < m_i^* \; \forall i < t$; (2) $\pi^*(m_t^*) = \pi_t$ (i.e., the infimum is achievable). For convenience, we denote $L = \{\pi_t, \pi_{t-1}, \ldots \pi_1\}$ as the set of "lower-side" polices and $U = \{\pi_{2T+1}, \pi_{2T}, \ldots \pi_{t+1}\}$ as the set of "upper-side" policies.

As m is increased beyond m_{t+1}^*, let m' be the *infimum value* among all m's whose optimal policy is from $L = \{\pi_t, \pi_{t-1}, \ldots \pi_1\}$ (note, the definition of m' is different from m_t^* since at this point we do not know whether the smallest m's optimal policy is π_t or some π_i with $i < t$ yet). That is, either the optimal threshold policy at m' is from L (when the infimum is achievable) or there exists an infinite sequence $\{\bar{m}_l\}_{l=1}^{\infty}$ that converges *from the right side* to m' (i.e., $\bar{m}_l \geq m'$ for all s) and the optimal policy for any $\bar{m}_l$ is from policy set L (when the infimum is not achievable). For notational convenience, we will think of the former achievable case also as that there is a sequence $\{\bar{m}_l\}_{l=1}^{\infty}$ that converges to m' and the optimal policy for any $\bar{m}_l$ is from L (letting all $\bar{m}_l = m'$ will do). In fact, a stronger conclusion holds. That is, we can choose an infinite-length sequence $\{\bar{m}_l\}_{l=1}^{\infty}$ such that the optimal policy for each $\bar{m}_l$ will be the same. This simply follows from the fact that $\{\bar{m}_l\}_{l=1}^{\infty}$ has infinite length, and their optimal policy is from a finite set L. So some policy from L must be optimal for infinitely many of $\bar{m}_l$'s. Therefore, we shall assume that $\bar{m}_l \to m'$ from the right side and the optimal policy for each $\bar{m}_l$ is some $\bar{\pi} \in L$.

Our main claim is that for subsidy m', the passive action and active action must both be optimal

at state b_t. Therefore, by definition, this implies the threshold policy π_t is optimal for m'. We thus have $m_t^* = m'$, $m_i^* > m_t^* \, \forall i < t$, and moreover π_t is indeed optimal for m_t^* (i.e., the infimum is achievable). This concludes the induction proof. The remainder of this proof will be devoted to prove this claim.

By definition of m', there exists a sequence $\{\underline{m}_u\}_{u=1}^{\infty}$ that converges to m' *from the left side* (i.e., $\underline{m}_u < m'$ for all t) and moreover the optimal policy for any $\underline{m}_u$ is from the policy set $U = \{\pi_{2T+1}, \pi_{2T}, \ldots \pi_{t+1}\}$. Similar to the above reasoning, we shall choose the sequence $\{\underline{m}_u\}_{u=1}^{\infty}$ such that their optimal policy is the same $\underline{\pi} \in U$.

We now prove that the passive action and active action must both be optimal at state b_t for m'. Assume, for the sake of contradiction, that the optimal action at b_t for subsidy m' is passive and that the active action is not optimal (the other case where the optimal action is active follows a similar contradiction argument). That means the optimal policy for m' has a threshold $b_{th}^*(m') < b_t$ and thus $\pi^*(m') \in L$. Moreover, since the active action is not optimal for b_t, $\underline{\pi}$ must not be optimal for m' and thus achieves strictly less reward than $\pi^*(m')$. Since $\underline{m}_u \to m'$, we thus have

$$\lim_{u \to \infty} V_{\underline{m}_u}(\underline{\pi}) = V_{m'}(\underline{\pi}) < V_{m'}(\pi(m')),$$

where the last inequality uses the fact that $\underline{\pi}$ is sub-optimal for m' because the active action is strictly sub-optimal for b_t. On the other hand,

$$V_{m'}(\pi(m')) = \lim_{u \to \infty} V_{\underline{m}_u}(\pi(m')) \leq \lim_{u \to \infty} V_{\underline{m}_u}(\underline{\pi})$$

These two inequalities contradict each other. This concludes our proof of the lemma. □

Let π_i be the optimal policy at some m_1.

$$\implies m_i^* \leq m_1$$

$$\implies m_j^* < m_i^* \leq m_1 \; \forall j > i \text{ using Lemma C.1.0.1}$$

Let $V_\pi(m, b)$ be the discounted reward of policy π at arbitrary state b as defined in Eq. 4.2. Then for any $V_{\pi_i}(m, b)$ and $V_{\pi_j}(m, b)$ such that $j > i$ we have:

$$V_{\pi_i}(m_j^*, b) < V_{\pi_j}(m_j^*, b) \; (\pi_j \text{ is optimal at } m_j^*) \tag{C.2}$$

$$V_{\pi_i}(m_i^*, b) \geq V_{\pi_j}(m_i^*, b) \; (\pi_i \text{ is optimal at } m_i^*) \tag{C.3}$$

$$m_j^* < m_i^* \text{ if } j > i \tag{C.4}$$

$$\implies \frac{dV_{\pi_i}}{dm} > \frac{dV_{\pi_j}}{dm} \forall j > i \tag{C.5}$$

Where Eq. C.2 is a strict inequality as implied by Lemma C.1.0.1 and Eq. C.5 follows from Fact 1 and the value function's linear dependence on m (whether discounted or average reward criterion). We now claim that $\forall m_j > m_i^*$, if π_j is optimal for m_j then we must have $j \leq i$. Towards a contradiction, assume $j > i$. Then similar to the above equations, we have the following:

$$V_{\pi_i}(m_j, b) \leq V_{\pi_j}(m_j, b) \; (\pi_j \text{ is optimal at } m_j) \tag{C.6}$$

$$V_{\pi_i}(m_i^*, b) \geq V_{\pi_j}(m_i^*, b) \; (\pi_i \text{ is optimal at } m_i^*) \tag{C.7}$$

$$m_i^* < m_j \tag{C.8}$$

$$\implies \frac{dV_{\pi_i}}{dm} \leq \frac{dV_{\pi_j}}{dm} \forall j > i \tag{C.9}$$

Where Eq. C.9 follows from Fact 1 and the value function's linear dependence on m (whether dis-

counted or average reward criterion). which contradicts Eq. C.5. Therefore, our claim holds. From C.1, that implies indexability. □

C.2 Technical Condition for Forward Threshold Policies to be Optimal

We restate Eq. 4.2 here:

$$V_m(b) = max \begin{cases} m + b + \beta V_m(\tau(b)) & \text{passive} \\ b + \beta(bV_m(P^a_{1,1}) + (1-b)V_m(P^a_{0,1})) & \text{active} \end{cases}$$

where $\tau(b) \coloneqq \tau_1(b)$ from Eq. 4.1. Simplified, $\tau(b)$ is simply a linear function of b given by the expression

$$\begin{aligned} \tau(b) &= bP^p_{1,1} + (1-b)P^p_{0,1} \\ &= (P^p_{1,1} - P^p_{0,1})b + P^p_{0,1} \end{aligned} \tag{C.10}$$

We will start by stating two facts, then proving three useful technical lemmas.

Fact 2. $\frac{d(\tau(b))}{db} = (P^p_{1,1} - P^p_{0,1}) \leq 1$.

Fact 3. $\forall b, b'$ s.t. $b \geq b'$, $\tau(b) \geq \tau(b')$.

Facts 2 and 3 follow from Eq C.10.

Lemma C.2.0.1. $V_m(b_1) - V_m(b_2) \geq b_1 - b_2, \forall b_1, b_2$ s.t. $b_1 > b_2$

Proof. We will proceed via induction, where the base case will be a one-step value function. Then we will show that the t-step value function assumption implies the t+1-step inductive value function hypothesis. In the base case the value function equals only the one-step immediate reward. It is sufficient to compare the value functions $V^1_m(b_1)$ and $V^1_m(b_2)$ element-wise, since if the true

optimal action for one of the value functions is passive and the other active, the bound can still be established by flipping the action of one of the value functions as needed. This gives:

Base case $V^1_m(b_1) - V^1_m(b_2) =$

$$m + b_1 - (m + b_2) = b_1 - b_2 \qquad \text{passive} \tag{C.11}$$

$$b_1 - b_2 = b_1 - b_2 \qquad \text{active} \tag{C.12}$$

is clearly satisfied. Now assume $V^t_m(b_1) - V^t_m(b_2)) \geq b_1 - b_2$. Then $V^{t+1}_m(b_1) - V^{t+1}_m(b_2)$

Case 1 (both passive):

$$\begin{aligned}
&= m + b_1 + \beta V^t_m(\tau(b_1)) - (m + b_2 + \beta V^t_m(\tau(b_2))) \\
&= b_1 - b_2 + \beta\Big(V^t_m(\tau(b_1)) - V^t_m(\tau(b_2))\Big) \\
&\geq b_1 - b_2 + \beta(\tau(b_1) - \tau(b_2)) \\
&\geq b_1 - b_2
\end{aligned} \tag{C.13}$$

Case 2 (both active):

$$\begin{aligned}
&= b_1 - b_2 + \beta\Big((b_1 - b_2)V^t_m(P^a_{1,1}) + (b_2 - b_1)V^t_m(P^a_{0,1})\Big) \\
&= b_1 - b_2 + \beta\Big((b_1 - b_2)(V^t_m(P^a_{1,1}) - V^t_m(P^a_{0,1}))\Big) \\
&= (b_1 - b_2)(1 + \beta(V^t_m(P^a_{1,1}) - V^t_m(P^a_{0,1})) \\
&\geq (b_1 - b_2)(1 + \beta * 0) \\
&= (b_1 - b_2)
\end{aligned} \tag{C.14}$$

□

Corollary 1. $V_m(b)$ is an increasing function in b, i.e., $V_m(b) \geq V_m(b'), \forall b, b'$ s.t. $b \geq b'$.

Proof. The proof follows from Lemma C.2.0.1 by setting $b_1 = b$ and $b_2 = b'$. □

Lemma C.2.0.2. $V_m(b_1) - V_m(b_2) \leq \frac{b_1 - b_2}{1-\beta}, \forall b_1, b_2$ s.t. $b_1 > b_2$

Proof. Proceed by induction again. The base case $V_m(b_1) - V_m(b_2) =$

$$m + b_1 - (m + b_2) = b_1 - b_2 \leq \frac{b_1 - b_2}{1-\beta} \qquad \text{both passive} \tag{C.15}$$

$$b_1 - b_2 = b_1 - b_2 \leq \frac{b_1 - b_2}{1-\beta} \qquad \text{both active} \tag{C.16}$$

which are both clearly satisfied. Now assume $V^t_m(b_1) - V^t_m(b_2) \leq \frac{b_1 - b_2}{1-\beta}$. Then, $V^{t+1}_m(b_1) - V^{t+1}_m(b_2)$

Case 1 (both passive):

$$\begin{aligned}
&= \big(m + b_1 + \beta V^t_m(\tau(b_1))\big) - \big(m + b_2 + \beta V^t_m(\tau(b_2))\big) \\
&= (b_1 - b_2) + \beta\big(V^t_m(\tau(b_1)) - V^t_m(\tau(b_2))\big) \\
&\leq (b_1 - b_2) + \beta\left(\frac{\tau(b_1) - \tau(b_2)}{1-\beta}\right) \\
&\leq (b_1 - b_2) + \beta\left(\frac{(b_1 - b_2)}{1-\beta}\right) \text{ by Fact 3} \\
&= \frac{b_1 - b_2}{1-\beta}
\end{aligned} \tag{C.17}$$

Case 2 (both active):

$$
\begin{aligned}
&= \Big(b_1 + \beta\big(b_1 V^t_m(P^a_{1,1}) + (1-b_1)V^t_m(P^a_{0,1})\big)\Big) - \\
&\Big(b_2 + \beta\big(b_2 V^t_m(P^a_{1,1}) + (1-b_2)V^t_m(P^a_{0,1})\big)\Big) \\
&= (b_1 - b_2) + \beta\Big(\big(b_1 - b_2\big)\big(V^t_m(P^a_{1,1}) - V^t_m(P^a_{0,1})\big)\Big) \\
&\leq (b_1 - b_2) + \beta\left((b_1 - b_2).\frac{P^a_{1,1} - P^a_{0,1}}{1-\beta}\right) \\
&\leq (b_1 - b_2) + \beta\left(\frac{(b_1 - b_2)}{1-\beta}\right) \text{ by Fact 2} \\
&= \frac{b_1 - b_2}{1-\beta}
\end{aligned}
\tag{C.18}
$$

□

Lemma C.2.0.3. $\frac{d(V_m(b))}{db} \geq 1 + \beta\alpha$

where, $\alpha = \min\{P^p_{1,1} - P^p_{0,1}, P^a_{1,1} - P^a_{0,1}\}$

Proof. Using Eq. 4.2, we get:

$$
\frac{d(V_m(b))}{db} = \begin{cases} 1 + \beta\frac{d(V_m(\tau(b)))}{d(\tau(b))}\frac{d(\tau(b))}{db} & \text{passive} \\ 1 + \beta(V_m(P^a_{1,1}) - V_m(P^a_{0,1})) & \text{active} \end{cases}
\tag{C.19}
$$

Case 1 (passive):

$$= 1 + \beta \frac{d(V_m(\tau(b)))}{d(\tau(b))}(P^p_{1,1} - P^p_{0,1}) \tag{C.20}$$

$$= 1 + \beta \lim_{\delta \to 0} \frac{V_m(\tau(b) + \delta) - V_m(\tau(b))}{\tau(b) + \delta - \tau(b)}(P^p_{1,1} - P^p_{0,1}) \tag{C.21}$$

$$\geq 1 + \beta(P^p_{1,1} - P^p_{0,1}) \text{ by Lemma C.2.0.1} \tag{C.22}$$

$$\geq 1 + \beta\alpha \tag{C.23}$$

Case 2 (active):

$$= 1 + \beta(V_m(P^a_{1,1}) - V_m(P^a_{0,1})) \tag{C.24}$$

$$\geq 1 + \beta(P^a_{1,1} - P^a_{0,1}) \text{ by Lemma C.2.0.1} \tag{C.25}$$

$$\geq 1 + \beta\alpha \tag{C.26}$$

□

Now we derive the technical condition for **Theorem 4.4.5**. In this case, proving that threshold policies are optimal is equivalent to proving that, if it is optimal to act now, then it is optimal to act for all later beliefs. Formally, if for a belief b, the optimal action is to act, then we must show that for a lower $b' < b$, the optimal action is also to act. To do this, we show that Theorem 4.4.5 implies that the derivative wrt b of the passive action value function is greater than the derivative wrt b of the active action value function:

$$(P^p_{1,1} - P^p_{0,1})(1 + \beta(P^a_{1,1} - P^a_{0,1}))(1 - \beta) \geq P^a_{1,1} - P^a_{0,1} \tag{C.27}$$

Note that since $(P^a_{1,1} - P^a_{0,1}) \leq 1, \implies (1+\beta(P^a_{1,1} - P^a_{0,1}))(1-\beta) \leq 1$, Eq.C.27 itself implies that $\alpha = P^a_{1,1} - P^a_{0,1}$. Thus, it becomes:

$$(P^p_{1,1} - P^p_{0,1})(1+\beta\alpha)(1-\beta) \geq P^a_{1,1} - P^a_{0,1} \tag{C.28}$$

$$\implies (P^p_{1,1} - P^p_{0,1})(1+\beta\alpha) \geq V_m(P^a_{1,1}) - V_m(P^a_{0,1}) \text{ by Lemma C.2.0.2} \tag{C.29}$$

$$\implies (P^p_{1,1} - P^p_{0,1})\frac{d(V_m(b))}{db} \geq V_m(P^a_{1,1}) - V_m(P^a_{0,1}) \text{ by Lemma. C.2.0.3} \tag{C.30}$$

$$\implies 1+\beta\frac{d(V_m(\tau(b)))}{d(\tau b)}\frac{d(\tau(b))}{db} \geq 1+\beta(V_m(P^a_{1,1}) - V_m(P^a_{0,1})) \text{ by Fact 2} \tag{C.31}$$

$$\implies \frac{d(V_m(b|a=0))}{d(b)} \geq \frac{d(V_m(b|a=1))}{d(b)} \tag{C.32}$$

$$\tag{C.33}$$

C.3 Technical Condition for Reverse Threshold Policies to be Optimal

Now we derive a technical condition for a reverse threshold policy. That is, a threshold policy in which if it is optimal to be passive in the current state, then it must also be optimal to act in all later states in the order. First we prove one more technical Lemma.

Lemma C.3.0.1. $\frac{d(V_m(b))}{db} \leq 1 + \frac{\beta\gamma}{1-\beta}$
where, $\gamma = \max\{P^p_{1,1} - P^p_{0,1}, P^a_{1,1} - P^a_{0,1}\}$

Proof. Using Equation C.2, we get:

$$\frac{d(V_m(b))}{db} = \begin{cases} 1+\beta\frac{d(V_m(\tau(b)))}{d(\tau(b))}\frac{d(\tau(b))}{db} & \text{passive} \\ 1+\beta(V_m(P^a_{1,1}) - V_m(P^a_{0,1})) & \text{active} \end{cases} \tag{C.34}$$

Case 1 (passive):

$$= 1 + \beta \frac{d(V_m(\tau(b)))}{d(\tau(b))}(P^p_{1,1} - P^p_{0,1}) \tag{C.35}$$

$$= 1 + \beta \lim_{\delta \to 0} \frac{V_m(\tau(b) + \delta) - V_m(\tau(b))}{\tau(b) + \delta - \tau(b)}(P^p_{1,1} - P^p_{0,1}) \tag{C.36}$$

$$\leq 1 + \frac{\beta}{1-\beta}(P^p_{1,1} - P^p_{0,1}) \text{ by Lemma C.2.0.2} \tag{C.37}$$

$$\leq 1 + \frac{\beta\gamma}{1-\beta} \tag{C.38}$$

Case 2 (active):

$$= 1 + \beta(V_m(P^a_{1,1}) - V_m(P^a_{0,1})) \tag{C.39}$$

$$\leq 1 + \frac{\beta}{1-\beta}(P^a_{1,1} - P^a_{0,1}) \text{ by Lemma C.2.0.2} \tag{C.40}$$

$$\leq 1 + \frac{\beta\gamma}{1-\beta} \tag{C.41}$$

□

Now to give a condition under which reverse threshold policies are optimal. Formally, if for a belief b, the optimal action is to be passive, then we must show that for a lower $b' < b$, the optimal action is also to be passive. We do this by showing that the Theorem 4.4.6 statement implies that the derivative wrt b of the passive value function is less than the derivative wrt b of the active action value function:

$$(P^p_{1,1} - P^p_{0,1})(1 + \frac{\beta(P^a_{1,1} - P^a_{0,1})}{1-\beta}) \leq P^a_{1,1} - P^a_{0,1} \tag{C.42}$$

Note that the Eq. C.42 itself implies that $\gamma = P^a_{1,1} - P^a_{0,1}$, thus giving:

$$(P^p_{1,1} - P^p_{0,1})(1 + \frac{\beta\gamma}{1-\beta}) \leq P^a_{1,1} - P^a_{0,1} \tag{C.43}$$

$$\implies (P^p_{1,1} - P^p_{0,1})(1 + \frac{\beta\gamma}{1-\beta}) \leq V_m(P^a_{1,1}) - V_m(P^a_{0,1}) \text{ by Lemma C.2.0.1} \tag{C.44}$$

$$\implies (P^p_{1,1} - P^p_{0,1})\frac{d(V_m(b))}{db} \leq V_m(P^a_{1,1}) - V_m(P^a_{0,1}) \text{ by Lemma C.3.0.1} \tag{C.45}$$

$$\implies 1 + \beta\frac{d(V_m(\tau(b)))}{d(\tau b)}\frac{d(\tau(b))}{db} \leq 1 + \beta(V_m(P^a_{1,1}) - V_m(P^a_{0,1})) \text{ by Fact 2} \tag{C.46}$$

$$\implies \frac{d(V_m(b|a=0))}{d(b)} \leq \frac{d(V_m(b|a=1))}{d(b)} \tag{C.47}$$

$$\tag{C.48}$$

C.4 Threshold Conditions for Average Reward Case

First we define the concept of *value boundedness*[42]:

Definition C.4.1 (Value Boundedness)**.** *For a given MDP, consider a value function $V_\beta(b)$, states $b \in \mathcal{B}$ and some index state $z \in \mathcal{B}$. Then an MDP is value bounded if for a constant M and function $M(b)$:*

$$M(b) < V_\beta(b) - V_\beta(z) < M \tag{C.49}$$

We now prove that Thm. 4.4.5 and Thm. 4.4.6 hold respectively under the average reward criterion as $\beta \to 1$ using Dutta's Theorem as follows[42]. Consider an MDP that is *value bounded*. Let $\pi_\beta(\cdot)$ be a stationary optimal policy for the discounted MDP. (1) Suppose $\pi_\beta(\cdot) \to \pi$ pointwise, as $\beta \to 1$. Then π is a stationary optimal policy for the average reward criterion. (2) Furthermore, given state ordering O, if for all discounted optimal policies $\pi_\beta(b)$, $O(b') \geq O(b)$ implies $\pi_\beta(b') \geq \pi_\beta(b)$ (i.e., threshold policies are optimal), then any sequence of discounted optimal

policies converge to an average optimal policy as $\beta \to 1$.

