UNIVERSITY OF SOUTHAMPTON

Faculty of Engineering and Physical Sciences
School of Electronics and Computer Science
Agents, Interaction and Complexity Group

Accounting for Real World Phenomena in Machine Learning and Mechanism Design

by

Nicholas Bishop

MEng

ORCiD: 0000-0001-7062-9072

A thesis for the degree of
Doctor of Philosophy

February 2023

University of Southampton

Abstract

Faculty of Engineering and Physical Sciences
School of Electronics and Computer Science

Doctor of Philosophy

Accounting for Real World Phenomena in Machine Learning and Mechanism Design

by Nicholas Bishop

As data becomes more readily available, individuals and organisations are increasingly relying on automated systems to make decisions on their behalf. Both machine learning and mechanism design play key roles in the design of such systems. Machine learning is often deployed to learn complex decision rules that mimic or improve upon those adopted by humans. Meanwhile, mechanism design is often deployed to ensure that decision rules satisfy certain axiomatic properties of interest to the designer, such as fairness and incentive compatibility. Unfortunately, many real world settings fall outside the scope of traditional machine learning and mechanism design frameworks. This thesis investigates how approaches from mechanism design and machine learning can be rigorously extended and adapted for such settings to yield meaningful theoretical guarantees.

In particular, we investigate three problem domains; 1) linear regression in the presence of strategic agents, 2) sequential resource deployment with reusable resources and 3) repeated matching with reusable resources. For the first problem domain, we provide a theoretical framework based on Stackelberg predictions games. When the incentives of agents can be captured by a square loss function, we provide a polynomial time algorithm minimising Stackelberg risk, a natural analog to risk in classical supervised learning. For the second problem domain, we introduce a new multi-armed bandit model, called the adversarial blocking bandit problem, which incorporates nonstationary reward sequences and resource unavailability. In particular, we provide finite-time regret guarantees for this setting, by benchmarking against an oracle algorithm which approximates the optimal arm pulling policy. Lastly, for the third problem domain, we introduce a new sequential matching setting, in which a central planner is tasked with constructing matchings repeatedly through time under the assumption that some goods or services may become temporarily unavailable once assigned. Motivated by the random serial dictatorship algorithm, we construct an algorithm for the setting which is approximately truthful and approximately maximises social welfare.

Contents

List of Figures

List of Tables

Declaration of Authorship

I declare that this thesis and the work presented in it is my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Parts of this work have been published as:
   Nicholas Bishop, Long Tran-Thanh, and Enrico Gerding. Optimal learning from verified training data. In *Advances in Neural Information Processing Systems*, volume 33, pages 9520–9529. Curran Associates, Inc., 2020b
   Nicholas Bishop, Hau Chan, Debmalya Mandal, and Long Tran-Thanh. Adversarial blocking bandits. In *Advances in Neural Information Processing Systems*, volume 33, pages 8139–8149. Curran Associates, Inc., 2020a
   Nicholas Bishop, Hau Chan, Debmalya Mandal, and Long Tran-Thanh. Sequential blocked matching. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(5): 4834–4842, 2022

Signed:..... Date:.....

Acknowledgements

It is surreal to think my time at the University of Southampton is coming to a close. Having started many years ago as an undergraduate in mathematics, obtaining a masters degree in computer science, and finally embarking on this journey as a graduate student, I have had the opportunity to meet many talented individuals of great character. First and foremost amongst these people are my supervisors, Long Tran-Thanh and Enrico Gerding. I would like to thank them for their sage advice and guidance throughout my PhD studies, as well as their faith in my research, which was at times stronger in my own!

I would also like to thank Debmalya Mandal, and Hau Chan, who took the time to offer me guidance and mentorship under no obligation. By working with them, I learned how to translate concepts and ideas into rigor and proof. In a similar vein, I would like to thank my other co-authors at the University of Southampton: David Bossens, Le Cong Dinh, and Tom Davies. Aside from being a good friend, David introduced me to the field of safe reinforcement learning. Likewise, Tom introduced me to the beautiful, but admittedly terrifying, world of algebraic topology. Cong has been the source of many fruitful discussions about online learning and bandit algorithms as well as a great travel companion. More generally, I would like to thank all my colleagues within the Agents, Interaction and Complexity Group, who made the dull days more tolerable and the good days even better.

Throughout the duration of my PhD studies, I have owed my sanity to my close friends: Ahmad, Jamie, Mike, Jack, Jeff, Pete, Alex, Lukas, Charlie, Danny, and Cameron. You have offered me escape from the tribulations of academia, and were never shy to take me down a peg or two when required. Our trips and shenanigans have been amongst the high points during my time as a graduate student. Then, we may specify the optimisation problem collectively faced

Most importantly, I would like to thank my parents, and my brother Chris. Without them, this thesis would not be possible. Chris is, and will forever remain, my greatest friend. I would not be the person I am today without the never-ending support and love of my parents, for which I am eternally grateful. And yes, the thesis is finally done, you can stop asking now!

Nomenclature

Chapter 3

α	Dummy variable for squared Euclidean norm of the weight vector
β	Least squares solution to Kernel regression
$\ell_{+1}(\cdot,\cdot)$	Loss function of each agent
$\ell_{-1}(\cdot,\cdot)$	Loss function of the learner
ϵ	Error of `SDP-BISECT`
γ	Cost parameter
λ	SDP variable
$\mathbf{w}$	Weight vector
$\mathbf{w}^\star$	Optimal weight vector
$\mathbf{x}$	Clean input example sampled from $\mathcal{D}$
$\mathbf{x}^\star$	Optimal modification
$\mathbf{y}$	Vector of learner labels
$\mathbf{z}$	Vector of agent labels
$\mathcal{D}$	Data Distribution
$\mathcal{F}$	Reproducing kernel Hilbert space
$\mathcal{H}$	Class of induced hypotheses
$\mathcal{R}_m(\cdot)$	Rademacher complexity
$\phi(\cdot)$	Feature map
τ	SDP variable
$\tilde{\mathbf{x}}$	Modified input example
$\tilde{X}$	Modified data matrix
$c(\cdot,\cdot)$	Cost function
$F(\cdot)$	Dinkelbach function
$h_{\mathbf{w}}(\cdot)$	Hypothesis induced by $\mathbf{w}$
$k(\cdot,\cdot)$	kernel function
q	Dinkelbach parameter
$q^\star$	Optimal Dinkelbach Parameter
S	Training dataset
X	Data matrix
$X^\star$	Optimal data matrix
y	Learner's label
z	Agent's label

Chapter 4

$B_j^{\max}$	Maximum budget variation in the batch $\mathcal{T}_j$
α	Approximation ratio in regret definition
Δ_T	Size of batch during `RGA`
γ	Tuning parameter for Exp3
$\mathcal{J}_B$	Set of power 2 path variation budgets for time horizon T
$\mathcal{P}$	Class of admissible policies
$\mathcal{T}_j'$	$\mathcal{T}_j$ with the first $K + \tilde{D}$ and last $\tilde{D}$ time steps removed
$\mathcal{T}_j$	Batch j in `RGA`
π	Arm pulling policy
$\pi^\star$	Optimal policy
π_t	Arm pulled by policy pi on time step t
$\tilde{B}$	Maximum path variation within a single meta-block
$\tilde{D}$	Maximal blocking duration
$\underset{\sim}{D}$	Minimal blocking duration
$B_T^{\max}$	Maximum variation budget for the time horizon T
B_j	Path variation within batch $\mathcal{T}_j$
B_T	Path variation budget over time horizon T
D	Vector sequence of blocking durations associated with every arm k
$D_{\max}^k$	Largest blocking duration associated with arm k
$D_{\min}^k$	Smallest blocking duration associated with arm k
D_k	Sequence of blocking durations associated with arm k
D_k^t	Blocking duration associated with arm k at time step t
H	Meta-block length
K	Number of arms
L_j	Number of changes in batch $\mathcal{T}_j$
L_T	Number of changes budget for the time horizon T
R_π^α	α-regret
R_T^π	Pseudo-regret
T	Time horizon
X	Vector sequence of rewards associated with every arm
X_t^π	Reward received under policy π at time step t
$X_t^\star$	Reward received under optimal policy $\pi^\star$ at time step t
X^k	Reward sequence associated with arm k
X_t^k	Reward when arm k is pulled on time step t

Chapter 5

0	The null assignment
α	Approximation ratio in regret definition
Δ^{s-1}	s-dimensional probability simplex
$\Delta_{\min}$	Smallest gap in mean rewards between two services for the same agent

$\emptyset$	The null matching
$\mathcal{M}_T^D$	The set of admissible matching sequences
$\mathcal{M}_T$	The set of matching sequences of length T
$\mathcal{P}$	Set of all linear orderings over S
μ	Matrix of cardinal rewards
$\mu_{i,j}$	Cardinal reward received by agent i when assigned service j
μ_i	Vector of cardinal rewards associated with agent i
π	Matching policy
ψ	Tuple containing the truthful report policies for each agent i
σ	Permutation of services
$\succ$	Preference profile containing the ordinal preferences of each agent i
$\succ_i^t$	Internal ordinal preference estimation of agent i at time step t
$\succ_{-i}$	The ordinal preferences of all agents apart from agent i
$\succ_i$	Linear ordering induced by agent i's cardinal preferences
$\succ_i^\star$	Optimal report for agent i
$\mathrm{SW}(\cdot,\cdot)$	Social welfare
$\mathrm{W}_i(\cdot,\cdot)$	Welfare of agent i
I_π^α	α-incentive compatible regret
$\tilde{\psi}$	Tuple containing the report policies for each agent i
$\tilde{\psi}_i$	Report policy for agent i
$\tilde{\succ}$	The report of all agents
$\tilde{\succ}_i$	Report of agent i
$\tilde{D}$	Maximum blocking duration
$\zeta(\cdot)$	Incentive ratio of a policy π
D	Matrix of blocking durations
$D_{i,j}$	The blocking duration assigned to service j when matched to agent i
$H_t^{\succ}$	Report history up to time step t
H_t^m	Matching history up to time step t
H_t^r	Reward history up to time step t
M	A matching sequence
m	A matching
$m(i)$	The service allocated to agent i in the matching m
$M(t,i)$	The service assigned to agent i in the tth matching of the matching
$M^\star(\mu,D)$	The optimal matching sequence
m_t	The tth matching in the matching sequence M
N	Set of n agents
R_π^α	Dynamic α-regret for a policy π
$r_{i,t}$	Noisy reward received by agent i on time step t
S	Set of s services
T	Time horizon

Chapter 1

Introduction

As institutions and companies begin to operate at larger scales, relying on human decision makers becomes increasingly inefficient, and in some cases infeasible. For example, consider the problem of email spam classification, in which an email service provider must identify whether an email is spam or not (Dalvi et al., 2004). Of course, given the frequency and volume of emails sent and received, service providers cannot rely on human processing. Instead, they must rely on automated decision processes that scale effectively. This issue of scalability is not unique to the problem of spam classification, and arises in many problem domains including online advertising, kidney exchange programs and insurance quotation. Even in cases where employing human decision makers is possible, the increasing availability of data has made automated decision making processes a more cost-effective and perhaps more efficient alternative. As a result, organisations have increasingly turned to approaches grounded in machine learning and mechanism design to meet their needs.

Briefly put, machine learning concerns the study of algorithms that learn and improve by leveraging data to learn correct behaviour. Hence, machine learning methods are natural replacements for humans when making decisions at scale. In contrast, mechanism design, also known as inverse game theory, concerns the design of decision policies, or mechanisms, in order to achieve desired outcomes under the assumption that all participating parties act rationally. In many real world problems, data is supplied directly by agents, individuals or organisations that are invested in the outcomes chosen by a decision maker. Therefore, decision makers should expect data providers to act rationally, and change the data they submit in order to obtain a better outcome. In this sense, mechanism design is highly relevant to the design of platforms that facilitate data acquisition and information exchange. That is, whilst machine learning can be leveraged by decision makers to learn complex decision rules similar to those a human would propose, mechanism design is necessary to ensure that any decision is implemented correctly, taking into account the incentives of the individuals and organisations affected.

As one would expect, generic frameworks for machine learning and mechanism design make broad assumptions, which capture a wide range of practical problems, whilst still allowing for strong theoretical guarantees with respect to worst case performance. Such approaches include the celebrated Probably Approximately-Correct (PAC) learning framework (Valiant, 1984), as well as the seminal Vickrey-Clarke-Groves family of mechanisms (Vickrey, 1961; Clarke, 1971; Groves, 1973). However, many real world problem domains do not coalesce with standard assumptions in machine learning and mechanism design. The goal of this thesis is to devise nontrivial extensions of standard algorithms and approaches from machine learning and mechanism design to address such settings. In particular, we will examine three problem domains that lie at the intersection of machine learning and mechanism design:

1. Linear Regression with Strategic Agents
2. Sequential Deployment of Reusable Resources
3. Repeated Matching of Reusable Services

To motivate these problem domains, and to see how they are connected, consider the standard supervised machine learning setting often adopted by theorists, illustrated in Figure 1.1. Within this setting, the world is modeled as a data distribution from which a learner can sample independently and identically. Using the samples they observe, the learner may train some complex model, such as a neural network. Eventually the learner will deploy their model to make decisions. For example, perhaps the learner will use a model that they have trained on labeled to data to classify new and unlabeled data points. Whilst enabling for a systematic theoretical analysis, this framework fails to capture the dynamics of learning in many real world scenarios. In many cases, it is not the learner who retrieves data, but instead an intermediary agent, who may have their own goals or interests. Moreover, the decisions a learner makes often have a significant impact on the state of the world and therefore the intermediary agents that inhabit it. These concerns are illustrated in Figure 1.2. As a result, a learner must carefully consider how the decisions they make may impact the intermediary agents they interact with. Problem domain 1 captures this phenomena in the context of linear regression, and is detailed in Section 1.1. Moreover, practical settings often place behavioural constraints decision makers. For example, taking an action may rely on a resource which is scarce or expensive. In this case, the decision maker cannot take this action too frequently. Moreover, the constraints placed on a decision maker may vary through time based on how their previous decisions have impacted the real world. Figure 1.3 illustrates how this concern may be incorporated into the standard machine learning framework. Problem domain 2 considers this problem in the context of sequential resource allocation. For more details, we refer the

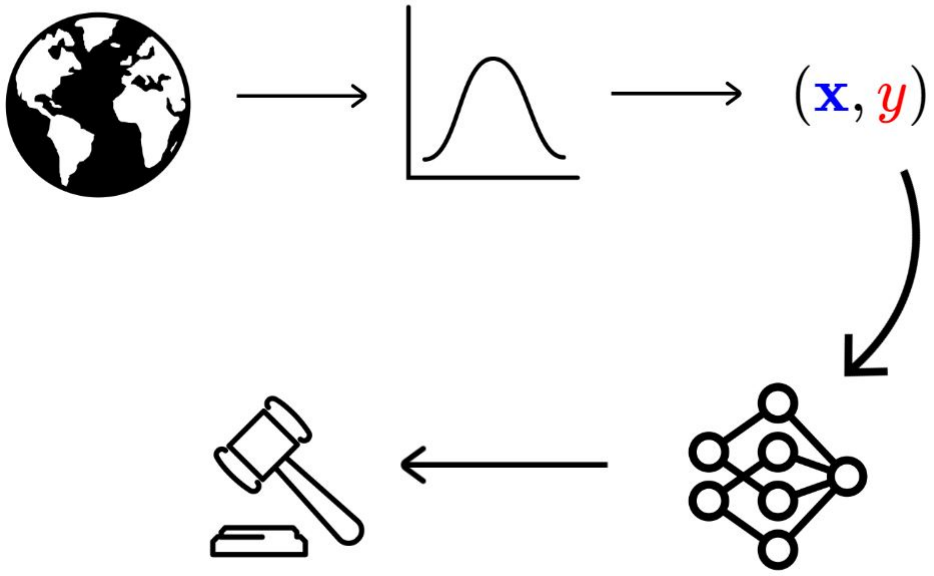

FIGURE 1.1: A depiction of the classical machine learning setting, in which the learner first models the environment (depicted above the the earth) using a data distribution, before gathering a sample which they use to train a model (such as a neural network), that is in turn used to make decisions (depicted by the gavel) in the real world.

reader to Section 1.2. Note that the concerns addressed in problem domains 1 and 2 are not mutually exclusive. That is, a learner may need to account for the behaviour of strategic agents whilst simultaneously obeying constraints imposed on their decision making policy by the real world. Such settings are illustrated by Figure 1.4 which can be viewed as an amalgamation of Figures 1.2 and 1.3. In this sense, problem domain 3 suffers from both the phenomena present in problem domains 1 and 2. More precisely, the learner is tasked with matching scarce resources to a group of agents who may behave strategically in how they choose their preferences to the decision maker. At the same, the learner may be constrained in how they can allocate resources through time. In the sections that follow, we treat each of the problem domains above in turn, describing their relevance to real world applications as well as discussing why standard approaches associated with each domain are insufficient. After this, we outline concrete research requirements for each problem domain, stating the properties any methodology or algorithm needs to satisfy to adequately address each problem domain. Finally, we discuss our own research contributions to each problem domain in detail, before presenting them in later chapters.