(2) and (1) together imply that any MDP that admits threshold optimal policies under discounted reward criteria also admits threshold optimal policies under average reward criteria. By construction, any MDP that satisfies Thm. 4.4.5 or Thm. 4.4.6 admits threshold optimal policies under the discounted reward criterion. Therefore, to prove that those conditions hold under the average reward criterion as $\beta \to 1$, we need only prove that any CoB is value bounded.

Theorem C.4.2. *Any Collapsing Bandit is value bounded.*

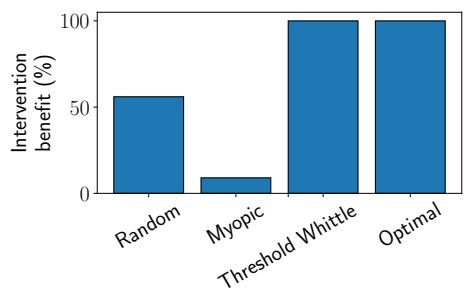
C.5 Example When the Myopic Policy Fails

We present an example in which the myopic baseline is barely better than No Calls, while Threshold Whittle is *optimal*. Consider the system with $N = 2$ and $k = 1$ and the transition probabilities shown in Fig. C.1a.

$$P^{p,1} = \begin{bmatrix} 0.97 & 0.03 \\ 0.03 & 0.97 \end{bmatrix} \quad P^{a,1} = \begin{bmatrix} 0.96 & 0.04 \\ 0.01 & 0.99 \end{bmatrix}$$

$$P^{p,2} = \begin{bmatrix} 0.25 & 0.75 \\ 0.03 & 0.97 \end{bmatrix} \quad P^{a,2} = \begin{bmatrix} 0.23 & 0.77 \\ 0.01 & 0.99 \end{bmatrix}$$

(a)



(b)

Figure C.1: For the example transition matrices, Myopic performs worse than random, while Threshold Whittle is nearly optimal.

Fig. C.1b shows how various policies perform on these two processes. The myopic policy is worse than random and threshold Whittle is nearly optimal. The myopic policy always acts on process 2 because the immediate reward it considers, $(b_{t+1}|a = 1) - (b_{t+1}|a = 0)$ is marginally higher for process 2 than process 1. However, process 1 is better to pull in the long run because process 2 has a

large $P^p_{0,1}$, making it self-correcting, meaning the process is likely to become adhering quickly even without an intervention. However, process 1 has a very small $P^a_{0,1}$ and $P^p_{0,1}$ and is thus difficult to revive from the bad state even with an intervention, making it important to keep intervening to stop the process from ever entering the bad state.

The following analysis shows that the myopic policy always prefers to pull arm 2:

For process 1:

$$(b_{t+1}|a=0) = 0.97.b_t + 0.03.(1-b_t) \qquad = 0.94.b_t + 0.03$$

$$(b_{t+1}|a=1) = 0.99.b_t + 0.01.(1-b_t) \qquad = 0.95.b_t + 0.04$$

$$\text{Thus, } \Delta b_t = (b_{t+1}|a=1) - (b_{t+1}|a=0) \qquad = 0.01 + 0.01.b_t < 0.02$$

Similarly, for process 2:

$$\Delta b_t = 0.02$$

The myopic policy chooses the arm with the greater Δb_t.

C.6 Learning Online

So far we assumed that *all transition probabilities are known.* However, in a real deployment, the transition probabilities of processes would be unknown at the start, and it would be desirable to learn the transition probabilities online in tandem with planning. To develop an online planning regime for our algorithm, we use the tuberculosis medication adherence monitoring domain as a case study and motivating example.

We implement a Thompson sampling-based learning method[152], which is a heuristic which has been shown to work well in practice and has been frequently used in the bandit literature[72]. In

Thompson sampling, we sample from a posterior distribution over the estimated parameters and use the samples for planning. This allows for "sub-optimal" actions to be taken periodically, building exploration implicitly into planning. Then, as arms are pulled we use the observations to update our posterior distribution. We maintain a Beta distribution posterior over the parameters of each row of a patient's transition matrix and sample from it each day to generate a matrix with which the system can plan for that round.

Additionally, we consider two specific features of the TB medication adherence monitoring domain that can be used to accelerate learning with Thompson sampling. First, it is reasonable to assume that patients (processes) might remember some number of previous days of their medication adherence behavior. Thus, when the agent pulls an arm, the arm may reveal state observations for some number of previous days which we call *buffer length*. The larger the buffer length, the faster learning will converge since more observations are obtained for updating the posterior. We parameterize buffer length and evaluate its effect on learning and planning in experiments. Second, we verify with real data that virtually all patients adhere to the natural constraints on the transition probabilities given in Section 4.3. We exploit this known structure on the transition probabilities – i.e., that processes tend to degrade when passive and that interventions must have positive effect – to identify a constrained probability space from which we would like to sample when learning online. We implement a version of Thompson sampling called *constrained* Thompson sampling which samples from this joint, constrained probability space via rejection sampling.

On-demand index computation algorithm. When we learn online, the transition matrices for a process change every day, and thus pre-computing the Whittle indices for every belief state as in Alg. 4.4.1 is inefficient. We can address this by identifying and solving only the indifference equation that is relevant to the current state of the process. We use the insight that for a threshold of X_i on the current chain i, the corresponding threshold X_j on chain j would be the state with the

largest belief lower than $b(X_i)$, i.e., $X_j = \min_u \{u : b_j(u) < b(X_i)\}$. The Whittle index for X_i is then obtained by solving for $m : J_m^{(X_i,X_j)} = J_m^{(X_i+1,X_j)}$. These computations are repeated every day yielding overall complexity of $\mathcal{O}(|\Omega|T^2)$ per process.

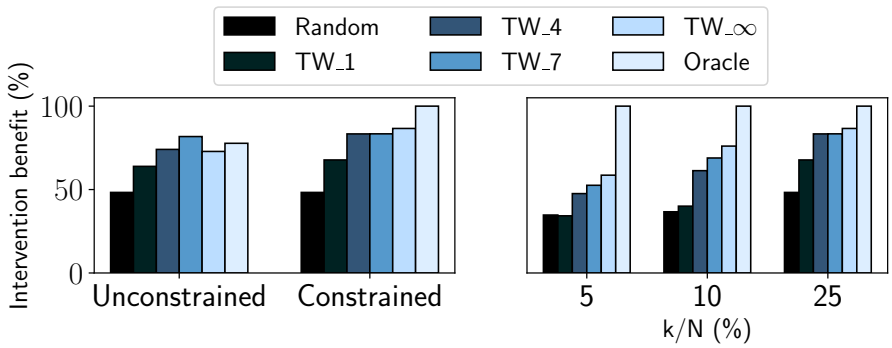


Figure C.2: (left) Constrained Thompson sampling improves learning. (right) buffer lengths of 4–7 perform well for various values of k/N, using constrained Thompson sampling. TW_X is the on-demand index algorithm run in tandem with Thompson sampling and a buffer length of X.

Fig. C.2 (right) evaluates the impact of varying buffer lengths for various ratios of k/N. Note that in these experiments, Oracle fully observes states, but must still learn transition probabilities online. Critically, we see that even when simulated patients report 4–7 observations per arm-pull, the performance is close to that of the non-Oracle learning upper bound (buffer length=∞) for any k/N. This is a key consideration for deployment in a medication adherence context: patients need only remember their last 4–7 doses on average for our approach to be nearly effective as possible in the TB context.

Fig. C.2 (left) compares the performance of learning policies with and without constrained Thompson sampling for $k/N = 25\%$. All policies benefit from the constrained sampling approach, suggesting that imposing our knowledge of the transition probability constraints was beneficial to learning.

C.7 Sensitivity Analysis

In Fig. C.3, we investigate Threshold Whittle's performance relative to the choice of parameters used to perturb the real data from the TB medication adherence domain. All the plots show that Threshold Whittle's performance is robust to the choice of parameters.

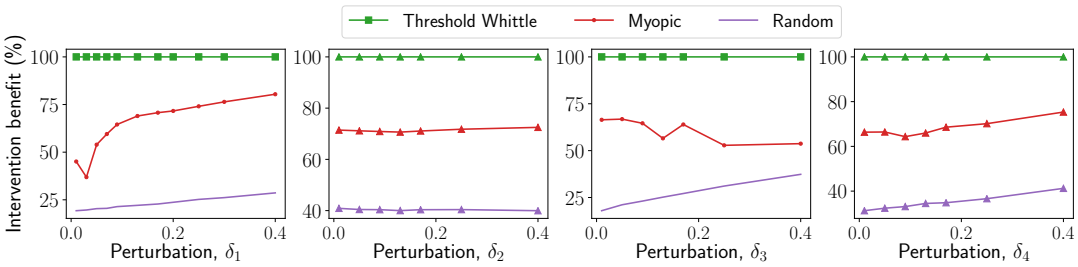


Figure C.3: Performance of Threshold Whittle is robust to perturbation of the transition matrix parameters. Note that 100% corresponds to the performance of Threshold Whittle for this plot only.

C.8 Performance on Reverse Threshold Optimal Processes

Here we investigate why Threshold Whittle demonstrates near-optimal performance even on reverse-threshold-optimal processes. We randomly sample forward and reverse threshold optimal processes, checked with Thm. 4.4.5 and Thm. 4.4.6, respectively, using $\beta = 0.95$, then compute their indices with the Threshold Whittle algorithm. Figures. C.4a and C.4b show a few samples of these trajectories for reverse and forward threshold optimal processes, respectively. Via similar arguments from the proof in Appendix C.1, it can be shown that the true Whittle indices for reverse (forward) threshold optimal processes should always be increasing (decreasing) in belief. Fig. C.4a shows that for such reverse threshold optimal processes, the indices computed by Threshold Whittle do tend to increase in belief as expected, which may lead to Threshold Whittle's good performance even though it is not guaranteed to be optimal on those processes. (And for completeness, Fig. C.4b

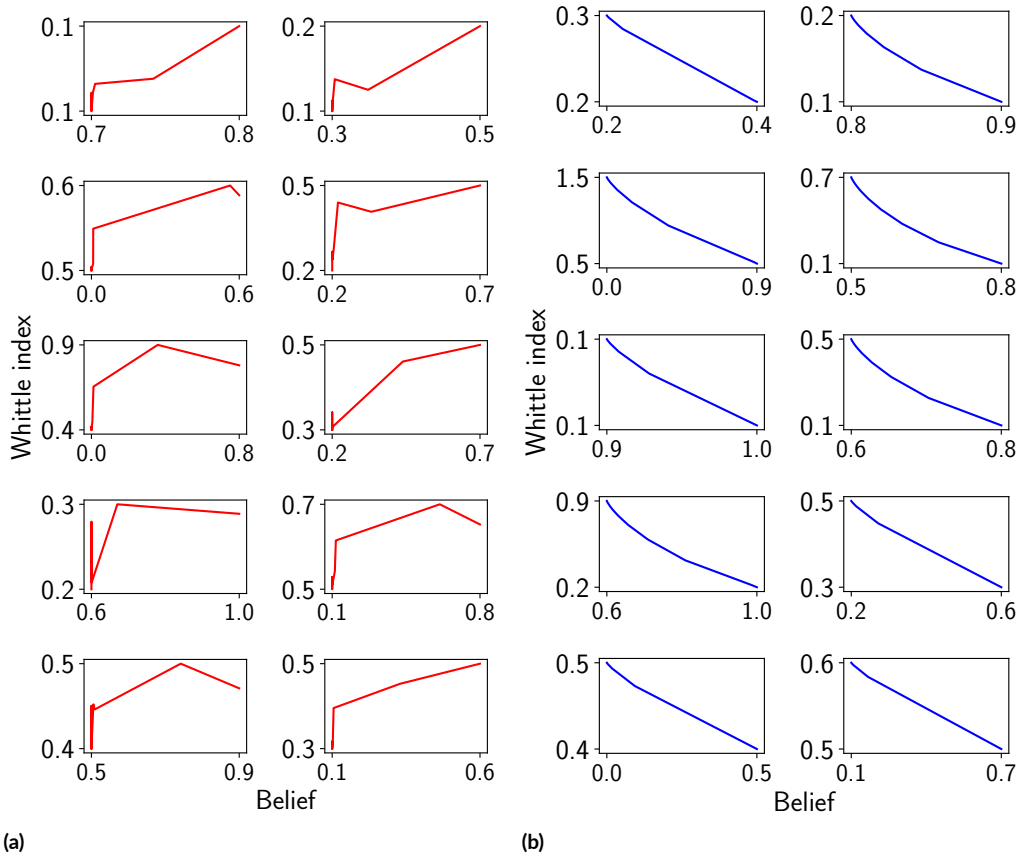


Figure C.4: (a) Threshold Whittle-computed indices vs. reachable beliefs for 10 randomly sampled reverse threshold optimal processes (one line per process). These indices tend to increase in belief, as expected for reverse threshold optimal processes according to the proof in Appendix C.1. (b) Threshold Whittle-computed indices vs. reachable beliefs for 10 randomly sampled forward threshold optimal processes (one line per process). These indices always decrease in belief, as expected for forward threshold optimal processes according to the proof in Appendix C.1.

shows that for forward threshold optimal policies, the indices computed by Threshold Whittle always decrease in belief as expected.)

D

Appendix to Chapter 5

D.1 Proofs

D.1.1 Proof of Proposition 5.4.1

Proposition D.1.1. *To learn the value λ that minimizes Eq. 5.4 given a state $\boldsymbol{s}$, the λ-network, parameterized by Λ, should be updated with the following gradient rule:*

$$\Lambda_t = \Lambda_{t-1} - \alpha \left(\frac{B}{1-\beta} + \sum_{n=1}^{N} D_n(s_n, \lambda_{t-1}(\boldsymbol{s})) \right) \tag{D.1}$$

where α is the learning rate and $D_n(s_n, \lambda)$ is the negative of the expected β-discounted sum of action costs for arm n starting at state s_n under the optimal policy for arm n for a given value of λ.

Proof. The gradient update rule is derived by taking the gradient of Eq. 5.4 with respect to λ, which has two main terms, $\lambda B/(1-\beta)$, and the sum over Q_n, the Q-functions with respect to λ. Looking more closely at Q_n, the only terms which are a function of λ are the costs of actions taken by the policy that Q_n implies, i.e., terms $-\lambda c_j$. Thus, the gradient of Q_n is the negative expected discounted sum of costs taken by the optimal policy at the given value of λ, i.e., $\frac{dQ_n}{d\lambda} = -\mathbb{E}[\sum_{t=0}^{H} \beta^t c_{n,t}]$, where $c_{n,t}$ is the cost of the action taken on arm n in round t. □

D.1.2 Proof of Proposition 5.4.2

Proposition D.1.2. *Given arm policies corresponding to optimal Q-functions, Prop. 5.4.1 will lead Λ to converge to the optimal as the number of training epochs and $K \to \infty$.*

Proof. Eq. 5.4 is convex in λ, which follows from definition of Q_n, i.e., the max over piece-wise linear functions of λ is also a convex function in λ. Thus the learning task of Λ is also convex. Therefore, all that is required for asymptotic convergence of Λ is that (1) the gradients we estimate via Prop. 5.4.1 are accurate, and that (2) all inputs, i.e., all states $\boldsymbol{s}$, are seen infinitely often in the limit. (1) is achieved by the assumption that optimal Q-functions are given, an analytic condition that is achieved in practice by allowing the arm-networks to train for a reasonable number of rounds under a given output of the λ-network, before updating Λ. Specifically, given optimal Q-functions and their corresponding optimal policies, the sampled sums of spent budget from those optimal policies represent an unbiased estimator of each D_n. Note, though that to be an *unbiased* estimator, this relies on not imposing the budget constraint *at training time*, a procedure we carry out in practice.* Thus (1) is achieved. (2) is achieved by following a training procedure that uniformly randomly samples start states $\boldsymbol{s}$ for each round of training until convergence. Thus the proposition is established. □

D.1.3 Proof of Proposition 5.4.3

Proposition D.1.3. *RR-DPO converges in a finite number of steps to the minimax regret-optimal policy.*

Proof. A common strategy for establishing optimal convergence of the double oracle is to show that the pure strategy sets of both players can be exhausted. We can achieve this in our setting under the conditions (1) that each player has a finite strategy set, i.e., is possible to be exhausted and (2) that each oracle gives an optimal best response. Since the agent pure strategy set is already finite, we can achieve (1) by discretizing the nature oracle—in effect by rounding the outputs of the

*It is critical to note that at test time, *we always impose the budget constraint* — i.e., all of our methods solve the original constrained RMAB problem — they only use the Lagrangian relaxation as a tool to find good policies to the original constrained problem.

policy network. For (2), for analytical purposes, we make the common assumption that our oracles internally converge to their optimal values, i.e., in our case, the arm-networks and λ-network converge optimally. However, since our networks learn the Lagrange-relaxed version of the problem, some additional tools are needed. Speficially, we must identify conditions in which DDLPO-Act gives policies which approach π^*_ω. This can be achieved in the binary-action setting with $\alpha =$ 'Whittle', which uses a binary search procedure to identify a value of λ such that exactly B arms have $Q_n(a = 1, \lambda) > Q_n(a = 0, \lambda)$, then acting on those arms. This procedure is equivalent to the Whittle index policy, which is asymptotically optimal for binary-action RMABs[161]. □

D.1.4 Proof of Proposition 5.4.4

Proposition D.1.4. *In the Robust RMAB problem with interval uncertainty, the max regret of a reward-maximizing policy can be arbitrarily large compared to a minimax regret-optimal policy.*

Proof. Consider a binary-action RMAB problem with two arms A and B. Let the reward from each arm be R when the arm is in a *good* state and 0 in a *bad* state. Our problem is to plan the best action with a budget of 1 and horizon of 1. Supposing the initial state is *bad* for each arm, the transition probabilities for the transition matrix for each arm n is $\begin{bmatrix} 1 & 0 \\ 1-p_n & p_n \end{bmatrix}$ where the uncertain variable p_n is constrained to be within $p_A, p_B \in [0, 1]$. Each value in the matrix corresponds to the probability of an arm at state *bad* transitioning to *bad* (column 1) or *good* (column 2) if we take the *passive* (row 1) or *active* action (row 2).

To compute a reward-maximizing policy that does not consider robustness to uncertainty, we must optimize for one instantiation of the uncertainty set, which requires making one of three assumptions.

- *Case 1:* If we assume $p_A = p_B$, then an optimal policy is to act with probability a_A on arm A and a_B on arm B as long as $a_A + a_B = 1$. W.l.o.g., suppose $a_A \geq a_B$; then nature would set

$p_A = 0$ and $p_B = 1$, imposing regret at least $R/2$.

- *Case 2:* If $p_A > p_B$, then the optimal policy would be to always act on arm A with probability $a_A = 1$ and never act on B ($a_B = 0$). Nature would then set $p_A = 0$ and $p_B = 1$ to impose regret R.

- *Case 3:* If $p_A < p_B$, the case is symmetric to Case 2 and result in regret R. Clearly, max regret is minimized when our action is such that $a_A + a_B = 1$; in this setting, we learn this optimal policy only under Case 1. Following Case 2 or 3, the difference between our regret and the minimax regret is $R/2$, which grows arbitrarily higher as $R \to \infty$.

A slight modification to this problem renders Case 1 non-optimal. Let the reward be R when arm A is in a *good* state and $R - 1$ for arm B, so the optimal policy learned under the assumption from Case 1 leads to $a_A = 1$ and $a_B = 0$. Then nature could respond with $p_A = 0$ and $p_B = 1$, yielding reward 0 and regret $R - 1$, while the minimax regret–optimal policy achieves a minimum reward of $(R - 1)/2$ (by playing $a_A = 0.5$ and $a_B = 0.5$ where nature responds with $p_A = 0$ and $p_B = 1$). Thus, the gap again can grow arbitrarily high as $R \to \infty$ provided that $R > 1$. We therefore have that in all cases, any reward-maximizing policy can achieve arbitrarily bad performance in terms of regret. □

D.2 DDLPO-Act subroutines

Here we provide the integer program which implements `QKnapsack`, one of the action-selection procedures used in Alg. 5.4.2 to take actions at test time. `QKnapsack` takes λ and $Q_n(s, a, \lambda)$ from the learned λ-network and arm networks, respectively, and returns the combination of actions that maximizes the sum of Q-values over all arms, subject to the costs of each action C and the budget constraint B.

$$\max_{X} \sum_{n=1}^{N} \sum_{j=1}^{|\mathcal{A}|} x_{nj} Q_n(s_n, a_{nj}, \lambda) \tag{D.2}$$

$$\text{s.t.} \sum_{i=n}^{N} \sum_{j=1}^{|\mathcal{A}|} x_{nj} c_j \leq B \tag{D.3}$$

$$\sum_{j=1}^{|\mathcal{A}|} x_{nj} = 1 \quad \forall n \in 1 \ldots N \tag{D.4}$$

$$x_{nj} \in \{0, 1\} \tag{D.5}$$

In Alg. D.2.1, we give the procedure `BinaSearch` which implements a binary search over the learned $Q(\lambda)$-values to find a charge λ for which exactly B arms prefer to act rather than not act. This mimics the Whittle index policy in binary-action settings.

D.3 Experimental Domain Details

D.3.1 ARMMAN

The MDPs in the ARMMAN domain [18] have three ordered states representing the level of engagement of the beneficiaries in the previous week. Rewards are better for lower states, i.e., $R(0) = 1, R(1) = 0.5, R(2) = 0$. At each step, the beneficiary may only change by one level, e.g., low-to-medium or high-to-medium but not low-to-high. They also assume that beneficiaries follow one of three typical patterns, A, B, and C, resulting in three MDPs with different transition probabilities. There are two patterns of effects present that differentiate the beneficiary types. (1) For each of the above types, the planner can only make a difference when the patient is in state 1. Type A responds very positively to interventions, but regresses to low reward states in absence. Type B has a similar but less amplified effect, and type C is likely to stay in state 1, but can be prevented from regressing

Algorithm D.2.1 BinaSearch (for the Whittle Index Policy)

Input: State $\boldsymbol{s}$, arm critic networks $\varphi_1, \ldots, \varphi_N$, budget B, tolerance ε.

$q_{nj} = \varphi_n(s_n, a_{nj}, \lambda = 0) \quad \forall n \in [N], \forall j \in [|\mathcal{A}|]$
$lb = 0$
$ub = \max_{n \in [N], j \in [|\mathcal{A}|]} \{q_{nj}\}$
while $ub - lb > \varepsilon$ **do**
  $\lambda = \frac{ub+lb}{2}$
  $q_{nj} = \varphi_n(s_n, a_{nj}, \lambda) \quad \forall n \in [N], \forall j \in [|\mathcal{A}|]$
  if fewer than B arms have $q_{n,j=1} > q_{n,j=0}$ **then**
    $ub = \lambda$ {Charging too much, decrease}
  else if more than B arms have $q_{n,j=1} > q_{n,j=0}$ **then**
    $lb = \lambda$ {Can charge more, increase}
  else if exactly B arms have $q_{n,j=1} > q_{n,j=0}$ **then**
    break
$\boldsymbol{a} = \mathbf{0}$
$a_n = 1$ where $q_{n,j=1} > q_{n,j=0}$
if $ub - lb \leq \varepsilon$ **then**
  break ties randomly s.t. $||\boldsymbol{a}||_1 = B$
return $\boldsymbol{a}$

to state 2 when an action is taken. (2) Further, types A and C have only a 10% chance of staying in the high reward state, while type B has a 90% chance of staying there.

We converted these patient types to robust versions where the transition probabilities are uncertain as follows:

$$T^i_{s=0} = \begin{bmatrix} p^i_{000} & 1 - p^i_{000} & 0.0 \\ p^i_{010} & 1 - p^i_{010} & 0.0 \end{bmatrix},$$

$$T^i_{s=1} = \begin{bmatrix} 0.0 & 1 - p^i_{102} & p^i_{102} \\ p^i_{110} & 1 - p^i_{110} & 0.0 \end{bmatrix},$$

$$T^i_{s=2} = \begin{bmatrix} 0.0 & 1 - p^i_{202} & p^i_{202} \\ 0.0 & 1 - p^i_{212} & p^i_{212} \end{bmatrix},$$

where i indexes the type (i.e., A, B or C). We then set each $p^i_{sas'}$ to be in a range of width 0.5 centered on the entries from each of the A, B, C beneficiary types for $s \in \{1, 2\}$. To add additional heterogeneity to the experiments, for $s = 0$, we set the range to 1.0 so that any beneficiary type can be made to have some non-negligible chance of staying in the good state, rather than only type B beneficiaries. The full set of parameter ranges are given in the Table D.1 below.

Param	L	U	L	U	L	U
Type A			Type B		Type C	
p^i_{000}	.00	1	.00	1	.00	1
p^i_{010}	.00	1	.00	1	.00	1
p^i_{102}	.50	1	.35	.85	.35	.85
p^i_{110}	.50	1	.15	.65	.00	.50
p^i_{202}	.35	.85	.35	.85	.35	.85
p^i_{212}	.35	.85	.35	.85	.35	.85

Table D.1: Upper (U) and lower (L) parameter ranges for the robust ARMMAN environment.

In all experiments, 20% of arms were sampled from type A, 20% from type B and 60% for type C. To add additional heterogeneity, for each of the 50 random seeds we uniformly sample a sub-range contained within the ranges given in Table D.1. In the agent oracle experiments, for each of the 50 random seeds, since these require fully instantiated transition matrices, we uniformly sample each parameter value for each arm according to its type such that the values are contained in the ranges given in Table D.1.

D.3.2 SIS Epidemic Model

In this domain, each arm follows its own compartmental SIS epidemic model. Each arm's SIS model tracks whether each of N_p members of a population is in a susceptible (S) or infectious (I) state. This can be tracked with N_p states, since it can be computed how many people are in state I if only the number of people in state S and the population size N_p is known.

To define a discrete SIS model, we instantiate the model given in Yaesoubi & Cohen [171] section 4.1 with a Δt of 1. We also augment the model to include action effects and rewards. Specifically, $R(N_S) = N_S/N_p$, where N_S is the number of susceptible (non-infected) people. Further, there are three actions $\{a_0, a_1, a_2\}$ with costs $c = \{0, 1, 2\}$. Action a_0 represents no action, a_1 divides the contacts per day κ (λ in Yaesoubi & Cohen [171]) by a_1^{eff}, and a_2 divides the infectiousness r_{infect} ($r(t)$ in Yaesoubi & Cohen [171]) by a_2^{eff}. That is, taking action a_1 will *reduce* the average number of contacts per day in a given arm, and taking action a_2 will reduce the probability of infection given contact in a given arm, thus reducing the expected number of people that will become infected in the next round. However, to make this a robust problem, the relative effect sizes of each action for each arm will not be known to the planner, nor will the κ or r_{infect}. We impose the following uncertainty intervals for all arms: $\kappa \in [1, 10]$, $r_{infect} \in [0.5, 0.99]$, $a_1^{eff} \in [1, 10]$, and $a_2^{eff} \in [1, 10]$.

In the robust double oracle experiments, to add additional heterogeneity, for each of the 50 random seeds we uniformly sample a sub-range contained within the ranges given above for each arm. In the agent oracle experiments, for each of the 50 random seeds, since these require fully instantiated transition matrices, we uniformly sample each parameter value for each arm such that the values are contained in the ranges given above.

D.4 Hyperparameter Settings and Implementation Details

Neural networks: All neural networks in experiments are implemented using PyTorch 1.3.1 [123] with 2 fully connected layers each with 16 units and tanh activation functions, and a final layer of appropriate size for the relevant output dimension with an identity activation function. The output of discrete actor networks (i.e., the policy network from the agent oracle, and the policy network of agent A in the nature oracle) pass through a categorical distribution from which actions are randomly sampled at training time, without a budget imposed. It is critical not to impose the budget at training time, so that the budget spent by the optimal policy under a given λ will result in a meaningful gradient for updating the λ-network. The output of continuous actor networks (i.e., agent B in the nature oracle which selects environment parameter settings) instead are passed as the means of Gaussian distributions – with the log standard deviations learned as individual parameters separate from the network – from which continuous actions are sampled at training time. At test time, actions are sampled from both types of networks deterministically. For categorical distributions, we greedily select the highest probability actions. For Gaussian distributions, we act according to the means. All discount factors were set to 0.9. The remaining hyperparameters that were constant for all experiments for the agent and nature oracles are indicated in Table D.2. For **Robust Double Oracle** experiments, all agent and nature oracles were run for 100 training epochs. For **Agent Oracle** experiments, DDLPO was run for 100 training epochs for the synthetic and ARMMAN domains and 200 epochs for the SIS domain.

λ-network: Critical to training the λ-network is cyclical control of the temperature parameter that weights the entropy term in the actor loss functions. Recall that the λ-network is only updated every `n_subepochs`. In general, after each update to the λ-network, we want to encourage exploration so that actor networks explore the new part of the state space defined by updated predictions of λ. However, after `n_subepochs` rounds, we will use the cost of the sampled actor policies as a

gradient for updating the λ-network, and that gradient will only be accurate if the actor policy has converged to the optimal policies for the given λ predictions. Therefore, we also want to have little or no exploration in the round before we update the λ-network. In general, we would also like the entropy of the policy network to reduce over time so that the actor networks and λ-networks eventually both converge.

To accomplish both of these tasks, the weight (temperature) of the entropy regularization term in the loss function of the actor network will decay/reset according to two processes. The first process will linearly decay the temperature from some positive, but time-decaying starting value (see next process) τ_t immediately after each λ-network update, down to 0 after `n_subepochs`. The second process will linearly decay the temperature from a maximum τ_0 (*start entropy coeff* in Table D.2) down to $\tau_{\min}$ (*end entropy coeff* in Table D.2) by the end of training.

We found that it also helps to train the actor network with no entropy and with the λ-network frozen for some number of rounds before training is stopped (*lambda freeze epochs* in Table D.2).

Double Oracle: In all experiments in Sec. 5.5, we initialize the agent strategy list with HO, HM, and HP, and the nature strategy list with pessimistic, mean, and optimistic nature strategies, then run RR-DPO for 6 iterations. This produces a set of 8 agent strategies, 8 nature strategies, a table where each entry represents the regret of each agent pure strategy (row) against each nature pure strategy (column), and an optimal mixed strategy over each set that represents a Nash equilibrium of the minimax regret game given in the table. The regret table is computed by first computing the returns of each agent/nature pure strategy combination, then subtracting the max value of each column from all entries in that column (i.e., the best agent strategy for a given nature strategy gets 0 regret). The regret of RR-DPO is reported as the expected utility corresponding to the Nash equilibrium of the regret game given by the table, once that regret table is normalized to account for the returns of baselines (see next paragraph).

After this main loop completes, we then compute the regret of the baselines by evaluating each

baseline policy against each pure strategy in the nature strategy list. Then, we also run the nature oracle against each baseline policy to find a nature strategy that should maximize the regret of that baseline. The regret for each baseline is reported as the max regret against this new nature strategy, as well as all pure nature strategies from the main RR-DPO loop.

Hawkins Baselines: The Hawkins policies are implemented with gurobipy 9.1.2, a Python wrapper for Gurobi (9.0.3)[58] following the LP given in Hawkins[60] equation 2.5 to compute λ and $Q(s, a, \lambda)$ for each arm and the integer program in equation 2.12 to select actions.

RLvMid Baseline: We found that RLvMid found effective policies for the nature strategy it was trained against (as evidenced in Figure 5.2)(a-f), but that that learned policy could be brittle against other nature strategies. This is likely because different nature strategies produce different distributions of states, meaning RLvMid would fit policies well to states seen when planning against the mean nature strategy, but underfit its policies for states seen more often in different distributions. However, the lone RLvMid baseline policy can somewhat correct for this effect by training an ensemble of policies against slight perturbations of the mean nature strategy that adjust the parameter values output by nature by a small ε. In all experiments we train 3 RLvMid policies against 3 random perturbations of the mean nature strategy, then report the regret of RLvMid as the minimum of the max regrets returned by any of the 3.

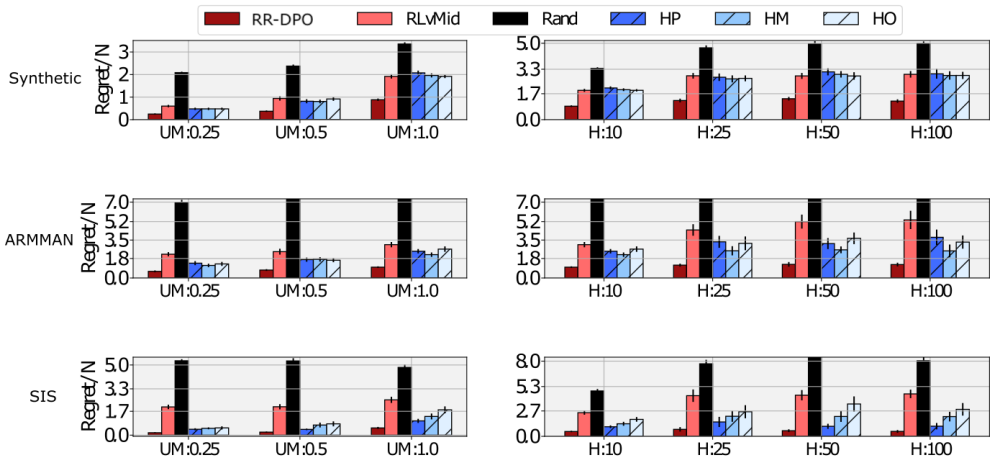


Figure D.1: (Left column) varies the uncertainty intervals to be 0.25, 0.5 and 1.0 times their widths (UM = uncertainty multiplier). The gap between our robust RR-DPO method and non-robust methods becomes larger as the uncertainty interval increases, and our robust algorithm RR-DPO always provides the lowest regret policies. (Right column) varies the horizon H in 10, 25, 50, 100. As expected, the gap between RR-DPO and the baselines either stays the same, or increases as H is increased, further demonstrating the robustness of our algorithm to various parameters.

parameter	value
agent	
clip ratio	2.0e+00
lambda freeze epochs	2.0e+01
start entropy coeff	5.0e-01
end entropy coeff	0.0e+00
actor learning rate	2.0e-03
critic learning rate	2.0e-03
lambda learning rate	2.0e-03
trains per epoch	2.0e+01
n_subepochs	4.0e+00
nature	
clip ratio	2.0e+00
lambda freeze epochs	2.0e+01
start entropy coeff	5.0e-01
end entropy coeff	0.0e+00
actorA learning rate	1.0e-03
criticA learning rate	1.0e-03
actorB learning rate	5.0e-03
criticB learning rate	5.0e-03
lambda learning rate	2.0e-03
trains per epoch	2.0e+01
n_subepochs	4.0e+00
n_sims	2.5e+01

Table D.2: Hyperparameter settings for agent and nature oracles for all experiments.

E

Appendix to Chapter 6

E.1 A Note on Language

Throughout chapter 6, we use the term "mother" to refer to pregnant women and people, birthing women and people, or postnatal women and people. We note that "mother" is gendered language that also typically refers to someone who has already given birth, which may not be the case for newly pregnant women and people who are enrolled in maternal health programs but have not yet

Table E.1: Notation table

Notation	Description
$[N]$	Set of N natural numbers $\{1, \ldots, N\}$
$[\underline{a}, \overline{a}]$	A real interval denoting all values of a such that $\underline{a} \leq a \leq \overline{a}$
N	Number of women (arms)
K	Number of interventions (service calls) that can be made each round
M	Number of groups
$\mathcal{S}_n$	Set of states (engagement status) of arm $n \in [N]$; $\mathcal{S}_n = \{0, 1\}$ (non-engaging state and engaging state)
$\mathcal{S}^N$	Combinatorial set of states over all arms
$\boldsymbol{s}$	Vector of states of all arms
$\mathcal{A}_n$	Set of actions (intervention decisions) on arm $n \in [N]$; $\mathcal{A}_n = \{0, 1\}$ (no-intervention and intervention)
$\mathcal{A}^N$	Combinatorial set of actions over all arms
$P^n_{s,a,s'}$	Probability of transitioning from state s to s' on action a for an individual n
$\overline{\underline{P}}^m$	Uncertainty intervals for all transition probabilities for group m
π_i	Planner's i-th pure strategy (Whittle index policy)
P_i	Adversary's i-th pure strategy (instantiation of P^m for all m)
α	Planner's mixed strategy
β	Adversary's mixed strategy
$G(\pi, P)$	Expected reward when planner plays a pure strategy π and adversary plays a pure strategy P
$R(\pi, P)$	Expected regret when planner plays a pure strategy π and adversary plays a pure strategy P

given birth.