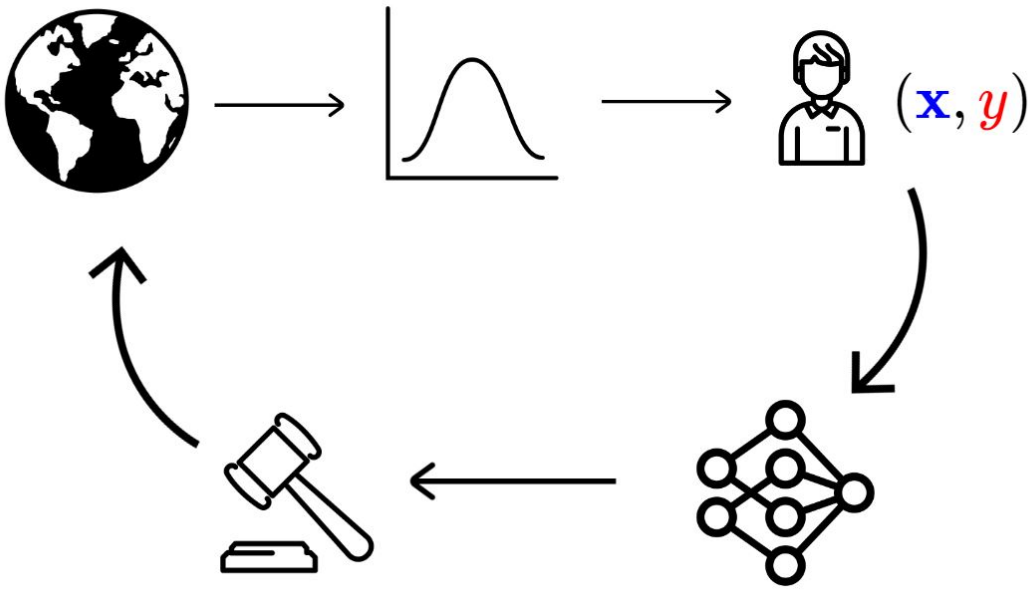

FIGURE 1.2: The role of data providers in the real world. In real world settings, unlike the classical supervised machine learning framework described in the previous figure, learners must often rely on independent agents to provide data samples. In such cases, it is important for the decision maker to account for the goals and incentives of each agent when training and deploying their model.

1.1 Linear Regression with Strategic Agents

As a starting point, consider the classical supervised machine learning setting. A learner is tasked with selecting a decision rule from a set, called a hypothesis class, to correctly label (or classify) input examples as positive or negative. To select a classifier, the learner is given access to a training dataset, composed of correctly labelled input examples. In the case of email spam classification, such a dataset may be composed of emails manually marked as spam (or not spam) by users. The learned classifier is then deployed to classify any future unlabelled data. Within standard learning frameworks, such as the PAC framework (Valiant, 1984), it is typically assumed that all data is drawn independently and identically from the same, but unknown, probability distribution. This motivates the approach of empirical risk minimisation (Mohri et al., 2018; Shalev-Shwartz and Ben-David, 2014), in which a classifier that correctly labels the most training data is selected from the hypothesis class. Since new data is sampled from the same distribution as the training data, one would hope that this classifier will perform well in the future. This intuition can be formalised by a uniform convergence argument which leads to probabilistic guarantees on the expected performance of the learned classifier on future data points. In what follows,

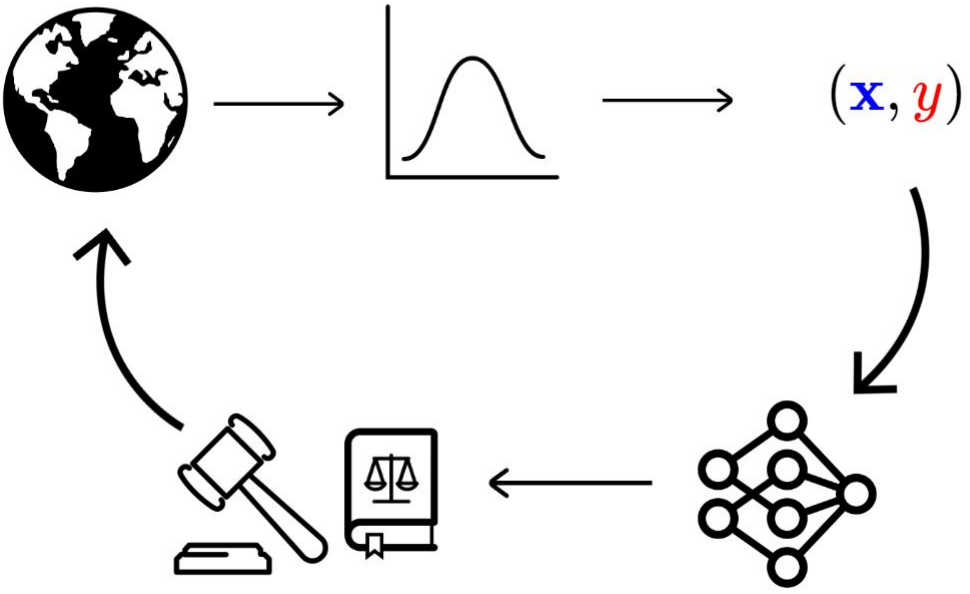

FIGURE 1.3: In real world settings, decision makers must account for the practical constraints imposed by the environment (as depicted by the rule book) . For example, taking a given action may correspond to consuming a resource with limited supply. As a result, decision makers must ensure that the outputs of their trained models adhere to, and perform well under, practical real world constraints.

we refer to the assumption described as a stationarity assumption, as it implies that the data distribution of interest to the learner remains fixed throughout time and does not change after the learner has deployed their chosen hypothesis.

Observe that, in many real world scenarios, assuming stationarity is unrealistic. Once again, consider the problem of spam classification (Dalvi et al., 2004). In this context, the classifier plays the role of a spam filter. Once deployed, users, including malicious email spammers, interact with the spam filter through sending emails. Over time, users will slowly learn what kind of emails are identified as spam by the filter and which are not. As a result, the email service provider should expect spammers to adjust the emails they send in the future in an attempt to bypass the spam filter. As recognised by Hardt et al. (2016), this phenomenon can be viewed as an instance of Goodhardt's law:

> When a measure becomes a target, it ceases to be a good measure.

That is, the learned spam filter implicitly relies on key email features to identify whether an email is spam. As spammers become aware of the filter that is employed, the spammers can game the system by manipulating these features and avoid

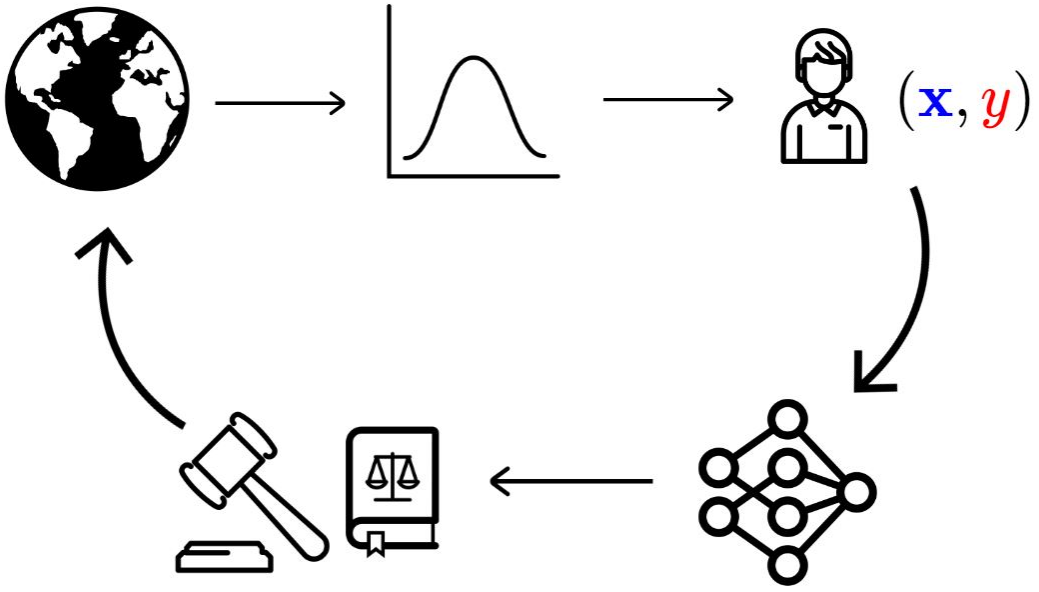

FIGURE 1.4: In many cases, the learner must account for both practical constraints imposed by the environment (depicted by the rule book) and the strategic behaviour of data providing agents. Put differently, many real world settings may be viewed as an amalgamation of the settings described by Figure 1.2 and 1.3.

detection. In other words, the data distribution is not stationary, and changes as a result of the classifier chosen by the learner after training. Therefore, it is difficult to make any theoretical guarantees regarding the performance of naive approaches, such as empirical risk minimisation, which do not account for how spammer behaviour may change once a spam filter is deployed.

Note that, in spam classification, each data point corresponds to an email, and therefore a user who is impacted by the filter's decision. When the user is honest, and not a spammer, the incentives of both the user and the filter are aligned; both the user and the filter want the email to be classified negatively. However, when the user is a spammer, the incentives of the user and the filter are misaligned; the user wants the email to be classified negatively, whilst the filter wants to classify the email positively. It is this misalignment of incentives which causes issue.

Observe that many other real world problems suffer from a similar misalignment of incentives. For example, consider the insurance quotation problem, in which an insurance broker must offer insurance quotes to customers depending on their individual characteristics, such as smoker status and age. In this setting, each data point is representative of an individual customer, who is clearly invested in the insurance quotation they are offered. As a result, the insurance broker should expect

customers to lie in attempt to receive a better insurance quotation. In addition, note that large scale insurance brokers cannot expect to investigate each individual customer for fraud, and therefore, building models resilient to strategic manipulation is of key importance.

Initial attempts at modelling settings such as spam classification more effectively were adversarial in nature. Unlike the classical supervised machine learning setting, adversarial machine learning settings (Vorobeychik and Kantarcioglu, 2018; Huang et al., 2011) assume that data is provided by an adversary, who attempts to maximise the error of the learner. In this sense, learning takes the form of a zero-sum game between the learner and the adversary. As a result, many algorithms for adversarial learning settings rely on techniques from minimax optimisation. Adversarial machine learning settings can vary significantly depending on the capabilities of the adversary (Biggio and Roli, 2018). For example, one may assume that all data is sampled from some fixed distribution, and that the adversary is permitted to perturb each sample by a fixed amount. In contrast, the adversary may be able to directly perturb the sampling distribution. Alternatively, the adversary may only be able to access data after training is completed. This last assumption is especially reasonable when the learner has access to some verification procedure. For example, an insurance broker could employ a fraud detection team to validate a small subset of customers so that training can be performed on a clean dataset.

Note that adversarial machine learning settings are intrinsically pessimistic[1]. In many problem scenarios, including those outlined above, data providers are not purely adversarial, and have their own goals and incentives which do not directly contradict the goals of the learner. In other words, adversarial settings are often too strict, and do not correctly capture the strategic interaction between data providers and the learner in many real world contexts. In some settings, strategic manipulation may benefit the learner. For example, if an honest user's email is incorrectly tagged as spam, they may perform edits so that the email passes through the spam the filter upon redelivery.

Such concerns have motivated the development of strategic classification (Hardt et al., 2016). In strategic classification, it is assumed that there is a classification which is particularly desirable, and that any data providing agent will manipulate their data in order to receive this classification. Note that strategic classification captures the setting of spam classification perfectly. In this case, each user wants their email to be classified negatively by the spam filter, and will make small formatting changes to their emails if they believe they would be classified as spam. If their email is already going to bypass the spam filter, there is no reason to make any changes.

[1]Note that our use of the term pessimistic is informal, and refers to the fact that agents are assumed to be adversarial in nature. Our use the term is not related its formal use within the context of bilevel optimisation.

Note that strategic classification makes the implicit assumption that one classification outcome is most desirable. However, many real world prediction settings are far more nuanced, and there is no single outcome preferred by all agents. For instance, consider the aforementioned insurance quotation problem. It is unclear what quotation a customer should be happy with. For example, a customer could have any of the following goals:

- To receive a better quotation than they have been offered in the past.
- To improve on the cost of their current insurance by a fixed amount.
- To reduce their quotation as much as possible.
- To receive a quotation that they believe is reasonable or fair.

Additionally, observe that insurance quotation is a regression problem, where data points are labelled with elements from the real line. As a result, it is unclear how strategic classification could be directly applied to this problem. In other words, the insurance quotation problem reflects demand for a more general framework for learning with strategic agents, in which a range of agent goals can be flexibly defined and dealt with. Such a framework needs to be able to capture a wide range of agent incentives, but also have meaningful guarantees with respect to computational tractability and sample efficiency.

To address this demand, we produce such a framework for linear regression, a foundational problem in theoretical machine learning. In particular, we consider a linear regression problem in which each data point is associated with a data providing agent. Furthermore, we assume that each data point comes with two labels. The first label represents the ground truth labelling of the input data point that the learner is interested in predicting. Meanwhile the second label reflects the labelling most preferred by the data providing agent. Under square loss, we show that the optimal linear model can be found in polynomial time. In addition, we show that a version of empirical risk minimisation, based on Stackelberg prediction games (Brückner and Scheffer, 2011), enjoys sample complexity guarantees similar to those of empirical risk minimisation in classical supervised machine learning settings. These results are described in detail in Chapter 3.