We stand by the need to provide compassionate medical care for trans and non-binary patients, and recognize that the word "mother" may not reflect the identity of all people. Some recent calls advocate the language "birthing person" or "birthing women and people" over "mother"[133,56]. Others point out that language such as "birthing person" may be dehumanizing and go against best practices in health communication of making communication easy to understand for patients with low literacy or education or are communicating in their non-native language[57].

For the above reasons and to keep our writing concise, in chapter 6 we use "mother" while recognizing that the term is imperfect. We hope to move to healthcare language that is inclusive of all who are in need of those services. We also highlight the need to reevaluate other related terminology such as "maternal health" to move towards more inclusive language, and thus more inclusive care.

E.2 Ethical Considerations

As our system aims to improve engagement with maternal health information, this deployment must be carefully trialed with guardrails developed before a large-scale deployment. We are collaborating with ARMMAN to carry out these trials and develop such guardrails. We recognize the system's recommendations are influenced by historical data, a potential source of bias, especially when data is scarce. A benefit of our method is to be robust to such scarcity-induced bias, a key step forward toward responsible deployment.

Thus, some of the key ethical considerations are in how GROUPS may be used to optimize resource allocation in real world settings. We discussed how we, specifically, will work toward responsible deployment with our partners, as well as how GROUPS represents a new capability in RMAB planning, by allowing one to encode uncertainty due to, e.g., data scarcity. However, we must also consider the impact of a system like GROUPS on society more broadly. For instance, with such a scalable and inherently black box optimization tool such as GROUPS, there may be a temptation to allow the scheduling and delivery of interventions to become fully automated. This could negatively impact, e.g., the availability of work for humans who may have previously scheduled or delivered the interventions, or negatively impact intervention recipients who perhaps could receive unwanted interventions with little option for human-mediated recourse. To avoid these negative impacts, we stress that our system should be seen as a supplemental tool on the toolbelt of intervention schedulers, to be considered among a range of existing criteria and expertise, rather than a replacement

solution.

E.3 Limitations

Our work takes a major step forward in scaling up RMAB solutions to perform robustly under uncertainty. However, it accomplishes this, in part, by taking advantage of the decomposable nature of the Whittle index and Whittle index regret, which both require the binary-action setting typically considered for RMABs. This could be limiting in domains where planners need to optimize over a suite of different *types* of interventions, rather than deciding between only {act, not act} for each arm. We note though, that our methods could be extended to the multi-action setting with, e.g., multi-action notions of the Whittle index[53], and corresponding notions of multi-action index regret.

The scalability of our method also relies on the existence of a reasonable number of groups within data. We validate that our method runs hundreds of times faster than the state of the art when there are $\sim$ 10 arms and groups (Fig. E.5). We also show that our method runs in about 15 minutes for the real maternal health dataset which has $\sim$ 15K arms and 40 groups (Fig. E.3). However, even our method may have difficulty scaling if there were, e.g., tens of thousands of groups in the data. Yet, in that case, the planner could create a smaller number of groups, each with more arms per group. Then uncertainty of each group might increase, but the planner could still use GROUPS to plan and achieve good performance. We demonstrate an example of such a tradeoff between number of groups and performance in Fig. 6.3(e). GROUPS performs better than the baselines across all cases.

E.4 Additional Notation and Preliminaries

The number of mothers in group $m \in [M]$ is equal to γ_m. The total number of mothers is equal to $N = \sum_{m\in[M]} \gamma_m$. Further, $\mathcal{S}^m = \times_{n\in[\gamma_m]}\mathcal{S}_n^m$ (resp. $\mathcal{A}^m = \times_{n\in[\gamma_m]}\mathcal{A}_n^m$) denotes the set of different state-profiles (resp. action-profiles) of the mothers in group m. An element of $\mathcal{S}^m$ (resp. $\mathcal{I}^m$) is denoted as $\mathbf{s}^m = (s_n^m)_{n\in[\gamma_j]}$ (resp. $\mathbf{a}^m = (a_n^m)_{n\in[\gamma_m]}$). Finally, $\mathcal{S} = \times_{m\in[M]}\mathcal{S}^m$ (resp. $\mathcal{A} = \times_{m\in[M]}\mathcal{A}^m$) denotes the different state-profile (action-profile) of all the N mothers, and $\mathbf{s}$ denotes an element of $\mathcal{S}$.

A policy π is a map from $\mathcal{S}$ to $\mathcal{A}$, such that for each $\boldsymbol{s} \in \mathcal{S}$, $\pi(\boldsymbol{s})$ has at most K actions as 1 (intervention) and the remaining are 0 (no-intervention). In particular, π maps a strategy profile of the N mothers to an action profile with at most k intervention actions. We use Π to denote the set of all such policies. We note that Π is equal to the set of pure strategies of the planner. Let $\alpha \in \Delta_I(\Pi)$ (respectively $\beta \in \Delta_I(\mathscr{P})$) be a mixed strategy. Then we use $P_\alpha(\pi)$ (resp. $P_\beta(P)$) to denote the probability of choosing $\pi \in \Pi_I$ (resp. $P \in \mathscr{P}_I$) under α (resp. β).

Finally, we often use $(\alpha^\star_{X,Y}, \beta^\star_{X,Y})$ to denote the minimax maximin regret strategies of the planner and adversary respectively, when the planner plays mixed strategies from $\Delta_X(\Pi)$ and the adversary plays mixed strategies from $\Delta_Y(\mathscr{P})$ (see Section 6.5 for the definitions of $\Delta_X(\Pi)$ and $\Delta_Y(\mathscr{P})$), and $X, Y \in \{M, I\}$. Note that $\alpha^\star_{X,Y}$ is the planner's minimax strategy and $\beta^\star_{X,Y}$ is the adversary's maximin strategy.

E.4.1 Permutations on Π and $\mathscr{P}$

Let $\mathfrak{G}_m$ be the set of all permutations of $[\gamma_m]$, where a permutation is a bijective map from $[\gamma_m]$ to $[\gamma_m]$. We use σ_m to denote the elements of $\mathfrak{G}_m$. Further, let $\mathfrak{G} = \mathfrak{G}_1 \times \cdots \times \mathfrak{G}_M$. We use $\sigma = (\sigma_1, \ldots, \sigma_M)$ to denote an element of $\mathfrak{G}$; note that here $\sigma_m \in \mathfrak{G}_m$.

Let $\mathbf{a} = (a_n^m)_{n\in[\gamma_m], m\in[M]}$ be an action profile. Then for a $\sigma = (\sigma_1, \ldots, \sigma_M) \in \mathfrak{G}$, define

$\sigma(\mathbf{a}) = (a^m_{\sigma_m(n)})_{n\in[\gamma_m],m\in[M]}$. In particular σ permutes the actions corresponding to the mothers in group m according to σ_m. Now we show how a $\sigma \in \mathfrak{G}$ defines bijective maps on $\mathscr{P}$ and Π. For every $\sigma \in \mathfrak{G}$ define the map $\varphi_\sigma : \Pi \to \Pi$ (respectively $\psi_\sigma : \mathscr{P} \to \mathscr{P}$) as follows: denote $\varphi_\sigma(\pi)$ as π' (resp. $\psi_\sigma(P)$ as P'), then $\pi'(\mathbf{s}) = \sigma(\pi(\mathbf{s}))$ (resp. $(P')^m_n = P^m_{\sigma_m(n)}$). We make the following observation which will be helpful later on.

Observation 1. *For all $\pi \in \Pi$, $P \in \mathscr{P}$, and $\sigma \in \mathfrak{G}$ the following holds $R(\pi, P) = R(\varphi_\sigma(\pi), \psi_\sigma(P))$.*

The following observation follows from the fact that for every $\sigma \in \mathfrak{G}$, φ_σ and ψ_σ define bijective maps on Π and P respectively.

Observation 2. *Let $\alpha \in \Delta_I(\Pi)$ and $\beta \in \Delta_I(\mathscr{P})$ be mixed strategies of the planner and adversary respectively. Then for any $\sigma \in \mathfrak{G}$ the following holds:*

$$R(\alpha, \beta) = \sum_{\pi,\mathbf{p}} P_\alpha(\varphi_\sigma(\pi)) \cdot P_\beta(\psi_\sigma(P)) \cdot R(\varphi_\sigma(\pi), \psi_\sigma(P)) \,.$$

Next, we define permutations which are transpositions. We say $\sigma_m \in \mathfrak{G}_m$ is a transposition if there exists $n, n' \in [\gamma_m]$ such that $\sigma_m(n) = n'$ and $\sigma_m(n') = n$, and for all $\ell \notin \{n, n'\}$, $\sigma_m(\ell) = \ell$. We say $\sigma = (\sigma_1, \ldots, \sigma_M) \in \mathfrak{G}$ is a transposition if for all $m \in [M]$, σ_m is a transposition. We note that every $\sigma \in \mathfrak{G}$ can be expressed as a composition of finitely many transpositions. We note the following observation.

Observation 3. *Let $\sigma \in \mathfrak{G}$ be a transposition. Then for every $\pi \in \mathcal{A}$ and $P \in \mathscr{P}$, $\varphi_\sigma(\varphi_\sigma(\pi)) = \pi$ and $\psi_\sigma(\psi_\sigma(\mathscr{P})) = P$.*

E.4.2 Mixed Strategies that Do Not Distinguish Between Mothers in the Same Group

We say a mixed strategy α of the planner is indifferent towards mothers from the same group if for all $\pi, \pi' \in \Pi$ such that there is a $\sigma \in \mathfrak{G}$ satisfying $\pi' = \varphi_\sigma(\pi), P_\alpha(\pi) = P_\alpha(\pi')$. We use $\Delta_M(\Pi)$ to denote such mixed strategies.

If we define the probability of intervening on mother n under a mixed strategy α as follows

$$P_\alpha(\text{intervene } n) = \sum_{\pi \in \Pi} P_\alpha(\pi) \sum_{\mathbf{s} \in \mathcal{S}} 1\{a^\pi_{\mathbf{s},n} = 1\}$$

then it is easy to see that for an $\alpha \in \Delta_M(\Pi)$ and mothers n, n' in the same group $P_\alpha(\text{intervene } n) = P_\alpha(\text{intervene } n')$.

E.5 Proofs of Theorem 1, 2, and 3

We refer the reader to Section E.4 for additional notations and missing definitions used in the proof. Additionally, we now define order-preserving maps, that we will use in the proof of Theorem 6.5.3.

E.5.1 Order-Preserving Maps

To prove Theorem 6.5.3 from Section 6.5.3, we require the assumption that there is a map $\psi : \Delta_I(\mathscr{P}) \to \Delta_M(\mathscr{P})$ such that

1. For every $\alpha_1, \alpha_2 \in \Delta_M(\Pi)$ and $\beta \in \Delta_I(\mathscr{P})$, $R(\alpha_1, \beta) > R(\alpha_2, \beta)$ iff $R(\alpha_1, \psi(\beta)) > R(\alpha_2, \psi(\beta))$, and

2. For every $\alpha \in \Delta_M(\Pi)$ and $\beta_1, \beta_2 \in \Delta_I(\mathscr{P})$ $R(\alpha, \beta_1) > R(\alpha, \beta_2)$ iff $R(\alpha, \psi(\beta_1)) > R(\alpha_2, \psi(\beta_1))$

While (1) and (2) may not hold for general mixed strategies in $\Delta_I(\Pi)$, it is likely to be true for mixed strategies in $\Delta_M(\Pi)$, since these strategies do not distinguish between mothers from the same group. Next, we describe a possible candidate map.

First we define a map $\varphi : \mathscr{P} \to \mathscr{P}$. Let $P \in \mathscr{P}$ be a pure strategy of the adversary which assigns different transition probabilities to the mothers in the same group, and for pure strategy $\mathbf{p}$ let $P^{m,n}_{s,a,s'}$ be the probability of the mother n in group m transitioning from state s to s' under action a. We define $\varphi(P) = \hat{P}$ as follows, $\hat{P}$ assigns the same transition probability to all the mothers in a group by averaging the transition probabilities of the mothers in that group. In particular, $\hat{P}^{m,n}_{s,a,s'} = \hat{P}^{m}_{s,a,s'} = \sum_{m \in [\gamma_m]} p^{m,n}_{s,a,s'}$. Notice that the pure strategy $\varphi(P)$ of the adversary is indifferent to mothers in the same group. Further, for a $P \in \mathscr{P}$, let $\varphi^{-1}(P) = \{P' \in P \mid \varphi(P') = \mathbf{p}\}$. Now let $\beta \in \Delta_I(\mathscr{P})$. Then $\psi(\beta) = \beta'$ is defined as follows: $P_{\beta'}(P) = \sum_{P' \in \varphi^{-1}(p)} P_\beta(P')$. Since, $\varphi^{-1}(P) \neq \emptyset$ only if $\mathbf{p}$ assigns the same transition probability to all the mothers in a group. Hence, we have $\psi(\beta) \in \Delta_M(\mathscr{P})$.

E.5.2 Proof of Theorem 6.5.1

We require the following proposition to prove the theorem.

Proposition E.5.1. *Suppose the minimax maximin regret strategies $(\alpha^\star_{I,I}, \beta^\star_{I,I})$ is such that there exists a permutation $\sigma \in \mathfrak{S}$ satisfying $P_\beta(P) = P_\beta(\psi_\sigma(P))$ for every $P \in \mathscr{P}_I$. Then there exists a planner mixed strategy $\alpha' \in \Delta_I(\Pi)$ such that $P_\alpha(\varphi_\sigma(\pi)) = P_\alpha(\pi)$ for every $\pi \in \Pi$, and $(\alpha', \beta^\star_{I,I})$ are minimax maximin regret strategies of the planner and adversary respectively at the individual level.*

Proof. Suppose there exists $\pi_m, \pi_\ell \in \Pi_I$ such that $\varphi_\sigma(\pi_m) = \pi_\ell$ but $P_\alpha(\pi_\ell) < P_\alpha(\pi_m)$. First, we

show in this case that $R(\pi_m, \beta^\star_{I,I}) = R(\pi_\ell, \beta^\star_{I,I})$.

$$
\begin{aligned}
R(\pi_\ell, \beta^\star_{I,I}) &\overset{=}{(i)} \sum_{\mathbf{p} \in P} P_{\beta^\star_{I,I}}(P) \cdot R(\pi_\ell, P) \\
&\overset{=}{(ii)} \sum_{\mathbf{p} \in P} P_{\beta^\star_{I,I}}(\psi_\sigma(P)) \cdot R(\pi_\ell, \psi_\sigma(P)) \\
&\overset{=}{(iii)} \sum_{\mathbf{p} \in P} P_{\beta^\star_{I,I}}(P) \cdot R(\pi_m, P) \\
&= R(\pi_m, \beta^\star_{I,I})
\end{aligned}
$$

In the above equations: (i) follows from the definition of regret of a pure strategy π_ℓ on the adversary's mixed strategy $\beta^\star_{I,I}$, (ii) follows as σ is a permutation (also see Observation 2) and from Observation 1 we have $R(\pi_m, P) = R(\pi_\ell, \psi_\sigma(P))$ for all $P \in \mathscr{P}$, and (iii) follows by using $P_{\beta^\star_{I,I}}(P) = P_{\beta^\star_{I,I}}(\psi_\sigma(P))$ for all $P \in \mathscr{P}$. Now construct a mixed strategy α' such that $P_{\alpha'}(\pi_m) = P_{\alpha'}(\pi_\ell) = \frac{P_\alpha(\pi_m)+P_\alpha(\pi_\ell)}{2}$, and for all $\pi \in \Pi_I$ such that $\pi \neq \pi_m$ and $\pi \neq \pi_\ell$ we have $P_{\alpha'}(\pi) = P_\alpha(\pi)$. Since $R(\pi_m, \beta^\star_{I,I}) = R(\pi_\ell, \beta^\star_{I,I})$, it is easy to see that (α', β) is a minimax maximin regret strategies at the individual level, and from construction $P_{\alpha'}(\pi_m) = P_{\alpha'}(\pi_\ell)$. □

Proof of Theorem 6.5.1. If $\alpha^\star_{I,I} \in \Delta_M(\Pi)$ then the Theorem follows. Hence, assume there are $\pi_m, \pi_\ell \in \Pi$ and a permutation $\sigma \in \mathfrak{G}$ such that $\varphi_\sigma(\pi_m) = \pi_\ell$ but $P_\alpha(\pi_m) > P_\alpha(\pi_\ell)$. Here, we use the subscript m and ℓ to denote more and less. Observe that we may assume without loss of generality that σ is a transposition (see App. E.4). This is because any permutation in $\sigma \in \mathfrak{G}$ can be expressed as a composition of transpositions. Hence, assuming σ is a transposition, we have $\varphi_\sigma(\pi_\ell) = \pi_m$.

Let $(\alpha^\star_{I,I}, \beta^\star_{I,I})$ be a minimax maximin regret strategies, that is, $\beta^\star_{I,I}$ is a regret maximizing mixed

strategy of adversary against $\alpha^\star_{I,I}$, that is

$$\beta^\star_{I,I} \in \operatorname*{argmax}_{\beta \in \Delta_I(\mathscr{P})} R(\alpha^\star_{I,I}, \beta) \,.$$

Construct β' such that $P_{\beta'}(P) = P_{\beta^\star_{I,I}}(\psi_\sigma(P))$ for all $P \in \mathscr{P}$. Note that since σ is a transposition, we also have $P_{\beta'}(\psi_\sigma(P)) = P_{\beta^\star_{I,I}}(P)$ for all $P \in \mathscr{P}$. Hence, as $\beta^\star_{I,I}$ is regret maximizing for the adversary against the planner's mixed strategy $\alpha^\star_{I,I}$, we have $R(\alpha^\star_{I,I}, \beta') \leq R(\alpha^\star_{I,I}, \beta^\star_{I,I})$.

First, we argue that $R(\alpha^\star_{I,I}, \beta') = R(\alpha^\star_{I,I}, \beta^\star_{I,I})$. Suppose $R(\alpha^\star_{I,I}, \beta') < R(\alpha^\star_{I,I}, \beta^\star_{I,I})$. Then writing the regret expressions for $R(\alpha^\star_{I,I}, \beta')$ and $R(\alpha^\star_{I,I}, \beta^\star_{I,I})$ we have

$$\begin{aligned} &\sum_{\mathbf{p}} \sum_{\pi} P_{\beta'}(\psi_\sigma(P)) \cdot P_{\alpha^\star_{I,I}}(\pi) \cdot R(\pi, P) \\ &\quad < \sum_{\mathbf{p}} \sum_{\pi} P_{\beta^\star_{I,I}}(P) \cdot P_{\alpha^\star_{I,I}}(\pi) \cdot R(\pi, P) \end{aligned} \tag{E.1}$$

Substituting $R(\pi, P) = R(\varphi_\sigma(\pi), \psi_\sigma(P))$ for all π, P (see Observation 1) we have

$$\begin{aligned} &\sum_{\mathbf{p}} \sum_{\pi} P_{\beta^\star_{I,I}}(\psi_\sigma(P)) \cdot P_{\alpha^\star_{I,I}}(\pi) R(\varphi_\sigma(\pi), \psi_\sigma(P)) \\ &\quad < \sum_{\mathbf{p}} \sum_{\pi} P_{\beta^\star_{I,I}}(P) \cdot P_{\alpha^\star_{I,I}}(\pi) R(\pi, P) \end{aligned} \tag{E.2}$$

Let α' be the mixed strategy of planner such that $P_{\alpha'}(\varphi_\sigma(\pi)) = P_{\alpha^\star_{I,I}}(\pi)$. Substituting this in the above equation we have

$$\begin{aligned} &\sum_{\mathbf{p}} \sum_{\pi} P_{\beta^\star_{I,I}}(\psi_\sigma(P)) \cdot P_{\alpha'}(\varphi_\sigma(\pi) R(\varphi_\sigma(\pi), \psi_\sigma(P)) \\ &\quad < \sum_{\mathbf{p}} \sum_{\pi} P_{\beta^\star_{I,I}}(P) \cdot P_{\alpha^\star_{I,I}}(\pi) R(\pi, P) \end{aligned} \tag{E.3}$$

From Observation 2 we have,

$$\sum_{\mathbf{p}} \sum_{\pi} P_{\beta^\star_{I,I}}(\psi_\sigma(P)) \cdot P_{\alpha'}(\varphi_\sigma(\pi)) \cdot R(\varphi_\sigma(\pi), \psi_\sigma(P)) \tag{E.4}$$

$$= R(\alpha', \beta^\star_{I,I})$$

Hence $R(\alpha', \beta^\star_{I,I}) < R(\alpha^\star_{I,I}, \beta^\star_{I,I})$. This contradicts the minimax theorem which states that $\alpha^\star_{I,I}$ is the regret minimizing mixed strategy of the planner against the adversary's mixed strategy $\beta^\star_{I,I}$. Hence, we have $R(\alpha^\star_{I,I}, \beta') = R(\alpha^\star_{I,I}, \beta^\star_{I,I})$. We note that the above equations also show that α' is the regret minimizing mixed strategy for the planner in response to adversary's mixed strategy β'.

Now construct $\tilde{\beta}$ such that

$$P_{\tilde{\beta}}(P) = \frac{P_{\beta^\star_{I,I}}(P) + P_{\beta'}(P)}{2} = \frac{P_{\beta^\star_{I,I}}(P) + P_{\beta^\star_{I,I}}(\psi_\sigma(P))}{2}.$$

We now argue that $(\alpha^\star_{I,I}, \tilde{\beta})$ is a minimax maximin regret strategies at the individual level, that is, $\alpha^\star_{I,I}$ is regret minimizing against $\tilde{\beta}$, and $\tilde{\beta}$ is regret maximizing against $\alpha^\star_{I,I}$. Since σ is a transposition, this implies $P_{\tilde{\beta}}(P) = P_{\tilde{\beta}}(\psi_\sigma(P))$ for all $\mathbf{p} \in P$. Further, as $R(\alpha^\star_{I,I}, \beta') = R(\alpha^\star_{I,I}, \beta^\star_{I,I})$, we have $R(\alpha^\star_{I,I}, \beta^\star_{I,I}) = R(\alpha^\star_{I,I}, \tilde{\beta})$, and hence, $\tilde{\beta}$ is a regret maximizing mixed strategy of the adversary against $\alpha^\star_{I,I}$, that is,

$$\tilde{\beta} \in \operatorname*{argmax}_{\beta \in \Delta_I(\mathscr{P})} R(\alpha^\star_{I,I}, \beta)\,.$$

Also, a similar argument, as from Equations E.1 to E.4, shows that $R(\alpha^\star_{I,I}, \beta') = R(\alpha', \beta')$, and hence $\alpha^\star_{I,I}$ is a regret minimizing mixed strategy for the planner in response to adversary's mixed strategy β'. This follows from the minimax theorem and that α' is the regret minimizing mixed strategy for the planner in response to adversary's mixed strategy β'. This together implies $\alpha^\star_{I,I}$ is the regret minimizing mixed strategy for the planner in response to adversary's mixed strategy $\tilde{\beta}$. In

particular, $(\alpha',)$

Now we use Proposition E.5.1, which shows that there exists a $\tilde{\alpha}$ such that $P_{\tilde{\alpha}}(\pi_m) = P_{\tilde{\alpha}}(\pi_\ell)$,

$$\tilde{\beta} \in \operatorname*{argmax}_{\beta \in \Delta_I(\mathscr{P})} R(\tilde{\alpha}, \beta)$$

and $R(\tilde{\alpha}, \tilde{\beta}) = R(\alpha^\star_{I,I}, \tilde{\beta}) = R(\alpha^\star_{I,I}, \beta^\star_{I,I})$. We can repeat this process finitely many times to show to construct a mixed strategy α such that for any two policies $\pi, \pi' \in \mathcal{A}$, if there exists a σ such that $\varphi_\sigma(\pi) = \pi'$ then $P_\alpha(\pi) = P_\alpha(\pi')$, and $\max_{\beta \in \Delta_I(\mathscr{P})}(R(\alpha, \beta)) = R(\alpha^\star_{I,I}, \beta^\star_{I,I})$. Since $\alpha \in \Delta_M(\Pi)$, we have, without loss of generality $\alpha = \alpha^\star_{M,I}$. □

E.5.3 Proof of Theorem 6.5.2

Let $(\alpha^\star_{M,M}, \beta^\star_{M,M})$ be a minmax-maximin regret strategies at the group level. Further, let $\pi, \pi' \in \Pi$ be such that $P_{\alpha^\star_{M,M}}(\pi) > P_{\alpha^\star_{M,M}}(\pi')$, and there exists a $\sigma \in \mathfrak{G}$ such that $\pi' = \varphi_\sigma(\pi)$. Let α be an planner mixed strategy such that $P_\alpha(\pi) = P_\alpha(\pi') = \frac{P_{\alpha^\star_{M,M}}(\pi) + P_{\alpha^\star_{M,M}}(\pi')}{2}$. First observe that since $\beta \in \Delta_M(\mathscr{P})$, $R(\pi, \beta) = R(\pi', \beta)$. It follows from this that (α, β) is also a minmax regret solution at the group level. We can repeat this process finitely many times to show that for any two policies $\pi, \pi' \in \Pi$, if there exists a σ such that $\varphi_\sigma(\pi) = \pi'$ then $P_\alpha(\pi) = P_\alpha(\pi')$.

E.5.4 Proof of Theorem 6.5.3

Let $(\alpha^\star_{M,M}, \beta^\star_{M,M})$ be a minimax-maximin regret strategies at the group level. Also, let $\beta^\star$ be the regret maximizing strategy of the adversary at the individual level against the planner's mixed strategy $\alpha^\star_{M,M}$, that is

$$\beta^\star = \operatorname*{argmax}_{\beta \in \Delta_I(\mathscr{P})} R(\alpha^\star_{M,M}, \beta)$$

Further, let $(\alpha^\star_{I,I}, \beta^\star_{I,I})$ be a minimax-maximin regret strategies at the individual level. In particular, from the minimax theorem

$$\beta^\star_{I,I} = \max_{\beta \in \Delta_I(\mathscr{P})} R(\alpha^\star_{I,I}, \beta)$$

Hence, we wish to show $R(\alpha^\star_{M,M}, \beta^\star) = R(\alpha^\star_{I,I}, \beta^\star_{I,I})$. Now let $(\alpha^\star_{M,I}, \beta^\star_{M,I})$ be a minimax-maximin regret strategies, when the planner plays at group level (from $\Delta_M(\Pi)$) and the adversary plays at the individual level (from $\Delta_I(\mathscr{P})$). Hence, again from the minimax theorem

$$\beta^\star_{M,I} = \max_{\beta \in \Delta_I(\mathscr{P})} R(\alpha^\star_{M,I}, \beta)$$

Recall from Theorem 6.5.1, we have

$$R(\alpha^\star_{I,I}, \beta^\star_{I,I}) = R(\alpha^\star_{M,I}, \beta^\star_{M,I})$$

Hence, to prove the theorem it is sufficient to show that $R(\alpha^\star_{M,M}, \beta^\star) = R(\alpha^\star_{M,I}, \beta^\star_{M,I})$. Since $(\alpha^\star_{M,I}, \beta^\star_{M,I}))$ is a minimax-maximin strategy regret strategies, when the planner plays at group level (from $\Delta_M(\Pi)$) and the adversary plays at the individual level, we have

$$R(\alpha^\star_{M,M}, \beta^\star) \geq R(\alpha^\star_{M,I}, \beta^\star_{M,I})$$

Suppose for contradiction

$$R(\alpha^\star_{M,M}, \beta^\star) > R(\alpha^\star_{M,I}, \beta^\star_{M,I}) \tag{E.5}$$

Now we have the following two equations:

$$R(\alpha^\star_{M,M}, \beta^\star) \geq R(\alpha^\star_{M,M}, \beta^\star_{M,I}) \tag{E.6}$$

$$R(\alpha^\star_{M,M}, \beta^\star_{M,I}) \geq R(\alpha^\star_{M,I}, \beta^\star_{M,I}) \tag{E.7}$$

Equation E.6 follows from $\beta^\star$ being the regret maximizing strategy of the adversary at the individual level against the planner's mixed strategy $\alpha^\star_{M,M}$, and Equation E.7 follows from the minimax theorem and $\alpha^\star_{M,I}, \beta^\star_{M,I}$ being the minimax maximin regret strategies when the planner plays at the group level and the adversary plays at the individual level. Now corresponding to Equations E.6 and E.7 assuming there is an order preserving map (see App. E.5.1) $\psi : \Delta_I(\mathscr{P}) \to \Delta_M(\mathscr{P})$, we have

$$R(\alpha^\star_{M,M}, \psi(\beta^\star)) \geq R(\alpha^\star_{M,M}, \psi(\beta^\star_{M,I})) \tag{E.8}$$

$$R(\alpha^\star_{M,M}, \psi(\beta^\star_{M,I})) \geq R(\alpha^\star_{M,I}, \psi(\beta^\star_{M,I})) \tag{E.9}$$

Equation E.8 follows from property 1 of the order preserving map, and Equation E.9 follows from property 2. From Equations E.5, E.8 and E.9, we have

$$R(\alpha^\star_{M,M}, \psi(\beta^\star)) > R(\alpha^\star_{M,I}, \psi(\beta^\star_{M,I})) \tag{E.10}$$

Finally we use property 2 of the order preserving map to claim the following two equations,

$$R(\alpha^\star_{M,M}, \psi(\beta^\star)) = R(\alpha^\star_{M,M}, \beta^\star_{M,M}) \tag{E.11}$$

$$R(\alpha^\star_{M,I}, \psi(\beta^\star_{M,I})) = \max_{\beta \in \Delta_M(\mathscr{P})} R(\alpha^\star_{M,I}, \beta) \tag{E.12}$$

Both equations require property 2 of the order-preserving map. Additionally, Equation E.11, follows because $\beta^\star$ (resp. $\beta^\star_{M,M}$) is regret maximizing for adversary at the individual level (resp. at the group level) against planner's strategy $\alpha^\star_{M,M}$, and Equation E.12 follows because $\beta^\star_{M,I}$ is regret maximizing for adversary at the individual level against planner's strategy $\alpha^\star_{M,I}$. Hence, from Equations E.10, E.11 and E.12, we have

$$R(\alpha^\star_{M,M}, \beta^\star_{M,M}) > \max_{\beta \in \Delta_M(\mathscr{P})} R(\alpha^\star_{M,I}, \beta)$$

The above equation contradicts the worst-case minimality of $\alpha^\star_{M,M}$, when both players play at the group level. Hence, we have $R(\alpha^\star_{M,M}, \beta^\star) \geq R(\alpha^\star_{M,I}, \beta^\star_{M,I})$.

E.6 Minimizing/Maximizing Whittle Indices

The binary quadratic program for simultaneously maximizing and/or minimizing the Whittle indices over one or more states of a group, given a set of interval parameter ranges on the transitions probabilities $[\underline{P}_{s,a,s'}, \overline{P}_{s,a,s'}]$, named MinMaxWhittleBQP is given as follows:

$$
\begin{aligned}
&\min_{W_{s'}} \sum_{s' \in I(\mathcal{S})} \theta_{s'} W_{s'} && \text{Primary objective} \\
&\min_{V^{s'}} \sum_{s' \in I(\mathcal{S})} V^{s'}(s', W_{s'}) && \text{Secondary objective} \\
&\text{s.t.} \\
&Q^{s'}(s, a, W_{s'}) = R(s) - W_{s'} C(a) + \gamma T(s, a, \cdot)^\intercal V^{s'}(\cdot, W_{s'}) \\
&V^{s'}(s, W_{s'}) \geq Q^{s'}(s, a, W_{s'}) \\
&V^{s'}(s, W_{s'}) \leq Q^{s'}(s, a, W_{s'}) + b(s, a, s')M \\
&\qquad \forall s \in \mathcal{S}, a \in \mathcal{A}, s' \in I(\mathcal{S}) \\
&\sum_{a \in \mathcal{A}} b(s, a, s') = |\mathcal{A}| - 1 \\
&\qquad \forall s \in \mathcal{S}, s' \in I(\mathcal{S}) \\
&W_{s'} = \gamma \left[T(s', 1, \cdot)^\intercal V^{s'}(\cdot, W_{s'}) - T(s', 0, \cdot)^\intercal V^{s'}(\cdot, W_{s'}) \right] \\
&\qquad \forall s' \in I(\mathcal{S}) \\
&T(s, a, s'') \in [\underline{P}_{s,a,s''}, \overline{P}_{s,a,s''}] \\
&\qquad \forall s \in \mathcal{S}, a \in \mathcal{A}, s'' \in \mathcal{S}
\end{aligned}
\tag{E.13}
$$

Where $I(\mathcal{S})$ is the set of all states for which the users wants to jointly optimize Whittle indices, $\theta_{s'} \in \{-1, 0, 1\}$ is the "sense" for the corresponding index to optimize, i.e., 1 to minimize, -1 to maximize, or 0 to not optimize the index for that state (note that θ corresponds to *obj* in Alg. 6.4.2), $W_{s'}$ is the Whittle index for state s', V and Q are the state and state-action value functions, respectively, $C(a) = a$ is the cost of an action, b is a binary variable that serves to enforce one of the Q constraints on V to be tight (ensuring the value function is solved, and thus Whittle index is valid), M is a large number, e.g., 10^4, T are variables that hold the transition probabilities, $R(s) = s$ is the reward, and γ is the discount factor. Note that all $T^\intercal V$ terms are quadratic, since both T and V are variables in this optimization.

E.7 Double Oracle and Whittle Index Algorithms

The outer loop of the double oracle algorithm is given in Alg. E.7.1. We also give ComputeWI in Alg. E.7.2, our binary-search based method for computing the Whittle index, given transition probabilities for a group P^m and a state s. Note also that ComputeWI could be implemented by using MaxWhittleBQP, and a small wrapper function to adjust the input appropriately. That is, MaxWhittleBQP expects intervals over transition parameters $\overline{\underline{P}}^m$, but computing the Whittle index only requires some choice of P^m in the intervals. Thus the wrapper needs to encode P^m as intervals, which can be accomplished by copying each transition probability of P^m to an interval with the same upper and lower bound, for each s, a, s'. Then, to get the Whittle index for a certain state of the group m, the wrapper should pass in a *sense* that negative-one-hot encodes the desired state. E.g., if one wants to compute the index for state $s = 1$ for an arm with two states, the wrapper should pass in a sense $\theta = [0, -1]$. With the above described inputs, specifically since the intervals will have the same upper and lower bound, the quadratic terms in the MaxWhittleBQP will become constant, effectively turning the binary quadratic program into a binary linear program

Algorithm E.7.1 Double Oracle

Input: Grouped RMAB simulator and parameter uncertainty intervals $\overline{\underline{P}}^m$ for all groups.
Parameters: Number of iterations T
Output: Agent mixed strategy α

1: $P_0 = \{P_0\}$, with P_0 selected at random
2: $\Pi_0 = \{\pi_{B_1}, \pi_{B_2}, \ldots\}$, where π_{B_i} are baseline and heuristic strategies
3: **for** epoch $e = 1, 2, \ldots, T$ **do**
4: Solve for (α_e, β_e), mixed Nash equilibrium of regret game with strategy sets P_{e-1} and Π_{e-1}
5: $\pi_e = \text{WI4MS}(\beta_e)$
6: $P_e = \textsc{RegretMaxWI}(\alpha_e)$
7: $P_e = P_{e-1} \cup \{P_e\}, \Pi_e = \Pi_{e-1} \cup \{\pi_e\}$
8: **return** α_e

that does not search over transition probabilities for the best Whittle index, but simply returns the Whittle index of the given transition probabilities. This represents a new way to compute Whittle indices that could also be of general interest.

E.8 Evaluating Each Oracle

Planner oracle We verify empirically that the planner oracle, described in Section 6.4.1, produces high quality best responses, i.e., reward-maximizing RMAB intervention policies π, across various problem sizes and intervals. As baselines, we compare against: **No action (NA)** which simulates the policy that takes action $a = 0$ on all arms at all time steps, representing a lower bound on reward; **Random** which takes action $a = 1$ on K randomly chosen arms each round; and **Brute Force** which enumerates the entire feasible RMAB policy space, simulates the average reward of each policy, then returns the reward of the best-performing policy. Brute force can only be computed for small problem sizes, since its computation cost is exponential in N and K. We evaluate on test data generated by, for each seed, randomly sampling transition intervals $[\underline{P}^m_{s,a,s'}, \overline{P}^m_{s,a,s'}]$ $\forall c, s, a, s'$, and randomly sampling a mixed nature strategy. Results, shown in Fig. E.1, are reported as the aver-

Algorithm E.7.2 Compute Whittle Index (ComputeWI)

Input: Group m, state s_I, transition probabilities P^m for group m, tolerance ε.
Output: Whittle index $W^m(s_I)$

$ub, lb =$ InitBSBounds(P^m) {Return upper and lower bounds on $W^m(s_I)$ given P^m, e.g., 1, 0.}
{Now binary search for the Whittle index}
while $ub - lb > \varepsilon$ **do**
  $\lambda = \frac{ub+lb}{2}$
  $a =$ ValueIteration(P^m, s_I, λ) {Run value iteration for the MDP defined by P^m with λ-adjusted reward function $r(s, a, \lambda) = s - a\lambda$, and return corresponding $\pi^\star(s_I)$}
  if a=0 **then**
    $ub = \lambda$ {Charging too much, decrease}
  else if a=1 **then**
    $lb = \lambda$ {Can charge more, increase}
$W^m(s_I) = \frac{ub+lb}{2}$
return $W^m(s_I)$

age reward over ten random seeds. Our approach, WI4MS performs nearly as well as brute force for the small problem size, and outperforms all baselines as the problem size increases to the scale of the maternal health intervention problem.