1.2 Sequential Deployment of Reusable Resources

In real world scenarios, decisions often correspond to the deployment of important reusable resources. For example, consider a disaster response scenario in which emergency resources must deployed in real time. Any decision corresponds to the

active deployment of a scarce resource, such as emergency service vehicles or expert personnel, who will be unavailable in the meantime. After a certain amount of time depending on the task given, such resources may become available for redeployment. That is, the future availability of resources, and therefore actions, depends on the past actions of the decision maker. Such problems are typically referred to as blocking problems (Basu et al., 2021a, 2019; Atsidakou et al., 2021; Papadigenopoulos and Caramanis, 2021). In blocking problems, a decision maker must interact with an environment by repeatedly taking actions over a fixed number of time steps. Whenever an action is taken, the decision maker receives a reward, depending on the state of the environment, and the action is blocked (or unavailable) for a fixed number of time steps. The goal of the decision maker is to maximise their cumulative reward over the time horizon.

Unfortunately, trivial methods of integrating blocking into traditional models for sequential decision making typically lead to intractability. For example, consider Markov decision processes, the environmental model adopted by most reinforcement learning frameworks (Sutton and Barto, 2018; Szepesvári, 2010). At each time step in a Markov decision process, the environment enters a new state, depending on the previous state of the environment and the previous action taken by the decision maker. One way to extend Markov decision processes to blocking problems is to consider the current subset of blocked actions as part of the environmental state. However, the number of action sets that can be blocked is exponential in the total number of actions available to the decision maker. Thus, extending Markov decision processes to incorporate blocking in this manner can lead to an exponential increase in the size of the state space. As sample complexity bounds for reinforcement learning typically rely on the cardinality of the action space and the state space, this is undesirable.

An even simpler model for sequential decision making is the multi-armed bandit (MAB) problem (Bubeck and Cesa-Bianchi, 2012; Lattimore and Szepesvári, 2020). The term bandit refers to slot machines which have a single arm that can be pulled. In the MAB problem, the decision maker has access to K arms (or actions), and can pull one arm on each time step. After pulling an arm, the decision maker receives a reward. In the standard, stochastic MAB problem it is assumed that the reward for pulling an arm is sampled identically and independently from an associated distribution which remains fixed throughout time.

A strategy for pulling arms through time is called a policy. Ideally, the decision maker would like to adopt a policy which maximises their expected cumulative reward over the course of the time horizon. However, this is impossible, as the decision maker has no initial knowledge about the rewards associated with each arm. Instead, the decision maker can benchmark its own policy against a number of baseline policies. In the stochastic setting, the set of policies which pull the same arm on every time step are often used as a baseline. This is motivated by the fact that it is optimal for the

decision maker to pull the arm with highest expected reward on every time step, and as such this class of policies always contains the optimal policy. We refer to the regret of the decision maker as the difference between the cumulative reward received by the decision maker's policy and the cumulative reward received by the best baseline policy in hindsight. The goal of the decision maker is then to adopt a policy which incurs small regret in expectation.

Several well known algorithms for the stochastic bandit setting provide finite time regret guarantees, ensuring the expected regret experienced by the decision maker is bounded by a term sublinear in the length of the time horizon. One example is the upper confidence bound (UCB) algorithm originally proposed by Auer et al. (2002). Adapting such algorithms for settings with blocking presents several challenges. Firstly, it is unclear which baseline policies to compare against. It may be infeasible for the decision maker to pull the arm with the highest expected reward on every time step, and as a result, comparing against policies which pull the same arm repeatedly is unrealistic. Instead, one can compare to an offline oracle, which has full information regarding the reward distributions of the arms and returns an approximately optimal policy. Note that constructing such an oracle is nontrivial and involves designing an approximation algorithm for a combinatorial problem with a close relationship to many (hard) scheduling problems. Secondly, relating the performance of a bandit algorithm to the performance of the offline oracle is nontrivial, and typically requires some similar arm selection criteria to be used by both algorithms.

For example, Basu et al. (2019) extends the UCB algorithm to the blocking setting by benchmarking against an offline greedy algorithm which simply pulls the arm with the highest expected reward amongst those available. The extension of UCB proposed by the authors is essentially greedy, optimistically pulling the arm with the highest upper confidence bound amongst those available. This allows the authors to relate the cumulative reward of UCB to the greedy oracle, as eventually, the UCB algorithm will approximate the mean reward associated with each arm to a high degree of accuracy and select the same arms as the greedy oracle. Similar approaches have been formulated for a wide variety of bandit settings by numerous authors (Basu et al., 2021a; Atsidakou et al., 2021; Papadigenopoulos and Caramanis, 2021).

Aside from the blocking of resources, observe that the relative effectiveness of a given resource is likely to vary and change throughout time. Once again, consider a disaster response setting. Resources may correspond to emergency vehicles and specialised personnel whose relevance and use varies over time as the disaster progresses. In some cases, the disaster may have even been caused by a malicious group who respond dynamically to the decision maker in order to limit the effectiveness of deployed resources. Therefore, stochastic MABs, which assume the reward distribution associated with each resource is fixed throughout time, fail to reflect the evolving nature of many real world settings. Considering this, many MAB models

have been proposed in the literature seeking to relax the stationarity assumption implcit in stochastic MABs. These include nonstationary stochastic bandit settings (Besbes et al., 2014; Wei et al., 2016; Auer et al., 2019), which allow the reward distribution associated with each arm to change on each time step in accordance with a budget constraint, and adversarial bandits (Auer et al., 2002), which assume the reward associated with each arm is chosen by a malicious adversary on each time step. We refer to such MAB models collectively as nonstationary bandit models.

Summarising, stochastic MABs suffer from two deficiencies that make them inappropriate for modelling sequential resource deployment in many real contexts. The first, failure to model resource inavaibility, has been addressed by blocking bandit models. Whilst the second, failure to capture the varying effectiveness of resources through time, is addressed by nonstationary bandit models. Of course, any realistic model for sequential resource deployment should address both these of deficiencies *simultaneously*. Motivated by this, we propose the adversarial blocking bandits setting, which features both blocking and nonstationary rewards. Following the approach of blocking bandits, we first develop an efficient offline approximation algorithm for this setting to serve as a regret benchmark. Using ideas from nonstationary stochastic bandits, we then develop a new bandit algorithm with good finite time regret guarantees. The adversarial blocking bandits setting, and our proposed algorithms, are the subject of Chapter 4.

1.3 Sequential Matching of Reusable Services

In many real world settings, a decision maker is tasked with assigning a variety of resources amongst a number of individuals, or agents. For example, a cloud computing company is faced with task of assigning their hardware resources (e.g. GPUs) to jobs submitted by clients. Similarly, consider the problem faced by a recruiter who serves as intermediary between companies and freelance contractors. In this setting, the recruiter is tasked with assigning contractors to open positions at each company. Note that in both cases, the agents hold preferences over the resources being assigned. In cloud computing, customers want to be assigned hardware that completes their submitted task in a fast and cost-efficient manner. Meanwhile, in the recruitment setting, companies may prefer certain contractors over others, based on their suitability for the position advertised.

Such settings are examples of one-sided matching problems. In a one-sided matching problems, a central planner is tasked with assigning a set of resources, or services, to a set of agents. Each agent can only be assigned one service and each service can be assigned to at most one agent. In other words, agents and services are matched together. Additionally, we assume that agents hold private preferences over the set of

services. Preferences can be represented ordinally or cardinally. An agent with ordinal preferences holds a linear preference ordering defined over services. Naturally, each agent prefers being matched to services higher in their preference ordering compared to those lower in their preference ordering. In contrast, an agent with cardinal preferences holds positive real values, called utilities, describing how happy they are to be matched to each service. The higher an agent's utility for a given service, the happier the agent is with being assigned said service.

To aid the central planner, each agent reports their preferences over services. If the underlying preferences of the agents are ordinal, then the preferences reported by the agent are also ordinal. If the underlying preferences of each agent are cardinal, then the preferences of reported by each agent may either be ordinal or cardinal, depending on problem context. Note that an agent's report may not reflect their underlying preferences. That is, an agent may lie if they believe it is in their best interest.

The goal of the central planner depends significantly on problem context. However, generally speaking, the central planner aims to identify matchings which are both fair and efficient. Many different measures of fairness and efficiency have been proposed within the matching literature. The suitability of a given fairness or efficiency measure depends heavily on the underlying structure of agent preferences and reports. For example, when the underlying preferences of each agent are ordinal, it is natural to formulate notions of fairness and efficiency via first-order stochastic dominance (Bogomolnaia and Moulin, 2001; Hosseini et al., 2015, 2018). In the case of cardinal preferences, the central planner typically adopts fairness and efficiency measures explicitly defined in terms of the utility profiles of each agent, such as social welfare (Filos-Ratsikas et al., 2014), and Nash social welfare (Caragiannis et al., 2019; Abebe et al., 2020).

For large scale platforms that involve human decision makers, one-sided matching settings with cardinal preferences and ordinal reporting are of particular interest. In many cases, it is difficult for a human to ascribe an exact number to a service which describes its value. However, humans often find it easy to select the service they prefer most from a list of options. For instance, consider again the recruitment example introduced above. It may be difficult for a hiring manager to characterise the exact financial value a specific contractor would generate for their organisation. On other hand, the hiring manager may know they prefer one contractor over another based on the previous experience or expertise of both candidates. Put differently, cardinal preference structures with ordinal reporting allow for detailed modelling of an agent's preferences, whilst also capturing the potential difficulty an agent may have in communicating or expressing their own preferences to the central planner.

Unfortunately, under ordinal reporting, it is impossible for a central planner to optimise cardinal performance measures such as a social welfare and Nash social

welfare. This is because a single ordinal preference profile can correspond to many different cardinal utility profiles. Hence, the central planner must consider alternative performance measures, such as distortion (Procaccia and Rosenschein, 2006; Filos-Ratsikas et al., 2014), which account for the unavoidable loss in performance associated with ordinal reporting.

In addition to efficiency and fairness, it is important that any matching policy adopted by the central planner is truthful, or equivalently strategyproof. We say that an algorithm is truthful if it is always optimal for an agent make a report that reflects its true underlying preferences. From an ethical standpoint, truthfulness ensures that agents who are honest are rewarded. It is also unclear how to even design a practically useful algorithm which is not truthful, as such an algorithm would need to find optimal matchings with potentially false information. Once again, how truthfulness is formulated typically depends on the underlying preference structure assumed. Unfortunately, in cardinal settings with ordinal reporting, fairness, efficiency and startegyproofness may be impossible to simultaneously guarantee (Zhou, 1990). As a result, the central planner must be pragmatic and carefully trade-off desirable properties depending on problem context.

Several effective matching algorithms have been developed by the mechanism design community. In particular, the random serial dictatorship (RSD) algorithm (Abdulkadiroglu and Sonmez, 1998) is truthful and optimal in terms of distortion for agents with cardinal preferences in settings with ordinal reporting (Filos-Ratsikas et al., 2014). RSD is a simple algorithm, letting agents choose their most preferred agent from those that are unmatched in a randomly sampled order.

Note that one sided-matching is a single shot problem. That is, all agents are matched to services simultaneously and are never unmatched. In many practical settings, services are reusable, and agents are matched to services repeatedly over many time steps. Consider again the problem faced by cloud computing companies. Once a client is finished with a computational task, the hardware they are using is free to be reassigned. Moreover, clients are likely to return with new and similar jobs in the near future. Likewise, consider the problem faced by recruiters. Once a freelancer has finished their contracted period of employment, they become available for hire again. In addition, companies are likely to have similar openings in the future and thus may return to the recruiter repeatedly. Note how repeated one-sided matching problems are often blocking problems. The availability of a contractor, for example, depends on the company they were previously employed with and the duration of the associated contract.

Moreover, it is unrealistic to expect agents to be initially aware of their underlying preferences in such settings. For instance, a company will only know if they like a freelancer after they have worked with them several times. Similarly, a cloud

computing client may only have a good estimate of how long their jobs take to complete on a given machine after several trials. In other words, real world one-sided matching problems often have learning components. Matching markets with learning have been studied a number of times in the literature (Liu et al., 2020; Cen and Shah, 2022; Basu et al., 2021b), but avoid issues of reusability and service unavailability.

To address this gap in the literature, we propose a new model for repeated one-sided matching, called sequential blocked matching (SBM). In the SBM model, a central planner must produce a one-sided matching between agents and services on each time step. Once a service is matched, it becomes blocked and unavailable for a duration depending on the agent matched to. For this setting, we propose an extension of RSD which is both optimal in terms of distortion and satisfies a relaxed notion of truthfulness, called incentive ratio (Chen et al., 2012; Wang et al., 2020). Moreover, we propose a bandit version of our algorithm for settings in which agents are initially unaware of their own preferences. The SBM model is described in detail in Chapter 5.

1.4 Research Requirements

As previously mentioned, the aim of this thesis is to extend generic techniques and algorithms at the intersection of machine learning and mechanism design to address the problem domains outlined above. In what follows, we outline the research requirements for each problem domain in turn.

1. **Linear Regression with Strategic Agents**

   (a) **Flexible Incentive Modelling (Requirement 1a)** - As discussed in Section 1.1, existing machine learning frameworks either pay no attention to the incentives of data providing agents, or target agents with specific aims and goals. Any general framework for learning with strategic data providers must be able to represent a wide range of agent incentives. Similarly, such a framework should offer flexibility when it comes to the capabilities of data providing agents. Depending on problem context, agents may experience different costs for manipulating their data. For instance, insurance customers may face a far higher potential cost for data manipulation than email spammers, as data manipulation may correspond to the physical manipulation of evidential documents, such as income statements.

   (b) **Computational Tractability (Requirement 1b)** - With the increasing availability of data, it is important that learning algorithms scale effectively in terms computational complexity. Learning algorithms which scale poorly with either dataset size or data dimension, are typically impractical in large scale settings.

(c) **Polynomial Sample Complexity (Requirement 1c)** - Similarly, any proposed algorithm should be sample efficient, only requiring a small number of training examples to learn effective decision rules. This is particularly important when the verification of data is difficult. For example, insurance companies may need to employ fraud detection teams or expensive auditing procedures to verify customer data. Hence, an insurance broker must trade-off the size of any training dataset with the cost of obtaining it.

2. **Sequential Deployment of Reusable Resources**

(a) **Accounting For Nonstationary Reward Sequences (Requirement 2a)** - In many resource deployment scenarios, the environment changes rapidly, and in some cases, adversarially in response to the previous actions of the decision maker. Hence, any model for sequential resource deployment should allow for the rewards that are nonstationary. Moreover, note that policies which repeatedly take the same action may be suboptimal in nonstationary environments. As a result, the decision maker must adopt new baseline policies which explicitly account for the evolving nature of rewards through time.

(b) **Accounting For Resource Blocking (Requirement 2b)** - In real world settings, resources often become unavailable for a duration once deployed. Thus, any realistic model for sequential resource deployment must account for the blocking of resources. In settings where resources can become blocked for a duration, classic benchmark policies, such as repeatedly taking the best fixed action in hindsight, may be infeasible, and thus, unrealistic performance measures. Hence, algorithms for blocking settings must be compared against new baseline policies which are feasible, and have strong guarantees with respect to optimality. As computing an optimal policy in blocking settings often involves solving an intractable combinatorial optimisation problem, constructing such benchmark policies is nontrivial and presents a significant technical challenge.

(c) **Finite-time Regret Guarantees (Requirement 2c)** - Of course, any algorithm proposed should have strong guarantees with respect to a benchmark policy which satisfies Requirements 2a and 2b. More specifically, any proposed algorithm should satisfy a finite-time regret bound with respect to a suitable baseline policy that is sublinear in the length of the time horizon. This ensures that the performance of the algorithm proposed approaches that of the offline benchmark policy as the time horizon lengthens. Note that the design of such algorithms is nontrivial, and suffers from the issues encountered in both blocking and nonstationary settings simultaneously.

3. **Repeated Matching of Reusable Services**

   (a) **Performance Guarantees With Blocking (Requirement 3a)** - As in the single shot case, any proposed algorithm for repeated one-sided matching should satisfy a suitable performance benchmark. For example, when agents hold cardinal preferences but report ordinally, any proposed algorithm should have good guarantees with respect to distortion. Unfortunately, the properties of traditional single shot matching algorithms, such as RSD, do not carry over to repeated matching settings in which resources can be blocked. As a result, existing matching algorithms must be modified in order to account for service blocking in repeated settings.

   (b) **Truthfulness Under Blocking (Requirement 3b)** - As discussed in Section 1.3, truthfulness is a desirable property for any algorithm, as it is ensures that agents are rewarded for being honest. As opposed to the single shot setting, truthfulness is harder to guarantee in repeated matching settings with blocking. This is because actions that look best in the short term, may not be best in the long-term. Thus, when making assignments, a central planner must look far into the future to ensure that the long-term utility of an agent is maximised when they report truthfully. Thus, single shot matching algorithms, which are greedy and do not look forward in time, must be modified to ensure truthfulness in repeated settings.

   (c) **Accounting For Agents That Learn (Requirement 3c)** - In many real world settings, agents are initially unaware of their preferences. Such settings demand algorithmic frameworks that allow agents to learn their most preferred services over the time horizon. In other words, the central planner faces an exploration-exploitation trade-off between assigning less familiar services to agents, so that they may learn more about their preferences, and assigning services preferred by agents given their current knowledge.

1.5 Research Contributions

With the requirements of the previous section in mind, we present three theoretical frameworks, each aimed at addressing a specific problem domain. In what follows, we describe the salient properties of each framework, and their relationship to the research requirements described in the previous section.

1. **Stackelberg Prediction Games for Linear Regression (Chapter 3)** - Firstly, to address Problem Domain 1, we present a strategic learning framework for linear regression based on Stackelberg prediction games. Within this model, we assume that input examples are supplied by an agent, who samples from a

stationary distribution of interest to the learner. Each data point sampled from this distribution comes with two labels. Like classical supervised settings, the first label represents the prediction target for the learner. Meanwhile, the second label represents the outcome preferred by the agent. We refer to the first label as the learner's label, and the second as the agent's label. Note that by varying the agent's labels, a wide range of agent incentives can be modeled. As a result, the framework proposed partially addresses research Requirement 1a.

Aside from data provided by the agent, we assume that the learner has access to training samples taken directly from the distribution of interest to use for training. In practice, such a dataset may be procured through a costly verification process available to the learner. After training, we assume that the agent has full knowledge of the linear model chosen by the learner and manipulates the input examples they provide in order to minimise loss with respect to their own labels.

We then define an analog of empirical risk minimisation for this setting, which takes the form of a bilevel optimisation problem. In the case of square losses, we derive a polynomial time algorithm, based on semidefinite programming and bisection search, which minimises empirical risk. In other words, we address Requirement 1b for the square loss function. Lastly, we show that the sample complexity of empirical risk minimisation in the proposed framework inherits the sample complexity guarantees of empirical risk minimisation in the classical supervised setting, under minor assumptions. In doing so, we address Requirement 1c. This line of research led to the following published work:

> Nicholas Bishop, Long Tran-Thanh, and Enrico Gerding. Optimal learning from verified training data. In *Advances in Neural Information Processing Systems*, volume 33, pages 9520–9529. Curran Associates, Inc., 2020b

2. **Adversarial Blocking Bandits (Chapter 4)** - In order to address Problem Domain 2, we introduce the adversarial blocking bandits setting. Our model extends directly from the stochastic blocking bandits model proposed by Basu et al. (2019). That is, when an arm is pulled, it is blocked and cannot be pulled again for a short duration. Unlike Basu et al. (2019), we assume that the period of unavailability (or blocking duration) associated with each arm when it is pulled can change *arbitrarily* from time step to time step. In this sense, adversarial blocking bandits constitute a more general model of resource unavailability compared to other MAB settings that exist in the literature.

   Moreover, we assume that the reward associated with each arm is chosen by an adversary, who must obey a budget constraint. Motivated by the nonstationary bandit literature, we consider reward sequences constrained via a path variation budget (Besbes et al., 2014). As a result, adversarial blocking bandits consider

both blocking and nonstationarity simultaneously, addressing Requirements 2a and 2b.

To define a meaningful policy benchmark, we first study an offline version of the adversarial blocking bandits setting, in which the decision maker is fully aware of the rewards associated with each arm on every time step. We first prove that finding the optimal policy in this setting is NP-hard. Considering this, we study a greedy algorithm, called `Greedy-BAA`, which pulls the arm with the highest reward out of those available. We show that `Greedy-BAA` provides a good approximation guarantee against the optimal policy in hindsight, and adopt it as a policy benchmark for the bandit setting.

Using `Greedy-BAA` as a basis, we develop a number of algorithms for the bandit setting which aim to address Requirement 2c. First, we introduce the repeating greedy algorithm (`RGA`). When the path variation budget constraint imposed on rewards is known, we show that `RGA` achieves sublinear regret, where the regret is computed against the performance of `Greedy-BAA`. For cases in which the path variation budget is unknown, we devise `META-RGA`, a meta-bandit algorithm which achieves sublinear regret with respect to `Greedy-BAA`. This line of work resulted in the following publication:

> Nicholas Bishop, Hau Chan, Debmalya Mandal, and Long Tran-Thanh. Adversarial blocking bandits. In *Advances in Neural Information Processing Systems*, volume 33, pages 8139–8149. Curran Associates, Inc., 2020a

3. **Sequential Blocked Matching (Chapter 5)** - Lastly, to address Problem Domain 3, we propose a new repeated matching setting, which we refer to as the sequential blocked matching (SBM) setting. In an SBM model, a central planner is tasked with repeatedly selecting a matching between a set of agents and a set of services. Every time an agent is matched to a service, the service is unavailable (or blocked) for a fixed number of time steps. The duration for which a service is blocked may depend on the agent the service was matched to. We assume that agents have cardinal preferences, but report ordinally. First, we study an online version of this setting, in which each agent is fully aware of their own preferences in advance, and assume that agents report their preferences once at the beginning of the time horizon.

   To tackle this setting, we propose a repeated version of RSD, called repeated RSD (`RRSD`), which is optimal in terms of distortion (Procaccia and Rosenschein, 2006) when all agents report truthfully, addressing research Requirement 3a. Moreover, we show that `RRSD` satisfies an approximate notion of truthfulness, known as incentive ratio (Wang et al., 2020), that ensures can agents only gain a fixed amount by misreporting. In this sense, we make a significant contribution to research Requirement 3b.

Lastly, we consider a bandit version of the SBM setting, in which agents are initially unaware of their own preferences, with the goal of addressing research Requirement 3c. When an agent is matched, we assume the agent receives a noisy reward corresponding to its utility value for the service it was matched to. To aid the central planner, we assume that agents can report their preferences at every time step. We say that an agent is truthful if it reports preferences which are induced by the empirical means of the rewards observed so far. We then develop an bandit version of `RRSD` (`BRRSD`), based on the explore-then-commit paradigm (Lattimore and Szepesvári, 2020), which allows agents to learn their preferences during the matching process. Under the assumption that agents are truthful, we derive finite-time regret guarantee for `BRRSD`, using `RRSD` as a performance baseline. Moreover, we evaluate this bandit setting from the perspective of each agent, and show that under `BRRSD`, it is approximately optimal for agents to report truthfully. That is, truthful reporting provides finite-time regret guarantees against the best sequence of misreports in hindsight. This line of work has led to the following publication:

> Nicholas Bishop, Hau Chan, Debmalya Mandal, and Long Tran-Thanh. Sequential blocked matching. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(5):4834–4842, 2022

1.6 Thesis Outline

The remainder of this thesis is structured as follows:

- In Chapter 2, we review the literature relevant to each problem domain in turn. For Problem Domain 1, we begin by studying the classical supervised machine learning setting, and discuss its failings with respect to modelling strategic behaviour. Then, we review a number of strategic learning settings which attempt to address these shortcomings. In particular, we briefly review the vast literature on adversarial machine learning, strategic classification, performative prediction and Stackelberg prediction games. Next, we discuss literature relevant to Problem Domain 2. To start, we introduce the classical stochastic MAB setting, before discussing nonstationary MABs and MABs which impose constraints on the actions available to the learner. Lastly, we discuss literature relevant to Problem Domain 3, first reviewing the traditional single shot one-sided matching problem before discussing the pros and cons of existing models for repeated matching.
- In Chapter 3, we introduce a new framework for linear regression in the presence of strategic agents. In particular, we present an analog of empirical risk

minimisation for this setting, and derive statistical guarantees analogous to those for empirical risk minimisation in the classical supervised machine learning setting. Additionally, we present an algorithm for empirical risk minimisation in our setting based on semidefinite programming and bisection search, which provably converges to global optima.

- In Chapter 4, we introduce a multi-armed bandit setting designed to model the deployment of reusable resources, which directly extends the blocking bandit setting of Basu et al. (2019) to adversaries with bounded path variation budget. Following this, we investigate a full information version of this setting, and introduce a greedy algorithm, called greedy best available arm (`Greedy-BAA`), with strong approximation guarantees with respect to the best arm pulling policy. We then introduce a new bandit algorithm based on `Greedy-BAA`, called the repeating greedy algorithm (`RGA`), which achieves sublinear finite-time regret guarantees when the path variation budget is known. We then provide an extension of `RGA`, called `META-RGA`, which achieves sublinear finite-time regret guarantees even when the path variation budget is unknown to the decision maker.

- In Chapter 5, we introduce a new repeated one-sided matching setting that incorporates the blocking of reusable resources, called sequential blocked matching (SBM). To begin, we analyse a full information version of this setting in which agents are initially aware of their underlying preferences. We then propose an extension of random serial dictatorship, called repeated random serial dictatorship (`RRSD`), that achieves optimal distortion guarantees under the assumption that agents report truthfully. We then show that `RRSD` has bounded incentive ratio, which can be interpreted as a relaxed notion of truthfulness. After this, we present a bandit version of `RRSD` (`BRRSD`).

- Lastly, in Chapter 6, we conclude and expand upon directions for future work which would broaden the applicability of the research presented in previous chapters.

Chapter 2

Literature Review

In this chapter, we step through each problem domain in turn, and provide an overview of relevant research. Readers interested in a specific domain need only read the relevant sections of this review. In Section 2.1 we review literature relevant to Problem Domain 1, starting with a review of the classical supervised machine learning setting, before discussing relevant extensions such as the adversarial machine learning framework, strategic classification, performative prediction and Stackelberg prediction games. Then, in Section 2.2, we discuss Problem Domain 2, describing the classical MAB setting, and afterwards discussing various extensions designed to tackle resource unavailability and variation in rewards over time. Lastly, in Section 2.3, we investigate Problem Domain 3, discussing relevant algorithms for one-sided matching, before analysing existing frameworks for one-sided repeated matching.

2.1 Linear Regression with Strategic Agents

In this section, we examine research of relevance to Problem Domain 1. To begin, we introduce the classical supervised machine learning setting in Section 2.1.1. In particular, we review the method of empirical risk minimisation (ERM) and its associated guarantees regarding sample complexity. Unfortunately, as discussed in the previous chapter, conventional supervised settings fail to model the incentives and goals of data providing agents effectively. With this concern in mind, we survey literature from the adversarial machine learning framework in Section 2.1.2, where it is assumed that data is provided by a malicious adversary. Unfortunately, adversarial models are often overly pessimistic[1], not reflecting the underlying goals of data providers in the real world. Hence, in Section 2.1.3 we turn our focus to strategic

[1] As in the previous chapter, our use of the term pessimistic is informal, and refers to the fact that agents are assumed to be adversarial in nature. Our use the term is not related its formal use within the context of bilevel optimisation.

machine learning frameworks, which explicitly model the goals and incentives of data providers. In particular, we we examine strategic classification (Hardt et al., 2016). Unfortunately, we will find that most strategic classification models target agents with very specific goals or incentives. As a result, we pursue more general frameworks for learning with strategic agents in Section 2.1.4. More specifically, we review Stackelberg prediction games (Brückner and Scheffer, 2011), which serve as a basis for our work in Chapter 3, and performative prediction (Perdomo et al., 2020).

2.1.1 Classical Supervised Learning and Empirical Risk Minimisation

In this section, we review the classical supervised machine learning setting, with a specific focus on linear predictors. In the classical supervised machine learning setting, a learner is tasked with selecting a prediction rule, or hypothesis, $h : \mathcal{X} \to \mathcal{Y}$, from a set, or hypothesis class, $\mathcal{H}$, which maps elements of an input space, $\mathcal{X}$, to a label set, $\mathcal{Y}$. For the remainder of this section, we will assume that $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{Y} \subseteq \mathbb{R}$.

The learner is granted access to a training set, $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, containing a finite number of training examples, each consisting of an element of the input space paired with a corresponding target label. We assume that each example $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ is drawn independently and identically from some underlying probability distribution $\mathcal{D}$. We refer to $h(x)$ as the prediction made by hypothesis h given input x.

The quality of a prediction is evaluated via a loss function, $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$. More specifically, the loss incurred by a hypothesis h given an example (x, y) is equal to $\ell(h(x), y)$. Common choices of loss function include the zero-one loss, $\mathbb{1}[h(x) \neq y]$, and the squared loss, $(h(x) - y)^2$. The risk, $\mathcal{L}(h)$, associated with a hypothesis h, is the expected loss with respect to the distribution $\mathcal{D}$. That is,

$$\mathcal{L}(h) = \mathbb{E}_{\mathcal{D}}[\ell(h(x), y)].$$

The goal of the learner is to select a hypothesis with minimal risk:

$$h^\star \in \arg\min_{h \in \mathcal{H}} \mathcal{L}(h).$$

However, this goal is unrealistic, as the learner does not have direct access to the distribution $\mathcal{D}$, and thus cannot evaluate the risk of any hypothesis. Instead, the learner only has access to the sample S taken from the distribution $\mathcal{D}$. As a result, the learner must aim for a relaxed goal, which is practically achievable. This motivates the definition of probably-approximately-correct (PAC) learnability (Valiant, 1984).

Definition 2.1 (Agnostic PAC Learnability)**.** A hypothesis class $\mathcal{H}$ is agnostic PAC learnable with respect to a set $\mathcal{X} \times \mathcal{Y}$ and a loss function ℓ if there exists a polynomial

function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm $\mathcal{A}$ such that for every $\epsilon, \delta \in (0,1)$ and for every distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, running $\mathcal{A}$ on $m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d. examples generated by $\mathcal{D}$ returns a hypothesis $\hat{h}$ such that, with probability of least $1 - \delta$,

$$R(\hat{h}) \leq \min_{h \in \mathcal{H}} R(h) + \epsilon.$$

If $\mathcal{A}$ runs in poly$(1/\epsilon, 1/\delta)$-time, then $\mathcal{H}$ is said to be efficiently PAC-learnable. When such an algorithm $\mathcal{A}$ exists, it is called a PAC-learning algorithm for $\mathcal{H}$.

Note that PAC learnability relaxes the original goal of the learner in several ways. First of all, the learner is only expected to find a good hypothesis with high probability. Secondly, the learner is only expected to find an approximation of the best hypothesis in $\mathcal{H}$. However, note that the number of training samples required to get an ϵ-approximation with $1 - \delta$ probability must be polynomial in $1/\epsilon$ and $1/\delta$. The function $m_{\mathcal{H}}$ is typically referred to as the sample complexity of algorithm $\mathcal{A}$ for hypothesis class $\mathcal{H}$, and characterises the number of training examples required to get a meaningful approximation with high probability. In summary, the learner is in pursuit of a polynomial time algorithm $\mathcal{A}$ with polynomial sample complexity.