Adversary oracle We verify empirically that the adversary oracle, described in Section 6.4.2, produces high quality best responses, i.e., regret-maximizing environments P, across various problem sizes and intervals. As baselines, we compare against: **Random** which selects an P by uniformly randomly sampling $P^m \in \overline{\underline{P}}^m \; \forall m \in [M]$ and **Brute Force** which (1) discretizes the adversary's pure strategy space into $D = 3$ uniformly spaced values for each interval $[\underline{P}^m_{s,a,s'}, \overline{P}^m_{s,a,s'}]$, (2) enumerates all possible combinations of the discrete environment setting, denoted P_d, (3) simulates the average regret induced by each P_d by simulating the optimal WIP against P_d and simulating some input planner mixed strategy α against P_d, then taking the difference, (4) then returns the regret of the best-performing P_d. Brute force can only be computed for small problem sizes, since its compu-

tation cost is exponential in N, K, and D. For instance, even for $N = 2, K = 1$, and $D = 3$ brute force enumerates ~60k choices of P_d. We evaluate on test data generated by randomly sample transition intervals $[\underline{P}^m_{s,a,s'}, \overline{P}^m_{s,a,s'}]$ $\forall c, s, a, s'$. Also, since our adversary oracle implementation requires as input both a planner mixed strategy α and its corresponding mixed adversary β of some MSNE (see Alg. 6.4.2), we generate inputs to the oracle by first randomly generating agent and adversary pure strategy sets according to the sampled $[\underline{P}^m_{s,a,s'}, \overline{P}^m_{s,a,s'}]$, then running one iteration of the double oracle using these strategy sets to generate the required α and β. Results, shown in Fig. E.2, are reported as the average reward over ten random seeds. Our approach, RegretMaxWI performs nearly as well as brute force for the small problem size, and vastly outperforms the naive random baseline even as the problem size increases to the scale of the maternal health intervention problem.

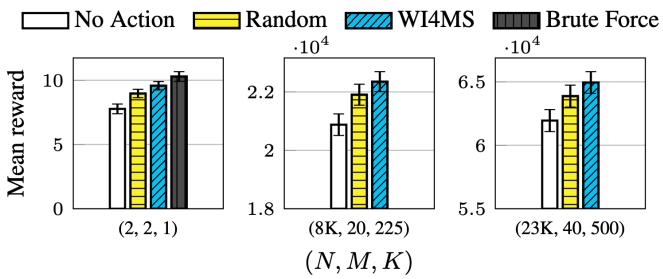


Figure E.1: Evaluating planner oracle best response quality (**WI4MS**). Objective is to maximize reward (higher is better). **No Action** simulates the policy that takes action $a = 0$ on all arms at all time steps, representing a lower bound on reward. **Random** takes action $a = 1$ on K randomly chosen arms each round. **Brute Force** enumerates the entire feasible RMAB policy space, simulates the average reward of each policy, then returns the reward of the best-performing policy. Brute force can only be shown for small problem sizes, since its computation cost is exponential in N and K. Our approach **WI4MS** performs well across all problem sizes.

E.9 Runtime Scalability of GROUPS

In Fig. E.3 we demonstrate the runtime improves achieved by our GROUPS robust planning method as the number of groups M decreases, with number of arms N held constant. Given Thm 6.5.3, this demonstrates that we can achieve large scaling up of our robust planning without losing perfor-

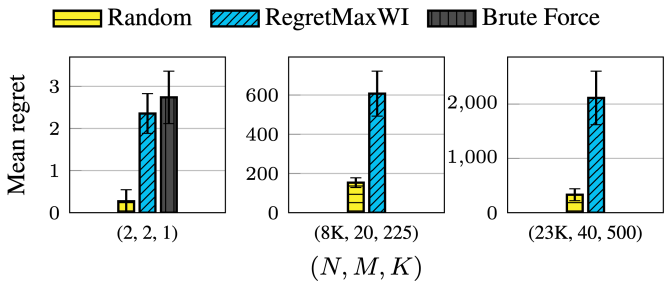


Figure E.2: Evaluating adversary oracle best response quality. Objective is to maximize regret (higher is better). Our approach, **RegretMaxWI**, performs nearly as well as a discretized brute force algorithm for small problems, and continues to perform far better than naive strategies for larger problems where brute force is intractable.

mance, under some mild assumptions, e.g., similarity of groups of arms.

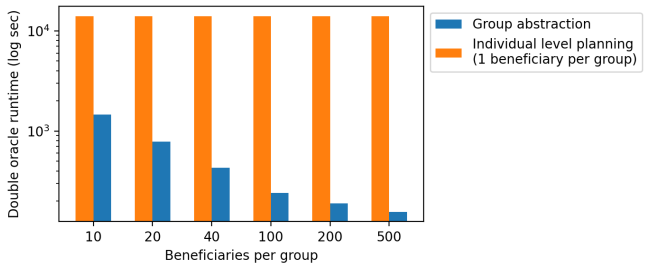


Figure E.3: Run time scalability of GROUPS. Holding the number of arms N constant, the runtime of GROUPS improves significantly as the number of groups M decreases.

E.10 Experiment Setup Details

All algorithms were implemented in Python 3.7.4 and mathematical programs were solved using Gurobi version 9.0.3 via the gurobipy interface[58]. Experiments were run on a cluster running CentOS with Intel(R) Xeon(R) CPU E5-2683 v4 @ 2.1 GHz with 8GB of RAM and four processors.

To compute regret, we simulate each planner strategy against the full set of the adversary's pure strategies (i.e., environment parameter settings, including both those computed by the adversary oracle as well as baseline responses pessimist, median, optimist, and random) to determine that which maximizes regret.

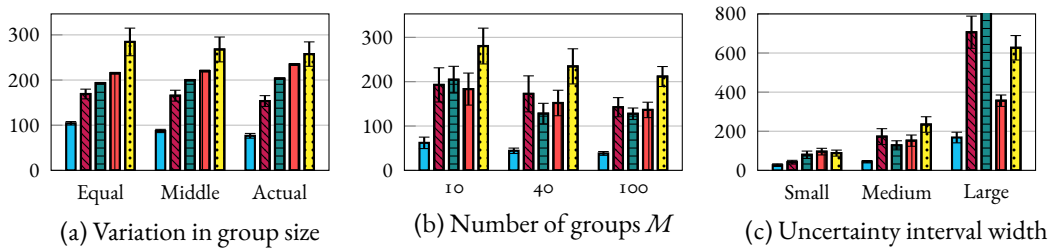


Figure E.4: More experimental results evaluating the regret (lower is better) incurred by GROUPS, our robust solution approach, compared to non-robust baselines across a variety of problem settings. Regret is interpreted, in real-world terms, as the maximum *preventable* missed messages across the uncertainty space. The problem setting used by ARMMAN, which we use as default values, are $K = 100$, $H = 10$, $N = 15{,}320$, *actual* group size, and $M = 40$ groups. Experiment (a) uses real data obtained from [104]; (b) and (c) use randomly sampled data, as described in section E.12. Each result is averaged over 30 random seeds.

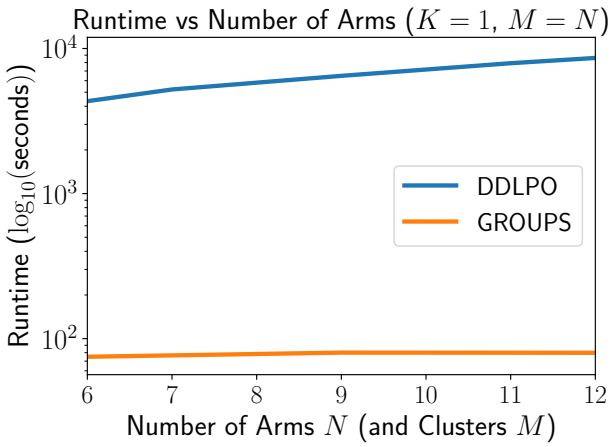


Figure E.5: Run time scaling of DDLPO [83] vs GROUPS. GROUPS is hundreds of times faster than DDLPO.

E.11 ARMMAN Consent for Data Collection and Analysis

In this section, we provide information about consent related to data collection, analyzing data, data usage and sharing. We highlight that this work is part of a long-standing research collaboration with ARMMAN, with continuous analysis of data performed in close consultation with ARMMAN researchers.

E.11.1 Secondary Analysis and Data Usage

This study falls into the category of secondary analysis. To evaluate the performance of the algorithms, we use unlinked, anonymized data generated during the course of implementation of the program, i.e., previously collected engagement trajectories of different beneficiaries participating in the service call program. The proposed algorithms are evaluated via a simulation-based method discussed in Section 6.6. This work does not involve deployment of the proposed algorithm or any other baselines to the service call program.

E.11.2 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 and restricted use of the data for research purposes only.

E.11.3 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.

E.12 Domain Descriptions

Maternal health The data used in chapter 6 are anonymized, aggregated summary statistics of engagement behavior of mothers enrolled in ARMMAN's mMtira program, and *does not* contain any demographic information.

From the full dataset of 23,008 mothers, we chose a subset of 15,336 mothers from this cohort who have a least one record of intervention, then computed summary statistics (i.e., frequentist transition probabilities) over that subset. To create an arm–group mapping, we run K-means clustering on those probabilities, and compute uncertainty intervals via bootstrapping followed by multiple imputation to compute standard deviations of the means[140].

We now describe how we set up the environment settings used for each simulation variant. As mentioned, the experiments in Figure 6.3(g)–(i) use the summary engagement statistics from ARMMAN. To vary the simulated number of mothers in Figure 6.3(i), we begin with the groupings from the summary data but scale up the number of mothers in each group by a factor of 10 and 20 to reach 153.2K and 306.4K mothers, respectively, with budget scaled accordingly to $K = 1000$ and $K = 7000$.

Statistics on the uncertainty intervals and group sizes for the summary ARMMAN dataset are displayed in Figures E.6 and E.7, respectively.

TB Derived from data obtained from[82], which contains anonymous records of daily adherence to tuberculosis (TB) medication. We used the 8,350 records with at least 30 days of adherence data. Though not collected in a grouped RMAB setting, we augmented the data to have groups by running K-Means grouping on the passive transition probabilities, and simulated uncertainty intervals for passive and active transitions as A_σ standard deviations of the mean of group centers. We then simulated uncertainty intervals: (1) for the passive transition probabilities as A_σ standard deviations

(default $A_\sigma = 3$) about each group center and (2) for the active transition probabilities by adding a random value $\eta \sim N(0.3, 0.3)$ to the corresponding passive transition (i.e., always preferable to act), and creating an uncertainty interval about the mean $1.5\times$ width of the passive uncertainty (i.e., less knowledge of active transitions vs. passive). These specific values were chosen to calibrate uncertainties to roughly match those of the maternal health domain, for which uncertainty statistics were available.

Statistics on the uncertainty intervals and group sizes for the TB dataset are displayed in Figures E.8 and E.9, respectively.

Synthetic A benchmark domain from Killian et al. [83] comprised of three "arm types" $[U, V, W]$, each with their own intervals, designed so that non-robust policies have higher regret than robust ones. Specifically,

$$T^n_{s=0} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}, \quad T^n_{s=1} = \begin{bmatrix} 1.0 & 0.0 \\ 1 - p_n & p_n \end{bmatrix} \tag{E.14}$$

$$\text{where} \quad \begin{aligned} p_U &\in [0.00, 1.00] \\ p_V &\in [0.05, 0.90] \\ p_W &\in [0.10, 0.95] \end{aligned} .$$

We augment the domain to allow homogeneous groups of each arm type, where the size and proportion of groups of each type may vary.

Randomly sampled data Used in Figure 6.3(b)–(c), we randomly generate transition probabilities and group sizes, drawn from normal distributions. We ensure that these transition probabilities are valid, that is that the probability transitioning to (or remaining in) the good state ($s' = 1$) with intervention ($a = 1$) is always higher than the probability of not intervening ($a = 0$), and

similarly that the probability when starting in the engaging state ($s = 1$) is always higher than the probability of starting in the not engaging state ($s = 0$).

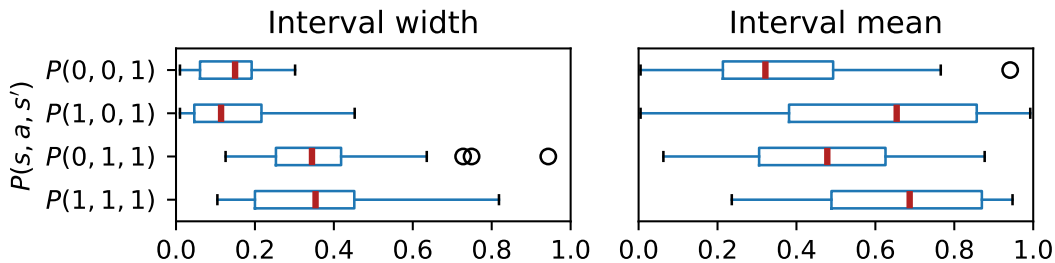


Figure E.6: Statistics on the uncertainty intervals from the ARMMAN data, averaged over 40 groups. Left shows the distribution of interval widths over all 40 groups. Right shows the distribution of interval midpoints over all 40 groups. Some uncertainty intervals are wider than 0.8, but the majority have width below 0.4. Uncertainty intervals of most groups are in the range $[0.3, 0.7]$.

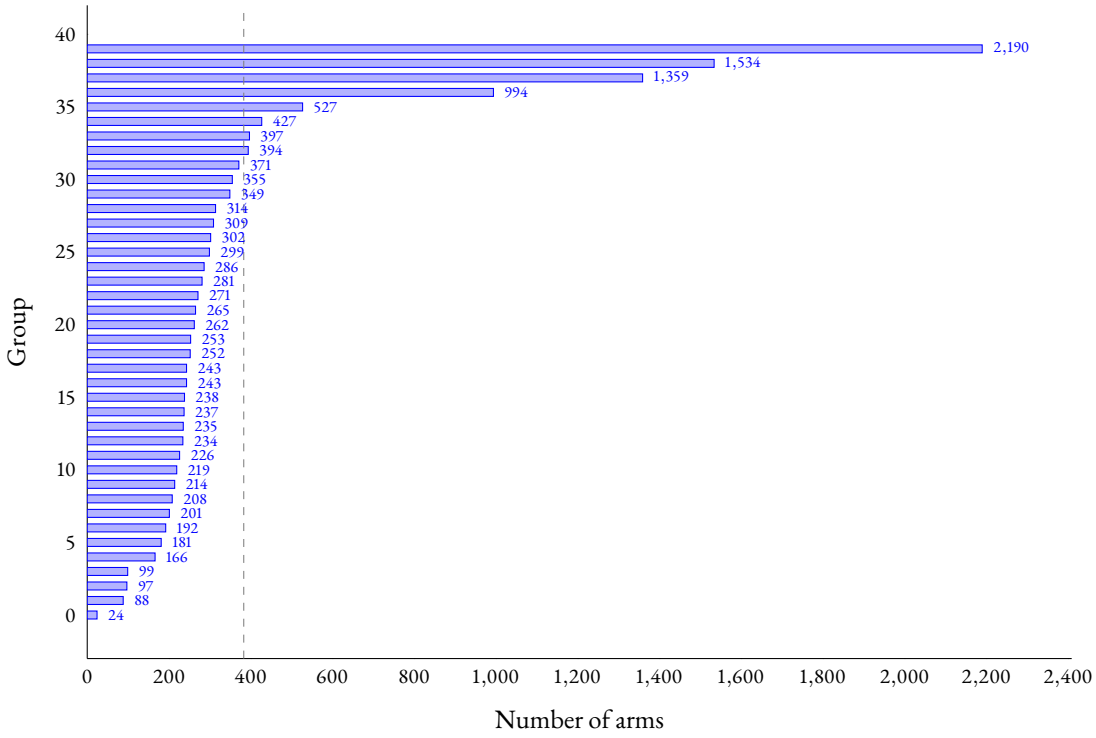


Figure E.7: Distribution of group sizes for the ARMMAN data.

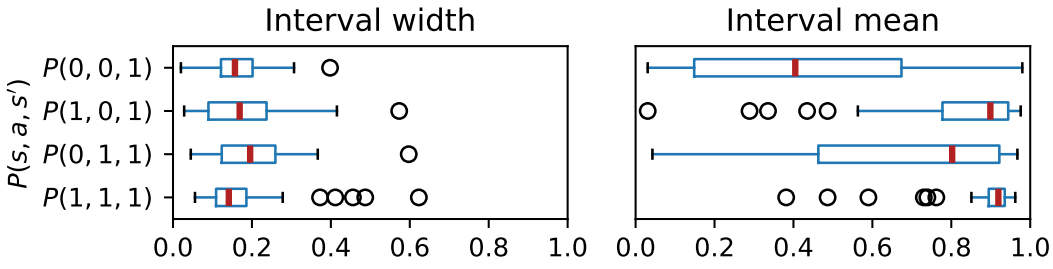


Figure E.8: Summary statistics on the uncertainty intervals from TB data obtained from[82], averaged over 60 groups. Left shows the distribution of interval widths over all groups. Right shows the distribution of interval midpoints over all groups. In general, transition probability medians are closer to 1 for this domain than the ARMMAN domain.