A natural approach is to select the hypothesis which performs best on the training dataset. This is known as empirical risk minimisation (ERM). More formally, given a training set S, we define the empirical risk $\hat{\mathcal{L}}(h)$ of a hypothesis as follows:

$$\hat{\mathcal{L}}(h) = \frac{1}{m} \sum_{i=1}^{m} \ell(h(x_i), y_i).$$

Performing ERM corresponds to choosing a hypothesis, h_{ERM}, which minimises the empirical risk:

$$h_{\text{ERM}} \in \arg\min_{h \in \mathcal{H}} \hat{\mathcal{L}}(h).$$

For many popular hypothesis classes and losses, approximate solutions for ERM can be found in polynomial time. For example, when $\mathcal{H}$ is a compact, convex set, and ℓ is a convex function, ERM corresponds to a convex optimisation problem which can often be solved via standard optimisation methods such as projected gradient descent and the interior point method (Dikin, 1967; Karmarkar, 1984; Boyd and Vandenberghe, 2004).

Sample complexity guarantees for ERM-based algorithms typically depend on a uniform convergence argument. That is, showing that $\hat{\mathcal{L}}(h)$ is a good approximation of $\mathcal{L}(h)$ for all $h \in \mathcal{H}$ simultaneously. For example, assume that the following bound holds for all $h \in \mathcal{H}$:

$$\left|\mathcal{L}(h) - \hat{\mathcal{L}}(h)\right| \leq \epsilon_{\text{GEN}} \tag{2.1}$$

and assume that the learner uses an algorithm $\mathcal{A}$ which given a sample S, returns an approximate empirical risk minimiser, $\hat{h}$ satisfying:

$$\hat{\mathcal{L}}(\hat{h}) \leq \hat{\mathcal{L}}(h) + \epsilon_{\text{OPT}} \quad \forall h \in \mathcal{H}. \tag{2.2}$$

Combining both bounds we find that for all $h \in \mathcal{H}$ we have:

$$\begin{aligned}\mathcal{L}(\hat{h}) - \mathcal{L}(h) &= \left[\mathcal{L}(\hat{h} - \hat{\mathcal{L}}(\hat{h})\right] + \left[\hat{\mathcal{L}}(\hat{h}) - \hat{\mathcal{L}}(h)\right] + \left[\mathcal{L}(h) - \hat{\mathcal{L}}(h)\right] \\ &\leq 2\epsilon_{\text{GEN}} + \epsilon_{\text{OPT}}.\end{aligned}$$

As already mentioned, many empirical risk minimisation problems take the form of convex minimisation problems, and as such bounds in the form of (2.2) can often be derived directly from convex optimisation methods. Bounds of the form (2.1) are known as generalisation bounds, and typically rely on some measure of the statistical complexity of the hypothesis class $\mathcal{H}$ and the smoothness of the loss function ℓ. One popular measure of statistical complexity is the Rademacher complexity, $\mathcal{R}_m(\mathcal{H})$ (Koltchinskii, 2001; Bartlett and Mendelson, 2002):

$$\mathcal{R}_m(\mathcal{H}) = \mathbb{E}\left[\sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} h(x_i)\sigma_i\right]$$

where σ_i are independent random random variables taking values in $\{-1, +1\}$ with equal probability, and $(x_1, \ldots, x_m)$ are sampled independently and identically from the distribution $\mathcal{D}$. Intuitively, the Rademacher complexity captures how well a hypothesis class can correlate with a random noise sequence, and thus serves as a measure of the expressiveness of $\mathcal{H}$. The following theorem provides an example of a generalisation bound which depends on the Rademacher complexity of the underlying hypothesis class $\mathcal{H}$.

Theorem 2.2 (Bartlett and Mendelson (2002))**.** *Assume the loss ℓ is Lipschitz with respect to its first argument with Lipschitz constant L_ℓ and that ℓ is bounded by c. For any $\delta > 0$ and with probability at least $1 - \delta$ simultaneously for all $h \in \mathcal{H}$, we have that*

$$\mathcal{L}(h) \leq \hat{\mathcal{L}}(h) + 2L_\ell \mathcal{R}_m(\mathcal{H}) + c\sqrt{\frac{\log(1/\delta)}{2m}}$$

Other measures of statistical complexity include the Gaussian complexity, the VC dimension (Vapnik and Chervonenkis, 2015), and the maximum discrepancy (Bartlett et al., 2002). Given Theorem 2.2, if one can bound the Rademacher complexity of a chosen hypothesis class $\mathcal{H}$, then sample complexity guarantees will follow immediately for ERM-based algorithms. One popular class is the set of bounded linear predictors:

$$\mathcal{H}_{\text{lin}} = \{\omega \,:\, F(w) \leq c\}$$

where the prediction of w given x is simply the inner product $\langle \omega, x \rangle$ and $F(w)$ is a strongly convex complexity function designed to limit the expressiveness of the linear predictors considered. For example, a common choice for F is the squared L_2-norm, $F(w) = \|\omega\|^2$. The following theorem provides an upper bound on the Rademacher complexity of such classes of linear predictors:

Theorem 2.3 (Kakade et al. (2008))**.** *Let C be a closed convex set and $\mathcal{X} = \{x : \|x\| \leq X\}$. Further, let $F : C \to \mathbb{R}$ be a σ-strongly convex function with respect to the dual norm $\|\cdot\|_*$ such that $\inf_{w \in C} F(w) = 0$. Define $\mathcal{W} = \{w \in C : F(w) \leq W_\star^2\}$. Then we have*

$$\mathcal{R}_m(\mathcal{W}) \leq X W_\star \sqrt{\frac{2}{\sigma m}}$$

In combination with Theorem 2.2, Theorem 2.3 tells us that ERM-based algorithms are efficient PAC learning algorithms for linear prediction problems with bounded Lipschitz loss functions. Note that this bound of the Rademacher complexity is tight in the case of the 2-norm (Awasthi et al., 2020b).

2.1.2 Adversarial Learning

Recall the email spam classification setting discussed throughout Chapter 1. In this setting, email spammers can slowly learn the spam filter adopted by the learner, and then design future emails with the intention of bypassing the filter. As a result, the examples the learner observes after the training phase may differ significantly from those in the training dataset. In other words, the spam classification problem is nonstationary, and the distribution $\mathcal{D}$, from which examples are sampled, changes as a result of the learner's choice of filter. Therefore, the PAC framework outlined in the previous section is no longer suitable. This is because the PAC framework assumes that all examples are sampled from a fixed distribution $\mathcal{D}$ which does not change.

Adversarial learning settings seek to relax this stationarity assumption, and assume that an adversary may manipulate individual data points, or the underlying sample distribution, with the goal of maximising the learner's loss. Adversarial frameworks for machine learning vary wildly depending on the capabilities of the adversary. We refer to adversarial models in which the adversary can manipulate data sampled during the training phase as data poisoning settings (Biggio and Roli, 2018). In contrast, we refer to models in which the adversary may alter data sampled after the training phase as evasion models (Biggio and Roli, 2018). In this thesis, we will focus on evasion problems. Note that email spam classification can be naturally cast as an evasion problem.

Early evasion models for spam classification were proposed by Dalvi et al. (2004), who studied an adversarial setting in which the adversary only has control over positively

labelled examples and seeks to have them classified negatively. Within this setting, a learner must choose a naive Bayes classifier to filter spam emails. The adversary receives utility for modifying individual spam emails so that they bypass the naive Bayes classifier chosen by the learner. When modifying a data point from, x to x', the adversary pays a penalty $c(x, x')$, defined by a semi-metric $c : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. If the cost of modifying a positively classified spam email outweighs the utility gained, then it is assumed the adversary makes no modification. Given a classifier, the authors show that computing an optimal modification is equivalent to solving a linear program (LP). Moreover, the authors provide an efficient algorithm for computing the optimal Bayes classifier, given that the adversary will alter future data points optimally after the training phase.

Similar settings were considered by Lowd and Meek (2005) from the perspective of the adversary. In their model, the adversary is initially unaware of the classifier chosen by the learner, but may repeatedly query the classifier by submitting input examples to learn how they are classified. When submitting an input example, $x \in \mathcal{X}$, the adversary pays a cost $c(x)$ according to a cost function $c : \mathcal{X} \to \mathbb{R}$. The goal of the adversary is to find a minimum cost input example which is classified negatively within a polynomial number of queries to the classifier. For linear classifiers, Lowd and Meek (2005) show that an adversary can learn an approximately optimal input sample in a polynomial number of queries. Nelson et al. (2012) extended this result to the family of convex inducing binary classifiers, which partition the space into two sets, one of which being convex.

Note that the aforementioned settings share many similarities that are typical of most evasion problems. After the training phase, data points are sampled from the same distribution as the training dataset, but are first given to an adversary who can modify the sampled input at a cost. We refer to such adversaries as cost sensitive. On the other hand, both settings differ in terms of the knowledge the adversary possesses regarding the learner's employed classifier. In the case of Dalvi et al. (2004), the adversary has full knowledge of the classifier adopted by the learner. Meanwhile, in the case of Lowd and Meek (2005) and Nelson et al. (2012), the adversary has no initial knowledge of the classifier and must learn it over time. In the terminology adopted by Biggio and Roli (2018) the latter settings are gray-box, or limited knowledge, whilst the setting proposed by Dalvi et al. (2004) is white-box, or perfect-knowledge.

Instead of limiting the capability of adversaries via a cost function, one can also consider adversaries restricting to a specific set of perturbations. That is, given a hypothesis $h \in \mathcal{H}$, chosen by the learner, and an input example $x \in \mathcal{X}$ sampled from $\mathcal{D}$, the goal of the adversary is to find a perturbation of x from a neighborhood $\mathcal{N}(x)$, which maximises the learner's loss:

$$x^\star \in \arg \max_{\tilde{x} \in \mathcal{N}(x)} \ell(h(\tilde{x}), y).$$

As a result, learning takes a minimax form in which the learner is trying to find a hypothesis $h^\star$ which minimises its loss in expectation, under the assumption that the adversary will choose a loss maximising perturbation:

$$h^\star \in \arg\min_{h\in\mathcal{H}} \mathbb{E}\left[\max_{\tilde{x}\in\mathcal{N}(x)} \ell(h(\tilde{x}), y)\right].$$

Note that the optimisation objective above is a natural generalisation of risk, typically referred to as the adversarial risk. As in the classical supervised machine learning setting, the learner does not have access to the distribution $\mathcal{D}$. A natural approach is to minimise an empirical version of the adversarial risk, as we did for the risk in the classical supervised setting:

$$\min_{h\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m} \max_{\tilde{x}_i\in\mathcal{N}(x_i)} \ell(h(\tilde{x}_i), y_i).$$

This approach is known as adversarial empirical risk minimisation. Problems of this type have been studied by the optimisation community under the topic of robust optimisation. In this context, $\mathcal{N}(x)$ is referred to as the uncertainty set associated with the point x. In many cases, different uncertainty sets correspond to different regularisation approaches. For example, in the case of linear prediction with the square loss function, adopting the L_2-ball of fixed radius around x as an uncertainty set corresponds to performing ridge regression on the original data (El Ghaoui and Lebret, 1997). Meanwhile, Globerson and Roweis (2006) study a robust optimisation problem in which up to K features may be deleted by the adversary, and show that the resulting minimax problem has a natural convex and quadratic formulation.

Note that while robust optimisation approaches mitigate concerns regarding the computational tractability of adversarial risk minimisation, issues of sample complexity remain. Within the statistical learning community, significant effort has been made to generalise traditional measures of statistical complexity, such as the Rademacher complexity, to adversarial contexts. In particular, Yin et al. (2019) studied an adversarial version of Rademacher complexity for adversaries bounded in L_∞-norm. In particular, the authors show that the adversarial Rademacher complexity of binary linear classifiers in this setting is always larger than the Rademacher complexity in the classical supervised setting. Awasthi et al. (2020a) provide a generalised version of this result which holds for adversaries bounded in any L_p-norm. Similarly, Cullina et al. (2018) develop an adversarial counterpart to VC dimension, and show that the adversarial VC dimension is equal to the VC dimension for halfspace classifiers for a wide range of adversaries.

A parallel line of work has sought to establish the relationship between adversarial robustness and generalisation. In the linear regression setting, Javanmard et al. (2020) characterize the precise trade-offs between standard and adversarial risk. Meanwhile,

Montasser et al. (2019) provide examples of hypothesis sets of finite VC dimension that can only be learner improperly in the adversarial setting. Similarly, Schmidt et al. (2018) design a simple learning model which suffers from significantly higher sample complexity in the adversarial setting.

As discussed in Chapter 1, adversarial learning models are often overly pessimistic. Whilst the standard PAC learning framework assumes that data providing agents have no incentives, adversarial learning models assume that the incentives of each data provider are completely misaligned with the learner. Reality typically lies somewhere between these two extremes. This motivates our investigation of strategic machine learning settings in the following sections.

2.1.3 Strategic Classification

Closely related to adversarial learning is the strategic classification framework. Introduced by Hardt et al. (2016), the learner plays a role of a jury, and the adversary plays the role of a contestant. The jury is tasked with selecting a binary classifier $h \in \mathcal{H}$ using a training set sampled from a distribution $\mathcal{D}$. After the training phase, the contestant can manipulate each newly sampled data point at a cost, with the goal of obtaining a positive classification. In other words, given a sampled input x, the contestant aims to produce a cost-effective manipulation $x^\star$:

$$x^\star \in \arg\min_{\tilde{x} \in \mathcal{X}} \mathbb{1}[h(\tilde{x}) \neq 1] + c(x, \tilde{x}).$$

The goal of the jury is then to minimise their expected loss under the assumption that the contestant will best respond. That is to find $h^\star \in \mathcal{H}$ such that:

$$h^\star \in \arg\min_{h \in \mathcal{H}} \mathbb{1}[h(x^\star) \neq y].$$

In what follows, we refer to the objective above as the strategic risk, as it is an analog for risk in the strategic classification setting. In their seminal work, Hardt et al. (2016) provide a polynomial time algorithm with sample complexity guarantees for strategic risk based on Rademacher complexity, under the assumption that the cost function c satisfies a separability condition.

Note that the strategic classification setting has subtle differences from the previously described adversarial settings. For instance, consider an example of the form $(x, 1)$. In this case, the contestant and the jury have aligned incentives. In other words, both the jury and the contestant want the example x to be classified positively. This is never the case in the adversarial settings considered above. In other words, the strategic classification setting allows data providing agents to possess incentives and goals that do not directly oppose those of the learner.

Since the initial work of Hardt et al. (2016), strategic classification has received significant attention from both the machine learning and mechanism design communities. Chen et al. (2020) consider an online version of strategic classification in which the contestant may report any point within a ball of fixed radius surrounding a sample data point. Within, this setting the authors provide an algorithm with finite-time regret guarantees, however this algorithm requires access to a special in-oracle which determines whether a polytope lies within one of several halfspaces. Similarly, Dong et al. (2018) studies an online strategic classification setting in which negatively classified data points are strategic, and derive conditions on the cost function which guarantee the strategic loss incurred on each time step is a convex function. This allows standard online convex optimisation algorithms to be applied in order to achieve finite-time regret guarantees.