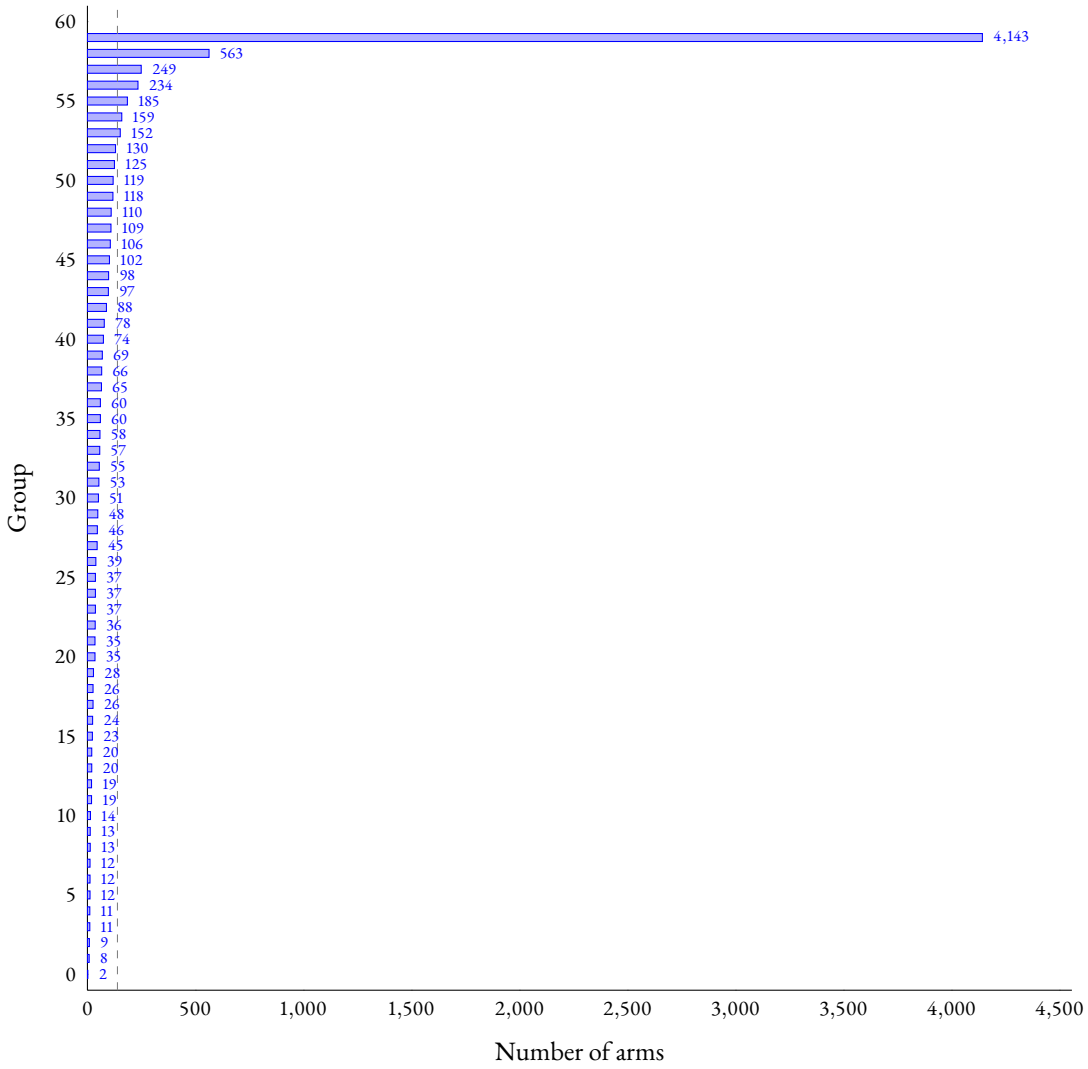


Figure E.9: Distribution of group sizes for TB adherence data.

References

[1] Abbou, A. & Makis, V. (2019). Group maintenance: A restless bandits approach. *INFORMS Journal on Computing*, 31(4), 719–731.

[2] Abounadi, J., Bertsekas, D., & Borkar, V. S. (2001). Learning algorithms for Markov decision processes with average cost. *SIAM Journal on Control and Optimization*, 40(3), 681–698.

[3] Achiam, J. (2018). Spinning Up in Deep Reinforcement Learning.

[4] Adelman, D. & Mersereau, A. J. (2008). Relaxations of weakly coupled stochastic dynamic programs. *Operations Research*, 56(3), 712–727.

[5] Ahmad, S. H. A. & Liu, M. (2009). Multi-channel opportunistic access: A case of restless bandits with multiple plays. In *2009 47th Annual Allerton Conference on Communication, Control, and Computing* (pp. 1361–1368).

[6] Akbarzadeh, N. & Mahajan, A. (2019). Restless bandits with controlled restarts: Indexability and computation of Whittle index. In *IEEE Conference on Decision and Control* (pp. 7294–7300).

[7] Al-Qaness, M. A. A., Saba, A. I., Elsheikh, A. H., Abd Elaziz, M., Ibrahim, R. A., Lu, S., Hemedan, A. A., Shanmugan, S., & Ewees, A. A. (2021). Efficient artificial intelligence forecasting models for COVID-19 outbreak in Russia and Brazil. *Process Safety and Environmental Protection*, 149, 399–409.

[8] Alam, M., D'Este, C., Banwell, C., & Lokuge, K. (2017). The impact of mobile phone based messages on maternal and child healthcare behaviour: a retrospective cross-sectional survey in Bangladesh. *BMC Health Serv. Res.*, 17(1).

[9] Ansell, P., Glazebrook, K. D., Nino-Mora, J., & O'Keeffe, M. (2003). Whittle's index policy for a multi-class queueing system with convex holding costs. *Mathematical Methods of Operations Research*, 57(1), 21–39.

[10] ARMMAN (2023). Research studies: Randomized cluster trial: mMitra and Arogya Sakhi. Accessed: 04-08-2023.

[11] Athey, S. & Imbens, G. (2016). Recursive partitioning for heterogeneous causal effects. *Proceedings of the National Academy of Sciences*, 113(27), 7353–7360.

[12] Avrachenkov, K. E. & Borkar, V. S. (2022). Whittle index based q-learning for restless bandits with average reward. *Automatica*, 139, 110186.

[13] 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.

[14] Bansal, G., Nushi, B., Kamar, E., Lasecki, W. S., Weld, D. S., & Horvitz, E. (2019a). Beyond accuracy: The role of mental models in human-ai team performance. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 7 (pp. 2–11).

[15] Bansal, G., Nushi, B., Kamar, E., Weld, D. S., Lasecki, W. S., & Horvitz, E. (2019b). Updates in human-ai teams: Understanding and addressing the performance/compatibility tradeoff. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33, 2429–2437.

[16] Bertsimas, D. & Niño-Mora, J. (2000). Restless bandits, linear programming relaxations, and a primal-dual index heuristic. *Operations Research*, 48(1), 80–90.

[17] Bhattacharya, B. (2018). Restless bandits visiting villages: A preliminary study on distributing public health services. In *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies* (pp. 1–8).

[18] Biswas, A., Aggarwal, G., Varakantham, P., & Tambe, M. (2021a). 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*.

[19] Biswas, A., Aggarwal, G., Varakantham, P., & Tambe, M. (2021b). Learning index policies for restless bandits with application to maternal healthcare. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems* (pp. 1467–1468).

[20] Bondi, E., Chen, H., Golden, C. D., Behari, N., & Tambe, M. (2022). Micronutrient deficiency prediction via publicly available satellite data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36 (pp. 12454–12460).

[21] Boutilier, C., Hsu, C.-W., Kveton, B., Mladenov, M., Szepesvári, C., & Zaheer, M. (2020). Differentiable meta-learning of bandit policies. In *Advances in Neural Information Processing Systems*, volume 33 (pp. 2122–2134).

[22] Box, G. E. & Draper, N. R. (1987). *Empirical model-building and response surfaces.* John Wiley & Sons.

[23] Braziunas, D. & Boutilier, C. (2007). Minimax regret based elicitation of generalized additive utilities. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence* (pp. 25–32).

[24] Brownstein, J. N., Chowdhury, F. M., Norris, S. L., Horsley, T., Jack Jr, L., Zhang, X., & Satterfield, D. (2007). Effectiveness of community health workers in the care of people with hypertension. *American Journal of Preventive Medicine*, 32(5), 435–447.

[25] Carvalho, N., Salehi, A., & Goldie, S. (2013). National and sub-national analysis of the health benefits and cost-effectiveness of strategies to reduce maternal mortality in Afghanistan. *Health Policy and Planning*, 28(1).

[26] Cassady, C. R. & Kutanoglu, E. (2005). Integrating preventive maintenance planning and production scheduling for a single machine. *IEEE Transactions on Reliability*, 54(2), 304–309.

[27] Chang, A. H., Polesky, A., & Bhatia, G. (2013). House calls by community health workers and public health nurses to improve adherence to isoniazid monotherapy for latent tuberculosis infection: A retrospective study. *BMC Public Health*, 13(1), 894.

[28] Channel, I. S. (2019). MBMC Recruitment 2018 for 04 District PPM Co-ordinator, TBHV Posts. https://www.indiastudychannel.com/jobs/428552-MBMC-Recruitment-2018-for-04-District-PPM-Co-ordinator-TBHV-Posts.aspx. Accessed: 02-03-2019.

[29] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321–357.

[30] Chollet, F. et al. (2015). Keras. https://keras.io.

[31] Christopher, J. B., Le May, A., Lewin, S., & Ross, D. A. (2011). Thirty years after Alma-Ata: A systematic review of the impact of community health workers delivering curative interventions against malaria, pneumonia and diarrhoea on child mortality and morbidity in sub-Saharan Africa. *Human Resources For Health*, 9(1).

[32] Cohen, K., Zhao, Q., & Scaglione, A. (2014). Restless multi-armed bandits under time-varying activation constraints for dynamic spectrum access. In *2014 48th Asilomar Conference on Signals, Systems and Computers* (pp. 1575–1578).

[33] Cook, P. F., Schmiege, S. J., Starr, W., Carrington, J. M., & Bradley-Springer, L. (2017). Prospective state and trait predictors of daily medication adherence behavior in HIV. *Nursing research*, 66(4), 275–285.

[34] Corden, M. E., Koucky, E. M., Brenner, C., Palac, H. L., Soren, A., Begale, M., Ruo, B., Kaiser, S. M., Duffecy, J., & Mohr, D. C. (2016). MedLink: A mobile intervention to improve medication adherence and processes of care for treatment of depression in general medicine. *Digital Health*, 2, 2055207616663069.

[35] Cordwell, S. A. (2015). Python Markov decision process (MDP) toolbox. PyPi, https://pypi.org/project/pymdptoolbox/.

[36] Côté, J., Fortin, M.-C., Auger, P., Rouleau, G., Dubois, S., Boudreau, N., Vaillant, I., & Gélinas-Lemay, É. (2018). Web-based tailored intervention to support optimal medication adherence among kidney transplant recipients: Pilot parallel-group randomized controlled trial. *JMIR Formative Research*, 2(2), e14.

[37] Cross, A., Gupta, N., Liu, B., Nair, V., Kumar, A., Kuttan, R., Ivatury, P., Chen, A., Lakshman, K., Rodrigues, R., et al. (2019). 99DOTS: A low-cost approach to monitoring and improving medication adherence. In *Proceedings of the 10th International Conference on Information and Communication Technologies and Development* (pp.15).

[38] Dehejia, R. H. & Wahba, S. (2002). Propensity score-matching methods for nonexperimental causal studies. *Review of Economics and Statistics*, 84(1), 151–161.

[39] Demonceau, J., Ruppar, T., Kristanto, P., Hughes, D. A., Fargher, E., Kardas, P., De Geest, S., Dobbels, F., Lewek, P., Urquhart, J., et al. (2013). Identification and assessment of adherence-enhancing interventions in studies assessing medication adherence through electronically compiled drug dosing histories: A systematic literature review and meta-analysis. *Drugs*, 73(6), 545–562.

[40] Dil, Y., Strachan, D., Cairncross, S., Korkor, A., & Hill, Z. (2012). Motivations and challenges of community-based surveillance volunteers in the northern region of Ghana. *Journal of Community Health*, 37(6), 1192–1198.

[41] Donti, P., Amos, B., & Kolter, J. Z. (2017). Task-based end-to-end model learning in stochastic optimization. In *Advances in Neural Information Processing Systems* (pp. 5484–5494).

[42] Dutta, P. K. (1991). What do discounted optima converge to?: A theory of discount rate asymptotics in economic models. *Journal of Economic Theory*, 55(1), 64–94.

[43] Elazan, S., Higgins-Steele, A., Fotso, J., Rosenthal, M., & Rout, D. (2016). Reproductive, maternal, newborn, and child health in the community: Task-sharing between male and female health workers in an Indian rural context. *Indian Journal of Community Medicine*, 41(1), 34.

[44] Evans, W. D., Wallace, J. L., & Snider, J. (2012). Pilot evaluation of the text4baby mobile health program. *BMC Public Health*, 12(1).

[45] Everwell (2019). Everwell. http://www.everwell.org/. Accessed: 01-29-2019.

[46] 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).

[47] Gafni, T. & Cohen, K. (2020). Learning in restless multi-armed bandits via adaptive arm sequencing rules. *IEEE Transactions on Automatic Control*.

[48] Gardner, R. (2017). Convex functions. https://faculty.etsu.edu/gardnerr/5210/Beamer-Proofs/Proofs-6-6-print.pdf. Accessed: 2021-01-15.

[49] Garfein, R. S., Collins, K., Muñoz, F., Moser, K., Cerecer-Callu, P., Raab, F., Rios, P., Flick, A., Zúñiga, M. L., Cuevas-Mota, J., et al. (2015). Feasibility of tuberculosis treatment monitoring by video directly observed therapy: A binational pilot study. *The International Journal of Tuberculosis and Lung Disease*, 19(9), 1057–1064.

[50] Gates Foundation (2021). Global progress and projections for maternal mortality. https://www.gatesfoundation.org/goalkeepers/report/2021-report/progress-indicators/maternal-mortality/. Accessed: 2023-03-30.

[51] Gilbert, H. & Spanjaard, O. (2017). A double oracle approach to minmax regret optimization problems with interval data. *European Journal of Operational Research*, 262(3), 929–943.

[52] Glazebrook, K. D., Hodge, D. J., & Kirkbride, C. (2011a). General notions of indexability for queueing control and asset management. *The Annals of Applied Probability*, 21(3), 876–907.

[53] Glazebrook, K. D., Hodge, D. J., & Kirkbride, C. (2011b). General notions of indexability for queueing control and asset management. *Annals of Applied Probability*, 21(3), 876–907.

[54] Glazebrook, K. D., Ruiz-Hernandez, D., & Kirkbride, C. (2006). Some indexable families of restless bandit problems. *Advances in Applied Probability*, 38(3), 643–672.

[55] Gocgun, Y. & Ghate, A. (2012). Lagrangian relaxation and constraint generation for allocation and advanced scheduling. *Computers & Operations Research*, 39(10), 2323–2336.

[56] Green, H. & Riddington, A. (2020). Gender inclusive language in perinatal services: Mission statement and rationale. *Brighton, England: Brighton and Sussex University Hospitals*.

[57] Gribble, K. D., Bewley, S., Bartick, M. C., Mathisen, R., Walker, S., Gamble, J., Bergman, N. J., Gupta, A., Hocking, J. J., & Dahlen, H. G. (2022). Effective communication about pregnancy, birth, lactation, breastfeeding and newborn care: The importance of sexed language. *Frontiers in Global Women's Health*, (pp.3).

[58] Gurobi Optimization, L. (2021). Gurobi optimizer reference manual.

[59] Haberer, J. E., Musinguzi, N., Tsai, A. C., Boum, Y., Bwana, B. M., Muzoora, C., Hunt, P. W., Martin, J. N., & Bangsberg, D. R. (2017). Real-time electronic adherence monitoring plus follow-up improves adherence compared with standard electronic adherence monitoring. *AIDS (London, England)*, 31(1), 169–171.

[60] Hawkins, J. T. (2003). *A Langrangian decomposition approach to weakly coupled dynamic optimization problems and its applications*. PhD thesis, Massachusetts Institute of Technology.

[61] He, T., Chen, S., Kim, H., Tong, L., & Lee, K.-W. (2012). Scheduling parallel tasks onto opportunistically available cloud resources. In *2012 IEEE 5th International Conference on Cloud Computing* (pp. 180–187).

[62] Hinch, R., Probert, W. J., Nurtay, A., Kendall, M., Wymant, C., Hall, M., Lythgoe, K., Bulas Cruz, A., Zhao, L., Stewart, A., et al. (2021). OpenABM-Covid19—an agent-based model for non-pharmaceutical interventions against COVID-19 including contact tracing. *PLOS Computational Biology*, 17(7), e1009146.

[63] Hodge, D. J. & Glazebrook, K. D. (2015). On the asymptotic optimality of greedy index heuristics for multi-action restless bandits. *Advances in Applied Probability*, 47(3), 652–667.

[64] Hsu, Y. (2018). Age of information: Whittle index for scheduling stochastic arrivals. In *2018 IEEE International Symposium on Information Theory (ISIT)* (pp. 2634–2638).

[65] Huo, X. & Fu, F. (2017). Risk-aware multi-armed bandit problem with application to portfolio selection. *Royal Society Open Science*, 4(11), 171377.

[66] Hussain, O. A. & Junejo, K. N. (2018). Predicting treatment outcome of drug-susceptible tuberculosis patients using machine-learning models. *Informatics for Health and Social Care*, (pp. 1–17).

[67] Iannello, F., Simeone, O., & Spagnolini, U. (2012a). Optimality of myopic scheduling and Whittle indexability for energy harvesting sensors. In *2012 46th Annual Conference on Information Sciences and Systems (CISS)* (pp. 1–6).

[68] Iannello, F., Simeone, O., & Spagnolini, U. (2012b). Optimality of myopic scheduling and Whittle indexability for energy harvesting sensors. In *2012 46th Annual Conference on Information Sciences and Systems* (pp. 1–6).: IEEE.

[69] Jiang, S., Song, Z., Weinstein, O., & Zhang, H. (2021). A faster algorithm for solving general lps. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing* (pp. 823–832).

[70] Johnson & Johnson (2017). MomConnect: Connecting women to care, one text at a time. https://www.jnj.com/our-giving/momconnect-connecting-women-to-care-one-text-at-a-time. Accessed: 2023-03-30.

[71] Jung, Y. H., Abeille, M., & Tewari, A. (2019). Thompson sampling in non-episodic restless bandits. *arXiv preprint arXiv:1910.05654*.

[72] Jung, Y. H. & Tewari, A. (2019). Regret bounds for Thompson sampling in episodic restless bandit problems. In *Advances in Neural Information Processing Systems*, volume 32.

[73] Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2), 99–134.

[74] Kalhori, S. R. N. & Zeng, X.-J. (2013). Evaluation and comparison of different machine learning methods to predict outcome of tuberculosis treatment course. *Journal of Intelligent Learning Systems and Applications*, 5(03), 184.

[75] Kardas, P., Lewek, P., & Matyjaszczyk, M. (2013). Determinants of patient adherence: A review of systematic reviews. *Frontiers in Pharmacology*, 4, 91.

[76] Kenya, S., Chida, N., Symes, S., & Shor-Posner, G. (2011). Can community health workers improve adherence to highly active antiretroviral therapy in the USA? A review of the literature. *HIV Medicine*, 12(9), 525–534.

[77] Kenya, S., Jones, J., Arheart, K., Kobetz, E., Chida, N., Baer, S., Powell, A., Symes, S., Hunte, T., Monroe, A., et al. (2013). Using community health workers to improve clinical outcomes among people living with HIV: A randomized controlled trial. *AIDS and Behavior*, 17(9), 2927–2934.

[78] Kerr, C. C., Stuart, R. M., Mistry, D., Abeysuriya, R. G., Rosenfeld, K., Hart, G. R., Núñez, R. C., Cohen, J. A., Selvaraj, P., Hagedorn, B., et al. (2021). Covasim: an agent-based model of COVID-19 dynamics and interventions. *PLOS Computational Biology*, 17(7), e1009149.

[79] Khalil, E., Dai, H., Zhang, Y., Dilkina, B., & Song, L. (2017). Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems* (pp. 6348–6358).

[80] Killian, J., Biswas, A., Shah, S., & Tambe, M. (2021a). Q-learning lagrange policies for multi-action restless bandits. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (pp. 871–881).