Meanwhile, a strategic analog of ERM for strategic classification has been investigated by Sundaram et al. (2021), who define a strategic version of VC dimension, called the strategic VC (SVC) dimension. In essence, the strategic VC dimension of a hypothesis class $\mathcal{H}$ can be viewed as the VC dimension of the class $\mathcal{H}_{\text{strat}} = \{h(r(h,\cdot)) : h \in \mathcal{H}\}$ where $r : \mathcal{H} \times \mathcal{X} \to \mathcal{X}$ is the contestant's best-response function to the hypothesis $h \in \mathcal{H}$ given a sampled input $x \in \mathcal{X}$:

$$r(h,x) = \arg\min_{\tilde{x}\in\mathcal{X}} \mathbb{1}\left[h(\tilde{x} \neq 1\right] + c(\tilde{x},x).$$

Put differently, $\mathcal{H}_{\text{strat}}$ is the set of classifiers which the learner can practically implement through their choice of hypothesis. Thus, if $\mathcal{H}_{\text{strat}}$ has bounded statistical complexity, one would expect a strategic version ERM to be an efficient PAC learning algorithm. In particular, Sundaram et al. (2021) show that the SVC dimension of a hypothesis class can be arbitrarily larger than its VC dimension even in the case of linear classifiers. This implies that learning optimal classifiers in the presence of strategic agents is potentially intractable unless reasonable assumptions are enforced on the cost function of the contestant. However, Sundaram et al. (2021) provide simple bounds on the SVC dimension under several reasonable assumptions. For example, the authors show that the SVC dimension is at most two under the separability assumption introduced by Hardt et al. (2016).

In contrast, Zhang and Conitzer (2021) study a strategic classification setting in which an uncertainty set for each data point is defined via a manipulation graph $G = (\mathcal{X}, E)$. An agent is only allowed to modify a sampled point from x to $\tilde{x}$ if there is an edge connecting x to $\tilde{x}$ in G. Within this setting, the Zhang and Conitzer (2021) consider a version of ERM that only optimises over hypotheses which are incentive compatible. A hypothesis is incentive compatible if and only if it is always optimal for the contestant to report the originally sampled point x over any neighboring manipulation $\tilde{x}$. In particular, it is shown that the VC dimension of incentive compatible hypotheses is bounded by the cardinality of the maximal independent set in the manipulation

graph. Moreover, if the manipulation graph satisfies a certain transitivity property, then restricting to incentive-compatible hypotheses is without loss of generality. Thus, under reasonable assumptions on the manipulation graph, one can recover the sample complexity guarantees of linear classification in the standard PAC setting.

A similar setting is studied by Lechner and Urner (2022), who investigate a strategic version of the 0-1 loss, which models the loss experienced by a classifier in a strategic environment. More specifically, the strategic loss associated with a hypothesis class h is defined as follows:

$$\ell^{\rightarrow}(h,x,y) = \begin{cases} 1 & \text{if } h(x) \neq y \\ 1 & \text{if } h(x) = 0 \text{ and } \exists \tilde{x} \text{ with } x \rightarrow \tilde{x} \text{ and } h(\tilde{x}) = 1 \\ 0 & \text{otherwise} \end{cases}$$

It follows trivially that $\ell^{\rightarrow} \leq \ell^{0\text{-}1} + \ell^{\perp}$, where $\ell^{0\text{-}1}$ is the standard 0-1 loss and $\ell^{\perp}$ denotes the strategic component loss:

$$\ell^{\perp} = \mathbb{1}\left[h(x) = 0 \wedge \exists \tilde{x} : x \rightarrow \tilde{x}, h(\tilde{x}) = 1\right].$$

In particular, Lechner and Urner (2022) show that the VC dimension of the loss class induced by the strategic loss on a hypothesis class $\mathcal{H}$ is upper bounded by the VC dimension of $\mathcal{H}$ plus the VC dimension of the loss class induced on $\mathcal{H}$ by the strategic component loss. As a result, the possibility of learnability depends intrinsically on the VC dimension of the loss class induced by the strategic component loss, and thus the underlying manipulation graph.

From the optimisation perspective, a natural approach to minimising strategic risk is to adopt a ERM-based approach, called strategic ERM (SERM), which accounts for the actions of the contestant. For example, for a cost sensitive adversary, who may report any $\tilde{x} \in \mathcal{X}$, SERM takes the following form:

$$\begin{aligned} &\min_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \ell(h(x_i^{\star}, y_i)) \\ &\text{s.t.} \quad x_i^{\star} = \arg\min_{\tilde{x} \in \mathcal{X}} \mathbb{1}\left[h(\tilde{x}_i) \neq 1\right] + c(\tilde{x}_i, x_i). \end{aligned}$$

Observe that SERM corresponds to ERM on $\mathcal{H}_{\text{strat}}$. Note that SERM is a bilevel optimisation problem, which are in general NP-hard. Moreover, the minimax structure present in adversarial ERM is not present in SERM. As a result, SERM-based algorithms typically require special approaches which exploit assumptions regarding the capability of the contestant and its cost function. In deriving their algorithm for separable cost functions, Hardt et al. (2016) show that SERM reduces to a one-parameter optimisation problem. Similarly, Sundaram et al. (2021) show that SERM for linear classification with a semi-norm induced cost function can be

reformulated as a convex optimisation problem under the assumption that the contestant is essentially adversarial. For the manipulation graph setting, Zhang and Conitzer (2021) reduce incentive compatible ERM to solving a min-cut problem which can be solved polynomial time.

Whilst effectively capturing the incentives of data providers in many settings, strategic classification is limited in terms of its flexibility. More specifically, it is assumed that there is one classification simultaneous desirable to all agents. Thus, strategic classification fails to meet Requirement 1a. In addition, strategic classification is intrinsically limited to classification problems. In the next section, we will discuss Stackelberg prediction games and performative prediction, which generalise strategic classification.

2.1.4 Stackelberg Prediction Games

Observe that most of the problems settings considered so far are examples of Stackelberg competition. First defined in the context of economic markets (Von Stackelberg, 2010), Stackelberg competition refers to any game theoretic setting in which one player, known as the leader, must first choose a strategy, before a second a player, known as the follower, responds with a strategy of their own. Note that the follower has an intrinsic advantage, typically having full knowledge of the strategy employed by the leader before making their decision. In the learning settings considered above, the learner plays the role of the leader, and must commit to a strategy, that is a hypothesis, first. Meanwhile, a data providing agent, that may be adversarial or strategic, plays the role of the follower, and modifies the inputs they sample, with full knowledge of the hypothesis chosen by the learner. The best the learner can guarantee is a Stackelberg equilibrium, that is, the hypothesis, or strategy, with the lowest risk given that the data providing agent will best respond.

In what follows, we will investigate the Stackelberg prediction game (SPG) model (Brückner and Scheffer, 2011), which makes this connection between Stackelberg competition and learning explicit. SPGs are particularly attractive due to their flexibility, allowing a wide range of agent goals and incentives to be modeled. In the SPG setting, learning is once again modeled as a Stackelberg game between the learner and a data providing agent. As before, the learner is tasked with selecting a hypothesis h from a hypothesis class $\mathcal{H}$. The learner evaluates the performance of each hypothesis on each sampled input example via their loss function $\ell_{-1} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$. After the learner selects a hypothesis, a data provider may change the underlying sampling distribution $\mathcal{D}$ to a new distribution $\tilde{\mathcal{D}}$, paying a cost, $C(\mathcal{D}, \tilde{\mathcal{D}})$, to do so. The data provider evaluates the performance of the distribution they have chosen via their own loss function $\ell_{+1} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$. The goal of the data provider is to select a distribution $\mathcal{D}^\star$ which provides the optimal trade-off between the expected risk with

respect to ℓ_{+1} and the cost of modification $C(\mathcal{D}, \mathcal{D}^\star)$:

$$\mathcal{D}^\star \in \arg\min_{\tilde{\mathcal{D}}} \mathbb{E}_{\tilde{\mathcal{D}}} \left[\ell_{+1}(h(x), y)\right] + \gamma C(\mathcal{D}, \tilde{\mathcal{D}}).$$

Note the presence of the hyperparameter $\gamma > 0$, which captures the trade-off between the performance of a given distribution $\tilde{\mathcal{D}}$ and its cost. The goal of the learner is to select a hypothesis $h^\star$ which minimises the expected loss with respect to ℓ_{-1} under the assumption that the data provider will best respond:

$$\begin{aligned} h^\star \in \arg\min_{h \in \mathcal{H}} \; & \mathbb{E}_{\mathcal{D}^\star} \left[\ell_{-1}(h(x), y)\right] \\ \text{s.t.} \quad & \mathcal{D}^\star \in \arg\min_{\tilde{\mathcal{D}}} \mathbb{E}_{\tilde{\mathcal{D}}} \left[\ell_{+1}(h(x), y)\right] + \gamma C(\mathcal{D}, \tilde{\mathcal{D}}). \end{aligned} \tag{2.3}$$

In other words, the learner aims to find a Stackelberg equilibrium. Of course, the learner does not have access to the distribution $\mathcal{D}$, so cannot solve Problem (2.3) directly. However, it is assumed that the learner does have access to a training dataset of input examples sampled from $\mathcal{D}$. In the real world, such a dataset may be obtained through an expensive verification procedure. Using their training dataset, the learner can compute a hypothesis $\hat{h}$ that approximately minimises an empirical version of Problem (2.3):

$$\begin{aligned} & \min_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \ell_{+1}(h(x_i^\star), y_i) \\ & \text{s.t.} \quad x_i^\star = \arg\min_{\tilde{x} \in \mathcal{X}} \ell_{-1}(h(\tilde{x}_i, y_i) + \gamma c(x_i \tilde{x}_i) \quad [i] \in m \end{aligned} \tag{2.4}$$

Note that, when moving from Problem (2.3) to Problem (2.4) an implicit assumption has been made regarding the cost function C. In particular, it is assumed that the data provider manipulates the data distribution $\mathcal{D}$ on an instant-wise basis, and that the cost of manipulating an input example from x to $\tilde{x}$ is given by $c(x, \tilde{x})$, where $c : \mathcal{X} \times \mathcal{X}$ is a cost function. Practically speaking, this assumption is not too severe, and is a realistic model of how data distributions are manipulated in real world settings. In addition, note that if we take both ℓ_{-1} and ℓ_{+1} to be the 0-1 loss, we recover the SERM problem for strategic classification. As a result, Problem (2.4) can be viewed as a generalised version of SERM.

As with SERM in the strategic classification setting, Problem (2.4) is a bilevel optimisation problem, and thus NP-hard for general loss functions ℓ_{-1} and ℓ_{+1}. However, under the assumption that both loss functions are convex, and that ℓ_{-1} is twice continuously differentiable, Brückner and Scheffer (2011) show that a stationary point of Problem (2.4) can be found via sequential quadratic programming. In Chapter 3, we will consider a special case of Problem (2.4), in which the incentives of the data provider are captured by an additional set of labels. For this setting, we will present a

polynomial time procedure which converges to global optima, improving upon the generic procedure of Brückner and Scheffer (2011) in this special case.

Closely related to Stackelberg prediction games is performative prediction (Perdomo et al., 2020). In performative prediction, each hypothesis, $h \in \mathcal{H}$, is associated with a distribution $\mathcal{D}(h)$ which describes the data the learner is likely to encounter if h is selected. As a result, the risk associated with any hypothesis h is the expected loss under the corresponding distribution $\mathcal{D}(h)$:

$$\mathcal{L}(h) = \mathbb{E}_{\mathcal{D}(h)} \left[\ell(h(x), y) \right]$$

In other words, the learner wants to identify a hypothesis which performs well under the distribution it induces. The quantity above is known as the performative risk. Unlike strategic classification and SPGs, no structural assumptions are made with respect to the function $D(h)$. As a result, performative prediction generalises both strategic classification and SPGs. Perdomo et al. (2020) outline two reasonable solution concepts for performative prediction. The first, known as performative optima, involves returning a hypothesis which minimises performative risk:

$$h^\star \in \arg\min_{h \in \mathcal{H}} \mathbb{E}_{\mathcal{D}(h)} \left[\ell(h(x), y) \right].$$

Note that this corresponds to identifying a Stackelberg equilbrium within the context of SPGs. An easier goal is to aim for performative stability, that is, to find a hypothesis which is a best response to the distribution it induces:

$$h^\star \in \arg\min_{h \in \mathcal{H}} \mathbb{E}_{\mathcal{D}(h^\star)} \left[\ell(h(x), y) \right].$$

Perdomo et al. (2020) presents a number of approaches for finding performatively stables hypotheses. The first is repeated empirical risk minimisation (RERM) in which the learner selects an initial hypothesis, h_1, and receives a training sample, S_1, from $\mathcal{D}(h_1)$. After this, the learner selects a new hypothesis, h_2, which minimises the empirical risk on S_1. Once again the learner receives a sample S_2, sampled from $\mathcal{D}(h_2)$, and finds a new hypothesis, h_3, which minimises the empirical risk on S_2. This process repeats predictably ad infimum. Under mild assumptions, Perdomo et al. (2020) show that RERM converges to a performatively stable hypothesis at a linear rate with high probability. Additionally, Perdomo et al. (2020) illustrate linear convergence for a similar method, where each step of empirical risk minimisation is replaced with a projected gradient step. Perdomo et al. (2020) also show that performatively stable hypotheses are good approximations of performative optima when the loss function is strongly convex and Lipschitz.

Investigating further, Drusvyatskiy and Xiao (2020) provide convergence guarantees under similar assumptions for a large range of descent methods. Similarly,

Mendler-Dünner et al. (2020) give linear convergence guarantees for stochastic gradient descent. Both methods improve upon RERM in the sense that only a single data sample is needed at each iteration. More recently, Miller et al. (2021) show that, under mild assumptions, finding performative optima corresponds to solving a convex optimisation problem when $\mathcal{D}(h)$ is mixture dominant. Note that many common strategic classification problems can be cast as performative prediction problems where mixture dominance holds. Thus, the result of Miller et al. (2021) implies that a wide variety of strategic classification problems can be solved using well established convex optimisation strategies.

Both performative prediction and Stackelberg prediction games offer sufficient generalisations of strategic classification to meet Requirement 1a. In Chapter 3, we choose to focus on SPGs, and linear regression, due to their lack of study within the strategic learning community.

2.2 Sequential Deployment of Reusable Resources

Next, we shift focus, and investigate work relevant to Problem Domain 2. First in Section 2.2.1, we review classical variations of the MAB problem, which has been used extensively to model resource allocation problems. As we will see, conventional methods for bandit learning typically aim to do well compared to policies which repeatedly deploy the same resource. In real world settings, the usefulness of a given resource may vary wildly over time depending on the underlying environment. Therefore, policies which rely on a single resource are unlikely to be effective, and thus do not serve as meaningful performance benchmarks (violating Requirement 2a). Hence, in practical settings, decision makers must be adaptive and deploy the right resources at the right time. This motivates our review of nonstochastic MAB settings in Section 2.2.2 wherein the decision maker is tasked with adaptively tracking the best resource through time.

Additionally, conventional MAB problems do not model resource availability explicitly (Requirement 2b). As we saw in Chapter 1, resources often become unavailable for short periods of time in real world scenarios. Therefore, in Section 2.2.3, we review constrained bandit models, that limit when and how often a given action can be taken. In particular, we pay special attention to the stochastic blocking bandit model (Basu et al., 2019), in which resources become unavailable, or *blocked*, for a period of time immediately after use. In Chapter 4, we present a new MAB problem designed to model sequential resource allocation problems, called the adversarial blocking bandits setting, which can be viewed as an amalgamation of the blocking and nonstochastic bandit settings investigated in this chapter.