[81] Killian, J. A., Perrault, A., & Tambe, M. (2021b). Beyond "To act or not to act": Fast lagrangian approaches to general multi-action restless bandits. In *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems*.

[82] 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).

[83] Killian, J. A., Xu, L., Biswas, A., & Tambe, M. (2022). Robust restless bandits: Tackling interval uncertainty with deep reinforcement learning. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 180 (pp. 990–1000).

[84] Kim, K., Kim, B., Chung, A. J., Kwon, K., Choi, E., & Nah, J.-w. (2018). Algorithm and system for improving the medication adherence of tuberculosis patients. In *2018 International Conference on Information and Communication Technology Convergence (ICTC)* (pp. 914–916).

[85] Kliiman, K. & Altraja, A. (2010). Predictors and mortality associated with treatment default in pulmonary tuberculosis. *The International Journal of Tuberculosis and Lung Disease*, 14(4), 454–463.

[86] Knight, V. & Campbell, J. (2018). Nashpy: A Python library for the computation of Nash equilibria. *Journal of Open Source Software*, 3(30), 904.

[87] Kool, W., Van Hoof, H., & Welling, M. (2019). Attention, learn to solve routing problems! In *International Conference on Learning Representations*.

[88] Kuleshov, V. & Precup, D. (2000). Algorithms for multi-armed bandit problems. *Journal of Machine Learning Research*, 1, 1–48.

[89] Lakshminarayanan, C. & Bhatnagar, S. (2017). A stability criterion for two timescale stochastic approximation schemes. *Automatica*, 79, 108–114.

[90] Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Pérolat, J., Silver, D., & Graepel, T. (2017). A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 30 (pp. 4190–4203).

[91] Le Ny, J., Dahleh, M., & Feron, E. (2008). Multi-UAV dynamic routing with partial observations using restless bandit allocation indices. In *2008 American Control Conference* (pp. 4220–4225).

[92] Lee, E., Lavieri, M. S., & Volk, M. (2019). Optimal screening for hepatocellular carcinoma: A restless bandit model. *Manufacturing & Service Operations Management*, 21(1), 198–212.

[93] Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17), 1–5.

[94] Li, S., Wu, Y., Cui, X., Dong, H., Fang, F., & Russell, S. (2019). Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33 (pp. 4213–4220).

[95] Lin, Y., Liu, S., & Huang, S. (2018). Selective sensing of a heterogeneous population of units with dynamic health conditions. *IISE Transactions*, 50(12), 1076–1088.

[96] 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.

[97] Liu, X., Lewis, J. J., Zhang, H., Lu, W., Zhang, S., Zheng, G., Bai, L., Li, J., Li, X., Chen, H., et al. (2015). Effectiveness of electronic reminders to improve medication adherence in tuberculosis patients: A cluster-randomised trial. *PLOS Medicine*, 12(9), e1001876.

[98] Löwe, B., Unützer, J., Callahan, C., Perkins, A., & Kroenke, K. (2004). Monitoring depression treatment outcomes with the patient health questionnaire-9. *Medical Care*, (pp. 1194–1201).

[99] Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., & Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, volume 30.

[100] Lundberg, S. (2019). A unified approach to explain the output of any machine learning model. https://github.com/slundberg/shap. Accessed: 01-18-2019.

[101] Mahajan, A. & Teneketzis, D. (2008). Multi-armed bandit problems. In *Foundations and Applications of Sensor Management* (pp. 121–151). Springer.

[102] Maillard, O.-A. (2013). Robust risk-averse stochastic multi-armed bandits. In *Proceedings of the 24th International Conference Algorithmic Learning Theory* (pp. 218–233).

[103] Mate, A., Killian, J., Xu, H., Perrault, A., & Tambe, M. (2020). Collapsing bandits and their application to public health intervention. In *Advances in Neural Information Processing Systems*, volume 33 (pp. 15639–15650).

[104] Mate, A., Madaan, L., Taneja, A., Madhiwalla, N., Verma, S., Singh, G., Hegde, A., Varakantham, P., & Tambe, M. (2022). 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*, volume 36 (pp. 12017–12025).

[105] Mburu, J. W., Kingwara, L., Ester, M., & Andrew, N. (2018). Use of classification and regression tree (CART), to identify hemoglobin A1C (HbA1C) cut-off thresholds predictive of poor tuberculosis treatment outcomes and associated risk factors. *Journal of Clinical Tuberculosis and Other Mycobacterial Diseases*, 11, 10–16.

[106] McMahan, H. B., Gordon, G. J., & Blum, A. (2003). Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the 20th International Conference on Machine Learning* (pp. 536–543).

[107] Meh, C., Sharma, A., Ram, U., Fadel, S., Correa, N., Snelgrove, J. W., Shah, P., Begum, R., Shah, M., Hana, T., et al. (2022). Trends in maternal mortality in India over two decades in nationally representative surveys. *BJOG: An International Journal of Obstetrics & Gynaecology*, 129(4), 550–561.

[108] Meshram, R. & Kaza, K. (2020). Simulation based algorithms for Markov decision processes and multi-action restless bandits. *arXiv preprint arXiv:2007.12933*.

[109] Meshram, R., Manjunath, D., & Gopalan, A. (2018). On the Whittle index for restless multi-armed hidden Markov bandits. *IEEE Transactions on Automatic Control*, 63(9), 3046–3053.

[110] Meuleau, N., Hauskrecht, M., Kim, K.-E., Peshkin, L., Kaelbling, L. P., Dean, T. L., & Boutilier, C. (1998). Solving very large weakly coupled Markov decision processes. In *Proceedings of the 15th National / 10th Conference on Artificial Intelligence / Innovative Applications of Artificial Intelligence* (pp. 165–172).

[111] Min, S., Moallemi, C. C., & Russo, D. J. (2020). Policy gradient optimization of Thompson sampling policies. *arXiv preprint arXiv:2006.16507*.

[112] Modi, N., Mary, P., & Moy, C. (2019). Transfer restless multi-armed bandit policy for energy-efficient heterogeneous cellular network. *EURASIP Journal on Advances in Signal Processing*, 2019(1), 46.

[113] Morgan, S. L. & Winship, C. (2014). *Counterfactuals and causal inference*. Cambridge University Press.

[114] Mundorf, C., Shankar, A., Moran, T., Heller, S., Hassan, A., Harville, E., & Lichtveld, M. (2018). Reducing the risk of postpartum depression in a low-income community through a community health worker intervention. *Maternal and Child Health Journal*, 22(4), 520–528.

[115] Murthy, N., Chandrasekharan, S., Prakash, M. P., Kaonga, N. N., Peter, J., Ganju, A., & Mechael, P. N. (2019). The impact of an mHealth voice message service (mMitra) on infant care knowledge, and practices among low-income women in India: Findings from a pseudo-randomized controlled trial. *Maternal and Child Health Journal*, 23(12), 1658–1669.

[116] Musiimenta, A., Tumuhimbise, W., Pinkwart, N., Katusiime, J., Mugyenyi, G., & Atukunda, E. C. (2021). A mobile phone-based multimedia intervention to support maternal health is acceptable and feasible among illiterate pregnant women in Uganda. *Digital Health*, 7.

[117] Nakhleh, K., Ganji, S., Hsieh, P.-C., Hou, I.-H., & Shakkottai, S. (2021). NeurWIN: Neural Whittle index network for restless bandits via deep RL. In *Advances in Neural Information Processing Systems*, volume 34 (pp. 828–839).

[118] Newman, P., Franke, M., Arrieta, J., Carrasco, H., Elliott, P., Flores, H., Friedman, A., Graham, S., Martinez, L., Palazuelos, L., et al. (2018). Community health workers improve disease control and medication adherence among patients with diabetes and/or hypertension in Chiapas, Mexico: An observational stepped-wedge study. *BMJ Global Health*, 3(1), e000566.

[119] Nguyen, T. H., Sinha, A., Gholami, S., Plumptre, A., Joppa, L., Tambe, M., Driciru, M., Wanyama, F., Rwetsiba, A., Critchlow, R., & Beale, C. M. (2016). CAPTURE: A new predictive anti-poaching tool for wildlife protection. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems* (pp. 767–775).

[120] Nguyen, T. H., Yadav, A., An, B., Tambe, M., & Boutilier, C. (2014). Regret-based optimization and preference elicitation for Stackelberg security games with uncertainty. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28 (pp. 756–762).

[121] Nino-Mora, J. (2001). Restless bandits, partial conservation laws and indexability. *Advances in Applied Probability*, (pp. 76–98).

[122] Papadimitriou, C. H. & Tsitsiklis, J. N. (1999). The complexity of optimal queuing network control. *Mathematics of Operations Research*, 24(2), 293–305.

[123] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32 (pp. 8024–8035).

[124] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

[125] Pellowski, J. A., Kalichman, S. C., Kalichman, M. O., & Cherry, C. (2016). Alcohol-antiretroviral therapy interactive toxicity beliefs and daily medication adherence and alcohol use among people living with HIV. *AIDS Care*, 28(8), 963–970.

[126] Pinto, L., Davidson, J., Sukthankar, R., & Gupta, A. (2017). Robust adversarial reinforcement learning. In *International Conference on Machine Learning* (pp. 2817–2826).

[127] Platt, A. B., Localio, A. R., Brensinger, C. M., Cruess, D. G., Christie, J. D., Gross, R., Parker, C. S., Price, M., Metlay, J. P., Cohen, A., et al. (2010). Can we predict daily adher-

ence to warfarin?: Results from the international normalized ratio adherence and genetics (in-range) study. *Chest*, 137(4), 883–889.

[128] Pratt, B., Wild, V., Barasa, E., Kamuya, D., Gilson, L., Hendl, T., & Molyneux, S. (2020). Justice: A key consideration in health policy and systems research ethics. *BMJ Global Health*, 5(4), e001942.

[129] Puterman, M. L. (2014). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

[130] 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).

[131] Qin, Z. Z., Ahmed, S., Sarker, M. S., Paul, K., Adel, A. S. S., Naheyan, T., Barrett, R., Banu, S., & Creswell, J. (2021). Tuberculosis detection from chest x-rays for triaging in a high tuberculosis-burden setting: An evaluation of five artificial intelligence algorithms. *The Lancet Digital Health*, 3(9), e543–e554.

[132] Rahedi Ong'ang'o, J., Mwachari, C., Kipruto, H., & Karanja, S. (2014). The effects on tuberculosis treatment adherence from utilising community health workers: A comparison of selected rural and urban settings in Kenya. *PLOS One*, 9(2).

[133] Rioux, C., Weedon, S., London-Nadeau, K., Paré, A., Juster, R.-P., Roos, L. E., Freeman, M., & Tomfohr-Madsen, L. M. (2022). Gender-inclusive writing for epidemiological research on pregnancy. *Journal of Epidemiology and Community Healthh*, 76(9), 823–827.

[134] RNTCP (2019a). Revised national TB control programme: Technical and operational guidelines for tuberculosis control in India 2016. https://tbcindia.gov.in/index1.php?lang=1&level=2&sublinkid=4573&lid=3177. Accessed: 02-02-2019.

[135] RNTCP (2019b). Technical and operational guidelines for tuberculosis control. http://www.tbonline.info/media/uploads/documents/technical_and_operational_guidelines_for_tuberculosis_control_%282005%29.pdf. Accessed: 02-02-2019.

[136] Roy, N., Basu, M., Das, S., Mandal, A., Dutt, D., & Dasgupta, S. (2015). Risk factors associated with default among tuberculosis patients in Darjeeling district of West Bengal, India. *Journal of Family Medicine and Primary Care*, 4(3), 388.

[137] Ruiz-Hernández, D., Pinar-Pérez, J. M., & Delgado-Gómez, D. (2020). Multi-machine preventive maintenance scheduling with imperfect interventions: A restless bandit approach. *Computers & Operations Research*, 119, 104927.

[138] Sabin, L. L., DeSilva, M. B., Gill, C. J., Li, Z., Vian, T., Wubin, X., Feng, C., Keyi, X., Guanghua, L., Haberer, J. E., et al. (2015). Improving adherence to antiretroviral therapy

with triggered real-time text message reminders: The China adherence through technology study (CATS). *Journal of Acquired Immune Deficiency Syndromes*, 69(5), 551.

[139] Sauer, C. M., Sasson, D., Paik, K. E., McCague, N., Celi, L. A., Fernandez, I. S., & Illigens, B. M. (2018). Feature selection and prediction of treatment failure in tuberculosis. *PLOS One*, 13(11), e0207491.

[140] Schomaker, M. & Heumann, C. (2018). Bootstrap inference when using multiple imputation. *Statistics in Medicine*, 37(14), 2252–2266.

[141] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[142] Shargie, E. B. & Lindtjørn, B. (2007). Determinants of treatment adherence among smear-positive pulmonary tuberculosis patients in Southern Ethiopia. *PLOS Medicine*, 4(2), e37.

[143] Shin, S., Furin, J., Bayona, J., Mate, K., Kim, J. Y., & Farmer, P. (2004). Community-based treatment of multidrug-resistant tuberculosis in Lima, Peru: 7 years of experience. *Social Science and Medicine*, 59(7), 1529–1539.

[144] Sombabu, B., Mate, A., Manjunath, D., & Moharir, S. (2020a). Whittle index for AoI-aware scheduling. In *IEEE International Conference on Communication Systems & Networks (COSMSNETS)* (pp. 630–633).

[145] Sombabu, B., Mate, A., Manjunath, D., & Moharir, S. (2020b). Whittle index for AoI-aware scheduling. In *2020 12th International Conference on Communication Systems & Networks*: IEEE.

[146] Sondik, E. (1978). The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs. *Operations Research*, 26(2), 282–304.

[147] Song, H., Jang, H., Tran, H. H., Yoon, S.-e., Son, K., Yun, D., Chung, H., & Yi, Y. (2019). Solving continual combinatorial selection via deep reinforcement learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence* (pp. 3467–3474).

[148] Subbaraman, R., de Mondesert, L., Musiimenta, A., Pai, M., Mayer, K. H., Thomas, B. E., & Haberer, J. (2018). Digital adherence technologies for the management of tuberculosis therapy: Mapping the landscape and research priorities. *BMJ Global Health*, 3(5), e001018.

[149] Sutton, R. S., Maei, H. R., & Szepesvári, C. (2009). A convergent O(n) temporal-difference algorithm for off-policy learning with linear function approximation. In *Advances in Neural Information Processing Systems*, volume 21 (pp. 1609–1616).

[150] Tamrat, T. & Kachnowski, S. (2012). Special delivery: An analysis of mHealth in maternal and newborn health programs and their outcomes around the world. *Matern Child Health Journal*, 16(5).

[151] Thomas, A., Gopi, P., Santha, T., Chandrasekaran, V., Subramani, R., Selvakumar, N., Eusuff, S., Sadacharam, K., & Narayanan, P. (2005). Predictors of relapse among pulmonary tuberculosis patients treated in a DOTS programme in South India. *The International Journal of Tuberculosis and Lung Disease*, 9(5), 556–561.

[152] Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4), 285–294.

[153] United Nations (2021). Sustainable development goal 3: Ensure healthy lives and promote well-being for all at all ages. https://sdgs.un.org/goals/goal3. Accessed: 2023-04-05.

[154] Verloop, I. M. (2016). Asymptotically optimal priority policies for indexable and nonindexable restless bandits. *The Annals of Applied Probability*, 26(4), 1947–1995.

[155] Villar, S. S. (2016). Indexability and optimal index policies for a class of reinitialising restless bandits. *Probability in the Engineering and Informational Sciences*, 30(1), 1–23.

[156] Wang, K., Xu, L., Taneja, A., & Tambe, M. (2023). Optimistic Whittle index policy: Online learning for restless bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[157] Wang, S., Huang, L., & Lui, J. (2020). Restless-UCB, an efficient and low-complexity algorithm for online restless bandits. In *Advances in Neural Information Processing Systems*, volume 33 (pp. 11878–11889).

[158] Watkins, C. J. & Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4), 279–292.

[159] Watkins, C. J. C. H. (1989). Learning from delayed rewards. *PhD Thesis*.

[160] Watterson, J. L., Walsh, J., & Madeka, I. (2015). Using mHealth to improve usage of antenatal care, postnatal care, and immunization: A systematic review of the literature. *BioMed Research International*.

[161] Weber, R. R. & Weiss, G. (1990). On an index policy for restless bandits. *Journal of Applied Probability*, 27(3), 637–648.

[162] Wells, K. J., Luque, J. S., Miladinovic, B., Vargas, N., Asvat, Y., Roetzheim, R. G., & Kumar, A. (2011). Do community health worker interventions improve rates of screening mammography in the United States? A systematic review. *Cancer Epidemiology, Biomarkers & Prevention*, 20(8), 1580–1598.

[163] Whittle, P. (1988). Restless bandits: Activity allocation in a changing world. *Journal of Applied Probability*, 25(A), 287–298.

[164] WHO (2023). Maternal mortality. https://www.who.int/news-room/fact-sheets/detail/maternal-mortality. Accessed: 2023-03-30.

[165] 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).

[166] World Health Organization (2013). *Definitions and reporting framework for tuberculosis – 2013 revision*. Geneva: World Health Organization.

[167] World Health Organization (2018). *Global tuberculosis report 2018*. World Health Organization.

[168] World Health Organization et al. (2017). *Handbook for the use of digital technologies to support tuberculosis medication adherence*. World Health Organization.

[169] World Health Organization et al. (2018). *WHO guideline on health policy and system support to optimize community health worker programmes*. World Health Organization.

[170] Xu, L., Perrault, A., Fang, F., Chen, H., & Tambe, M. (2021). Robust reinforcement learning under minimax regret for green security. In *Uncertainty in Artificial Intelligence*, volume 161 (pp. 257––267).

[171] Yaesoubi, R. & Cohen, T. (2011). Generalized Markov models of infectious disease spread: A novel framework for developing dynamic health policies. *European Journal of Operational Research*, 215(3), 679–687.

[172] Yang, F., Zhu, Z., Zhao, S., Yang, Y., & Luo, X. (2018). Optimal task offloading in fog-enabled networks via index policies. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)* (pp. 688–692).: IEEE.

[173] Yellapu, G. D., Rudraraju, G., Sripada, N. R., Mamidgi, B., Jalukuru, C., Firmal, P., Yechuri, V., Varanasi, S., Peddireddi, V. S., Bhimarasetty, D. M., et al. (2023). Development and clinical validation of Swaasa AI platform for screening and prioritization of pulmonary TB. *Scientific Reports*, 13(1), 4740.

[174] Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., & Wu, Y. (2022). The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35, 24611–24624.

[175] Yu, Z., Xu, Y., & Tong, L. (2018). Deadline scheduling as restless bandits. *IEEE Transactions on Automatic Control*, 63(8), 2343–2358.

[176] Zayas-Caban, G., Jasin, S., & Wang, G. (2019). An asymptotically optimal heuristic for general nonstationary finite-horizon restless multi-armed, multi-action bandits. *Advances in Applied Probability*, 51(3), 745–772.

[177] Zhao, Q., Krishnamachari, B., & Liu, K. (2008). On myopic sensing for multi-channel opportunistic access: Structure, optimality, and performance. *IEEE Transactions on Wireless Communications*, 7(12), 5431–5440.