2.2.1 The Stochastic Multi-Armed Bandit Problem

To begin, we first describe the classical the multi-armed bandit (MAB) problem. A decision maker is given access to a set of K actions. The decision maker is tasked with producing a sequence of actions over T discrete time steps. More specifically, at a given time step $t \in [T]$, a decision maker must take one action, $i \in [K]$. Upon taking an action i on time step t, the learner receives a reward, $X_t^i \in \mathbb{R}$. In what follows, we use i_t to denote the action chosen by the decision maker at time step t. The goal of the decision maker is to maximise their cumulative reward over the time horizon:

$$\max \sum_{t=1}^{T} X_t^{i_t}.$$

From one perspective, each action can be viewed as a slot machine, or a one-armed bandit. The decision maker can be viewed as a gambler, who at each time step, must choose a slot machine, or arm to pull. The goal of the gambler is to pull the slot machine on each time step that has the best pay-out. Given this point of view, we will use the terms arm and action interchangeably. Note that resource deployment problems often map directly to MAB problems. Each action available to the decision maker may represent a resource available for deployment. The reward, $X_t^{i_t}$, for an action i may reflect the utility of redeploying the corresponding resource given the current state of the environment at time step t.

Note that, so far, we have placed no assumptions on how the rewards, $X_t^{i_t}$, are generated. The first MAB problems investigated were stochastic (Lai and Robbins, 1985). In a stochastic MAB problem, it is assumed that the reward, $X_t^{i_t}$, associated with arm i at time step t, is sampled from an unknown distribution associated with arm i. We use $\mu_i \in \mathbb{R}$ to denote the mean reward associated with each action i. Clearly, within the stochastic context, it is optimal in expectation for the decision maker to repeatedly pull the arm with highest mean reward. In what follows, we will use to $\mu^\star$ to denote the highest mean reward:

$$\mu^\star = \max_{i \in [K]} \mu_i$$

Observe that stochastic MAB problems exhibit the exploration-exploitation trade-off common to many machine learning problems (Sutton and Barto, 2018; Bubeck and Cesa-Bianchi, 2012). In particular, the decision maker must explore, by trying each arm enough times so that they may correctly identify the arm with highest mean reward. However, if the decision maker spends too long exploring, they will not have enough time to exploit their knowledge and pull the arm which they believe to be optimal. On the other hand, if the decision maker spends too long pulling the arm they believe to be best, they risk not testing each arm sufficiently, and misidentifying the arm with optimal mean reward.

In short, the decision maker is in pursuit of a policy which optimally trades off exploration with exploitation. An intuitive way to evaluate a policy is to benchmark against the best policy in expectation. This leads us directly to the definition of regret.

Definition 2.4. The regret R_T of a decision maker is given by:

$$R_T = T\mu^\star - \sum_{t=1}^{T} X_t^{i_t}.$$

where $\mu^\star$ denotes the maximum mean reward: $\mu^\star = \max_{i \in [K]} \mu_i$.

Ideally, the decision maker would like to adopt a policy which achieves zero regret. However, this goal is unrealistic, as the decision maker has no initial knowledge regarding the mean rewards of each arm, and thus must risk incurring positive regret by exploring. Instead, the decision maker may hope to find a policy whose regret incurred per time step vanishes as the time horizon lengthens. More formally, the decision maker may hope to find a policy whose average regret grows sublinearly in in the time horizon T:

$$\lim_{T \to \infty} \frac{\mathbb{E}\left[R_T\right]}{T} = 0. \tag{2.5}$$

In particular, the decision maker can hope for an even stronger property known as consistency (Lattimore and Szepesvári, 2020).

Definition 2.5. A policy is consistent over a set of MAB problem instances if for all instances and all $p > 0$ it holds that

$$\frac{\mathbb{E}\left[R_T\right]}{T^p} = 0 \quad \text{as } T \to \infty.$$

In other words, a policy is consistent if the regret grows subpolynomially with the time horizon T. More generally, the decision maker is interested in finding a policy whose regret grows at the optimal rate on each problem instance. Note that the consistency and efficiency of a given policy depends on the class of MAB problem instances considered. Following the notation of Lattimore and Szepesvári (2020), we use $\nu = (P_i : 1 \in [K])$ to denote a stochastic MAB instance in which action i is associated with a reward distribution P_i. Two commonly studied classes of MAB instances, are those with rewards from $[0, 1]$, $\mathcal{E}_{[0,1]}$, and those that are 1-subgaussian, $\mathcal{E}_{SG}$:

$$\mathcal{E}_{[0,1]} = \{\nu = (P_i)_i : \text{supp}(P_i) \subseteq [0, 1]\}$$
$$\mathcal{E}_{SG} = \{\nu = (P_i)_i : P_i \text{ is 1-subgaussian for all } i\}.$$

In addition, we write $R_T(\nu)$ to describe the regret accumulated over a time horizon of length T on a bandit instance ν. Asymptotic consistency was first studied in the seminal paper of Lai and Robbins (1985), who establish asymptotic lower bounds on regret for single parameter MAB classes of order $\Omega(\log(T))$. This work was extended

to nonparametric MAB problem classes by Burnetas and Katehakis (1996). Note that these asymptotic bounds are instance dependent. That is, they take the form:

$$\liminf_{T\to\infty} \frac{\mathbb{E}\left[R_T(\nu)\right]}{\log(T)} \geq c^\star(\nu, \mathcal{E})$$

in which $c^\star(\nu, \mathcal{E})$ is a constant with respect to T that depends on the problem instance ν and the problem class $\mathcal{E}$. In contrast, general minimax bounds, which are instance independent, have also been established. In particular, (Auer et al., 2002) show that any policy must incur regret of order $\Omega(\sqrt{KT})$ on some worst-case instance in $\mathcal{E}_{[0,1]}$. Furthermore, finite-time instance-dependent bounds have been established by Garivier et al. (2019) and Lattimore (2018) of similar, or identical order to the asymptotic instance-dependent bounds already mentioned.

In what follows, we will outline two approaches for designing policies which aim to provide finite-time regret guarantees that match the aforementioned lower bounds described above. First, we will consider the explore-then-commit (ETC) framework, before describing the family of upper confidence bound (UCB) algorithms, which rely upon the principle of optimism in the face of uncertainty. ETC and UCB differ in that the former splits the time horizon into explicit exploration and exploitation phases, whilst the latter makes no such clear separation.

In the ETC framework, the decision maker spends the first mK rounds pulling each arm m times. This is known as the exploration phase, in which the decision maker computes an empirical estimate $\hat{\mu}_i$ of the mean reward associated with each arm. After the exploration phase, the decision maker pulls the arm with highest empirical mean reward on every round. Let $\Delta_i = \mu^\star - \mu_i$ denote the expected performance gap between arm i and the best arm. It is clear in the first mK rounds that any ETC algorithm will incur a regret of $m\sum_{i=1}^K \Delta_i$. Similarly, it is easy to see that the expected number of pulls of arm i in the remaining $T - mK$ rounds is given by $(T - mk)\mathbb{P}(\hat{\mu}_i \geq \max_{j\neq i} \hat{\mu}_j)$. Thus the average regret of any ETC algorithm is given by:

$$m\sum_{i=1}^K \Delta_i + (T - mK)\sum_{i=1}^K \mathbb{P}(\hat{\mu}_i \geq \max_{j\neq i} \hat{\mu}_j)\Delta_i$$

Note that both terms depend directly on m and as such the choice of m is extremely important when it comes to achieving good regret bounds. In addition, both terms depend on the performance gaps Δ_i. As a result, ETC algorithms which use additional knowledge regarding the performance gaps Δ_i when setting m typically perform better than those that do not. When the decision maker has no knowledge of the performance gaps Δ_i, m must be set dynamically based on the rewards observed to obtain meaningful guarantees. Unfortunately, even in the case of $K = 2$ Garivier et al. (2016) shows that ETC algorithms are asymptotically suboptimal under Gaussian reward distributions both, whether performance gaps are known or not.

In contrast to ETC algorithms, UCB algorithms do not partition the time horizon. Instead, UCB algorithms maintain an index for each arm. Each index used by a UCB algorithm corresponds to an upper confidence bound on the mean reward of each arm, based on the empirical means of rewards sampled so far. At each time step, a UCB algorithm simply selects the arm with the largest index. That is, UCB is optimistic in the face of uncertainty. By this, we mean that out of the mean rewards which are likely for a given arm, UCB algorithms assume that the arm has the best mean reward possible. For example, in the case of 1-subgaussian rewards the empirical mean reward of any arm satisfies the following confidence bound:

$$\text{UCB}_i(t) = \begin{cases} \infty & \text{if } N_i(t) = 0 \\ \hat{\mu}_i + \sqrt{\frac{2\log(1/\delta)}{N_i(t)}} \end{cases}$$

where $N_i(t)$ corresponds to the number of times the decision maker has taken action i so far. The corresponding UCB policy simply takes the action with maximum confidence bound:

$$i_t = \max_{i \in [K]} \text{UCB}_i(t).$$

Note that the confidence bound above does not depend on the performance gaps Δ_i, and as such this version of UCB can be run without knowledge of the performance gaps Δ_i. In fact, the class of UCB algorithms is minimax optimal regardless of whether the performance gaps are known or not (Bubeck and Cesa-Bianchi, 2012; Garivier and Cappé, 2011).

Within the context of sequential resource deployment, stochastic MAB problems often do not reflect reality. For example, in stochastic MAB problems the reward distribution associated with each action is fixed and does not change with time. In real world settings, the efficacy of a deployed resource varies through time depending on the underlying state of the environment. Additionally, resources are often unavailable for a period once deployed. In other words, the availability of resources may vary throughout time, depending on the past actions of the decision maker. Several extensions to the stochastic MAB problem has been curated which seek to mitigate these issues. In the section that follows, we will review both nonstochastic and nonstationary MAB problems which aim to adaptively track the best action through time. After this, we investigate constrained bandit models, which restrict the set of actions available to the decision maker on each time step.

2.2.2 Nonstationary Bandit Models

Nonstationary stochastic MAB problems relax the stationarity assumption implicit in the classical stochastic setting. That is, in nonstationary MAB problems, the reward distribution associated with each arm may change over time. In what follows, we will

use μ_t^i to denote the mean reward of arm i at time t. Of course, the definition for regret given in the previous section is no longer a meaningful measure of the decision maker's performance. In other words, the decision maker cannot expect a fixed policy, which pulls the same arm repeatedly, to be optimal in a nonstationary setting. Instead, an optimal policy must pull an arm with the highest mean reward, $\mu_t^\star = \max_{i \in [K]} \mu_t^i$, on each time step. More formally, the regret of a decision maker in nonstationary bandit problems is defined as follows:

$$R_T = \sum_{t=1}^{T} \mu_t^\star - \sum_{t=1}^{T} X_t^{i_t}. \tag{2.6}$$

If there is no relationship between the reward distributions for a given arm between time steps, then the decision maker cannot reliably predict the future rewards of each arm based on the feedback they have already received. As a result, additional structural assumptions on how the reward distributions evolve over time are required in order to achieve reasonable regret bounds. In fact, if no further structural assumptions are made on the reward distributions, then the nonstationary stochastic bandit problem is equivalent to the adversarial bandit problem. In the adversarial bandit problem, we assume that rewards are specified by a potentially malicious adversary on each time step, rather than being drawn from a distribution. For the adversarial bandit setting, Auer et al. (2002) introduce the Exp3 algorithm, which achieves $\mathcal{O}(\sqrt{TK\log(T)})$ expected weak regret under the assumption that rewards must lie in $[0, 1]$. In short, weak regret is the expected difference between the cumulative reward of the decision maker and the cumulative reward from the best policy among those that repeatedly play the same action:

$$\max_{i \in [K]} \sum_{t=1}^{T} X_k(t) - \mathbb{E}\left[X_t^{i_t}\right].$$

Moreover, Auer et al. (2002) show that Exp3 is optimal up to logarithmic factors when it comes to weak regret. Of course, we would like to identify policies which perform well with respect to the regret as defined in (2.6), rather than the weak regret.

A natural way to measure the complexity of a nonstationary MAB problem is to measure the number of time steps on which the reward distributions change. More formally, let L denote the number of time steps on which the reward distributions are updated:

$$L = \#\{1 \leq t \leq T \mid \exists i : \mu_{t-1}^i \neq \mu_t^i\}.$$

Note that if $L = T$, then we recover the entire class of nonstationary stochastic MAB problems. However, if L is small relative to T, one may hope to achieve vanishing regret. This motivates the development of sublinear regret bounds that depend on both L and T. The first bounds of this type were provided in the context of nonstochastic bandits by Auer et al. (2002), who introduce a variant of the Exp3

algorithm which obtains $\tilde{\mathcal{O}}(\sqrt{KLT})$ regret when the number of changes L is known. Meanwhile, variants of UCB have also been proposed that possess regret guarantees in terms of L and T. Kocsis and Szepesvári (2006) introduce Discounted-UCB, in which the empirical mean associated with each arm is replaced by a discounted average. Similarly, Garivier and Moulines (2011) introduce SW-UCB, which only computes the empirical mean reward associated with each arm in a sliding window that moves across the time horizon. More recently, Auer et al. (2019) proposed the ADSWITCH algorithm, which achieves $\tilde{\mathcal{O}}(\sqrt{KLT})$ regret without needing to know the number of changes L in advance.

Instead of considering the total number of changes, one can also consider the total variation of mean rewards over the entire time horizon. More formally, given a MAB problem instance, the total path variation of expected rewards (Besbes et al., 2014), is given by

$$\sum_{t=1}^{T-1} \max_{i \in [K]} |\mu_t^i - \mu_{t+1}^i|.$$

With this quantity in mind, Besbes et al. (2014) introduce a variant of Exp3, known as REXP3, which achieves $\tilde{\mathcal{O}}((KV_T)^{\frac{1}{3}} T^{\frac{2}{3}})$ regret on problem instances where the total path variation of expected rewards is bounded by V_T. Briefly, put REXP3 consists of splitting the time horizon into epochs of carefully chosen length, and running Exp3 within each epoch. Additionally, Besbes et al. (2014) provide a lower bound of order $\Omega((KV_T)^{\frac{1}{3}} T^{\frac{2}{3}})$ on the regret incurred by any algorithm on such a family of problem instances. As a result, REXP3 is essentially minimax optimal up to logarithmic factors. Karnin and Anava (2016) design an algorithm of similar structure, in which an ETC-style algorithm is run within epochs of adaptive length. Meanwhile, Wei et al. (2016) have developed a UCB-type algorithm with regret bounds which depend on the variation budget V_T and the statistical variance of each reward distribution.

Closely related to nonstationary stochastic MAB problems are restless bandits (Whittle, 1988). In a restless bandit problem, each action is associated with a Markov chain. Each time an action is taken by the decision maker, the Markov chain associated with each action makes a state transition. The decision maker only observes the current state of its chosen action. The reward, X_t^i, received by the decision maker for pulling arm i at time step t is a random variable which depends on the current state of the arm. We refer the interested reader to Ortner et al. (2012) for more details.

Observe that nonstationary stochastic MAB problems are natural models for sequential resource deployment settings where the utility of each resource may vary over time. In Chapter 4 we will leverage the concepts discussed in this section to develop our own model for sequential resource deployment. However, note that nonstationary bandit problems do not model resource unavailability. In the next section, we will discuss relevant constrained bandit models, which restrict when and how frequently a given action can be taken.

2.2.3 Constrained Bandit Problems

In many sequential decision problems, actions often become unavailable, depending on the state of the environment. This phenomena is modeled explicitly by sleeping bandits (Kleinberg et al., 2010), in which arms may ''fall asleep" and become periodically unavailable. Formally, the decision maker must choose their action at time step t from a set, $A_t \subseteq [K]$, of available actions. Observe that the original definition for regret, given in Section 2.2.1, is not suitable in the sleeping bandits setting, as the arm with highest mean reward may not always be available. As a result, many different notions of regret have been proposed in the literature as alternative performance measures for sleeping bandits.

For example, policy regret measures the difference in cumulative reward accrued by the decision maker and the best policy which deterministically maps a set of available actions to an available action. In contrast, the ranking regret measures the difference in cumulative reward by comparing against the best ranking of actions, which equates to a policy where the highest arm available in the ranking is pulled. Lastly, the per-action regret is the difference between the cumulative reward of a fixed action and the decision maker, only including rounds where the action was available. Note that policy regret upper bounds the ranking regret in all problem instances. For a thorough comparison of each regret notion, we refer the reader to Kale et al. (2016).

As with rewards, the set of available actions at each time step may be chosen by an adversary, or sampled stochastically from a stationary distribution. Kanade et al. (2009) develops an algorithm, based on follow-the-perturbed-leader, for sleeping bandits with stochastic action sets and adversarial rewards, which achieves $\tilde{\mathcal{O}}(K^{\frac{4}{5}}T^{\frac{4}{5}})$ ranking and policy regret. For both the case of adversarial action sets with stochastic rewards, and stochastic actions sets with stochastic rewards, Kleinberg et al. (2010) develop a UCB algorithm, called AUER, which incurs $\tilde{\mathcal{O}}(\sqrt{KT})$ ranking regret, and show that AUER is minimax optimal up to logarithmic factors.

For policy regret, Neu and Valko (2014) propose SLEEPINGCATBANDIT, a combinatorial semi-bandit algorithm, which incurs $\tilde{\mathcal{O}}(K^{\frac{2}{3}}T^{\frac{2}{3}})$ regret with stochastic action sets and adversarial rewards. Meanwhile, Saha et al. (2020) describe an extension of Exp3, which attains $\tilde{\mathcal{O}}(K^2\sqrt{T})$ policy regret under stochastic action sets and adversarial rewards when the probabilities of availability for each arm are independent. Saha et al. (2020) also provide a variant of their algorithm for the general case, where the probabilities of arm availability can be correlated, that incurs $\tilde{\mathcal{O}}(2^K\sqrt{T})$ policy regret but suffers from $\mathcal{O}(tK)$ time complexity on each round.

In the case, where both actions sets and rewards are adversarial, a variant of Exp4, and extension of the previously mentioned Exp3 algorithm for adversarial bandits, offers sublinear ranking regret, but is computationally inefficient. In fact, even in the online

learning setting, where the decision maker observes the rewards of all actions after each time step, achieving sublinear per-action regret is computationally intractable for all three notions of regret outlined (Kale et al., 2016; Kanade and Steinke, 2014).

In real world settings, deploying resources often comes with costs. For example, in an emergency response setting, deploying a search party may incur financial costs and may require the use of emergency vehicles, of which there are a limited number. As a result, decision makers often face budget requirements when making decisions. Knapsack bandits are designed to address these concerns. In a knapsack bandit each arm i has an associated cost, $c_i(t) \in \mathbb{R}^d$ at each time step. The decision maker is given a budget $B = (B_1, \ldots, B_d)$ which they must not exceed over the time horizon. That is, the decision maker must ensure:

$$\sum_{t=1}^{T} c_{i_t}(t) \leq B.$$

Knapsack bandits were first investigated under the name of budgeted bandits (Tran-Thanh et al., 2012; Ding et al., 2013). Tran-Thanh et al. (2012) study a one-dimensional budgeted setting in which the costs associated with each action are deterministic and fixed through time. In particular, Tran-Thanh et al. (2012) proposes the KUBE algorithm, which combines the density-ordered greedy algorithm for knapsack problems with a UCB algorithm to achieve asymptotically optimal regret of order $\mathcal{O}(\log(B))$ with respect to to the best policy in hindsight. Similarly, Ding et al. (2013) consider a one-dimensional setting, in which costs are drawn from a stationary distribution associated with each arm, and propose a UCB algorithm that achieves $\mathcal{O}(\log(B))$ regret compared to the policy which computes the optimal knapsack packing given the mean cost of each arm.

More recently, algorithms have been designed for the multidimensional setting. For a setting where costs are sampled from stationary distributions associated with each arm, Badanidiyuru et al. (2018) designs a primal-dual algorithm, based on upper confidence bounds, that achieves $\tilde{\mathcal{O}}(\sqrt{K\text{OPT}} + \text{OPT}\sqrt{K/B_{\min}})$[2] regret with respect to the optimal policy that knows both the cost and reward distributions. Contextual knapsack bandits are also well explored within the literature (Agrawal et al., 2016; Badanidiyuru et al., 2014). For example, Agrawal et al. (2016) consider a general contextual bandit setting in which rewards are passed to a concave function and costs are passed to a convex function, and develop a UCB algorithm which attains sublinear regret guarantees with respect to both cumulative reward and budget violation.

Moreover, adversarial knapsack bandits have also been investigated. For example, Rangi et al. (2019), consider a one-dimensional adversarial setting, designing a variant of Exp3 which achieves $\mathcal{O}(K^2 \log(K)/c_{\min}^3)$ regret, where $c_{\min}$ is the minimum cost of

[2]OPT denotes the cumulative reward of the optimal policy, and $B_{\min}$ denote the minimum budget across all dimensions

pulling an arm. Meanwhile, Immorlica et al. (2019), consider the multidimensional adversarial setting, and show that any policy must incur $\Omega(\log(T))$ regret with respect to the best distribution of actions.

Note that in many real world settings resources are reusable. For example, in the previously mentioned disaster response scenario, emergency vehicles can be reused after their first period of deployment is completed. In particular, the availability of a given resource typically depends on when it was last used or deployed. Neither sleeping or knapsack bandits explicitly model this phenomena (violating Requirement 2b). Next, we will introduce the stochastic blocking bandits setting (Basu et al., 2019), which explicitly models both the unavailability and reusability of resources.

The stochastic blocking bandit problem extends from the stationary stochastic MAB setting problem. However, unlike the classical stationary MAB problem, after an arm is pulled it is deterministically blocked, and cannot be pulled for the next $D_i - 1 \geq 0$ time steps. We refer to D_i as the blocking delay associated with arm i, and assume that blocking delays are known to the decision maker. Following our sleeping bandits notation, we let A_t denote the set of available arms at time step t. Note that the set of stochastic blocking bandit problems can be viewed as a subset of sleeping bandit problems, where actions sets are adversarial and rewards are stochastic. In other words, each stochastic blocking bandit problem is a sleeping bandit problem, where rewards are stochastic and action sets are chosen by an adaptive adversary who enforces blocking delays.

We say that a sequence of actions is admissible if an action is never taken when it is blocked. The goal of the decision maker is to select an admissible sequence of actions which maximises their cumulative reward. Just as in the classical stochastic MAB, to analyse the performance of the decision maker, we may compare against a meaningful benchmark policy. Clearly, in the stochastic blocking bandit setting, the policy which repeatedly pulls the arm with highest mean reward may not be admissible. As a result, the standard definition of regret, as given in Section 2.2.1, is not suitable. Instead, we may compare against the admissible sequence, $(i_1, \dots i_T)$, with highest expected reward:

$$\text{OPT} = \max_{(i_1,\dots,i_t)\in[K]^T} \sum_{t=1}^{T} \mu_{i_t}. \tag{2.7}$$

Note that this benchmark differs from those used in the sleeping bandits setting. In particular, this policy benchmark accounts for the fact that the availability of a given arm at a given time step depends on the previous actions of the decision maker. Meanwhile, the regret notions considered in the sleeping bandits literature do not account for how the subset of available actions at each time step might change if a different policy is adopted. Hence, standard algorithms for the sleeping bandits setting cannot be readily applied to blocking bandit problems.

Unfortunately, achieving good performance with respect to the benchmark described in Equation (2.7) is unrealistic. As proven by Basu et al. (2019), even with full information regarding the mean rewards of each arm, computing such an action sequence is NP-Hard if the random exponential time hypothesis is true. As a result, the decision maker cannot hope to have performance comparable to such a benchmark when the mean rewards of each arm are unknown. Instead, we should compare against a sequence which the decision maker could realistically produce with full information. This line of thought motivates the definition of α-regret.

Definition 2.6. For $\alpha \in (0, 1]$, the α-regret, R_T^α, of the decision maker is given by:

$$R_T^\alpha = \alpha \text{OPT} - \sum_{t=1}^{T} X_t^{i_t}$$

In other words, the α-regret compares the performance of the decision maker to an α-approximation of the best performing admissible action sequence in expectation. One way to construct algorithms with good α-regret is to first develop an α-approximation for the full information setting, then convert such an algorithm to the bandit setting. Greedy algorithms lend themselves particularly well to this approach for two reasons. First of all, greedy algorithms have been employed for many problem settings with similar combinatorial structure to blocking problems. Secondly, greedy algorithms combine naturally with both the ETC and UCB approaches outlined in Section 2.2.1.

In fact, Basu et al. (2019) apply this approach, first providing a greedy algorithm for the full information setting which simply pulls the arm with highest mean reward among those available at the current time step, and prove that such an algorithm is a $(1 - 1/e)$-approximation of the optimal admissible policy. Basu et al. (2019) then combine this algorithm with UCB. The resulting algorithm greedily pulls the arm with the highest confidence index among those available, providing an instance-dependent regret bound of $\mathcal{O}(\log(T))$.

Several extensions to the stochastic blocking bandits setting have been considered in the literature. Basu et al. (2021a) investigate a contextual bandit setting with fixed and known delays associated with each arm, providing an instance dependent α-regret guarantee of order $\mathcal{O}(\log(T))$, combining an online randomised rounding algorithm with UCB. Atsidakou et al. (2021) study a combinatorial bandit setting in which blocking delays are stochastic. More specifically, the authors consider a setting in which multiple arms can be pulled in each round, subject to feasibility constraints. Within this setting, the authors develop a UCB bandit algorithm based on a greedy heuristic for the full information setting with an instance-dependent approximate regret guarantee of order $\mathcal{O}(\log(T))$. Similarly, Papadigenopoulos and Caramanis (2021) studies a setting in which multiple arms can be pulled in one round subject to

matroid constraints, combining an offline greedy algorithm with UCB to obtain a worst-case regret bound of order $\mathcal{O}(\sqrt{T})$.

In Chapter 4, we will introduce an adversarial blocking bandit problem, designed to model the deployment of reusable resources, whose utility may vary over time. More specifically, we will combine concepts from both stochastic blocking bandits and nonstationary bandits. Following the approach of Basu et al. (2021a), we first develop an approximation algorithm for the full information setting, before applying the techniques of Besbes et al. (2014) to convert this approximation algorithm to our bandit setting.

Aside from those already mentioned, many other constrained bandit problems have been studied in the literature. For example, Chakrabarti et al. (2008) introduce the mortal bandits setting, in which arms have a limited lifetime and may become permanently unavailable either when pulled, or as a result of time passing. In particular, Chakrabarti et al. (2008) characterize the maximum reward achievable by any policy in expectation and present several heuristic algorithms which are optimal when the rewards associated with each arm are deterministic. Meanwhile, combinatorial bandits (Kveton et al., 2015; Combes et al., 2015; Chen et al., 2013), in which a subset of arms can be pulled on every round subject to feasibility constraints, have been studied extensively. As mentioned by Basu et al. (2019) stochastic blocking bandit problems can be formulated as combinatorial bandit problems, by grouping time steps into blocks which scale exponentially with the least common multiple of blocking delays. Of course, this means that combinatorial bandit algorithms are often not practical in the blocking bandit settings, as the number of action sets that need to be considered within each block may be exponential in the least common multiple of blocking delays, which could be very large.

2.3 Repeated Matching with Reusable Resources

Finally, we review research literature relevant to Problem Domain 3. We first introduce and survey the traditional one-sided matching problem, which serves as a basis for the setting we introduce in Chapter 5. Whilst doing so, we compare two popular and natural algorithms, the probabilistic serial (PS) mechanism, and random serial dictatorship (RSD). As discussed in the introduction, matching typically takes place repeatedly over many time steps in practical settings. Therefore, in Section 2.3.2, we shift our focus to repeated matching settings. In particular, we focus on repeated matching settings where agents are initially unaware of their preferences and must learn them over time, seeking a resolution to Requirement 3c. Whilst many such settings have been proposed in the literature, none model resource availability and reuse (violating Requirements 3a and 3b). This observation motivates our

development of the sequential blocked matching setting, which we introduce in Chapter 5.

2.3.1 One-Sided Matching

To start, we first review the traditional one-sided matching problem and its variations. One-sided matching models a wide range of real world problems, including the matching of college courses to students and organs to medical patients. For example, one-sided matching is also referred to as the capacitated house allocation problem, due to its early use in modelling the allocation of social housing (Hylland and Zeckhauser, 1979). More generally, one-sided matching problems model any scenario in which a set of indivisible goods or services must be assigned amongst a set of agents or individuals, who may prefer to be assigned some goods/services over others.

In a one-sided matching problem, a central planner is tasked with producing a (potentially randomised) matching, m, between a set of n agents and a set of s indivisible units. From now on, we will use the terms unit and service interchangeably. In any given matching, each agent can be assigned at most one service, and each service can be assigned to at most one agent. In other words, each agent is unit-demand. Additionally, it is assumed that agents hold private preferences over services, and thus, may prefer to be assigned one service over another. Note that the preferences of each agent are private, and initially unbeknownst to the central planner.

We say that an agent i holds ordinal preferences if their preferences can be fully described by a linear ordering, $\succ_i$, over services. The higher a service is in the linear ordering, the more it is preferred by the agent. Meanwhile we say that an agent i holds cardinal preferences if their preference for a given service j is described by a positive real number, μ_{ij}, known as a utility value. Under cardinal preferences, agent i prefers one service over another if and only if it has higher utility value. Note that any set of cardinal preferences induces a set of corresponding ordinal preferences.

To aid in the selection of an appropriate matching, each agent submits a report of their own preferences to the central planner. Agents may report cardinal or ordinal information, depending on context. We say that an agent is truthful if they submit a report that is consistent with their underlying preferences. Otherwise, we say that the agent has misreported. It is assumed that agents are rational, and will misreport their preferences if it improves the service they are assigned.

Typically, the goal of the central planner is to adopt a matching policy which is both fair and efficient. For example, consider the problem faced by a server administrator for a cloud computing company. The server administrator is tasked with assigning computing hardware to clients, who each have a job they need to complete. The

administrator wants to ensure that their assignment of hardware is efficient and minimises the cumulative run time of all jobs, whilst also ensuring that no customers are treated unfairly, and given subpar hardware compared to clients with similar job specifications.

Many different notions of fairness and efficiency have been proposed in the literature. When agents have ordinal preferences, it is natural to formulate efficiency via first-order stochastic dominance (Bogomolnaia and Moulin, 2001). More formally, let m_i denote the randomised assignment of units to agent i by a randomised matching m. Furthermore, let m_{ij} denote the probability agent i is matched to service j. Similarly, given a preference ordering $\succ_i$, let

$$\omega(\succ_i, j, m_i) = \sum_{k:k\succ_i j} m_{i,k}$$

denote the probability that the randomised matching policy employed by the central planner assigns service j, or better, to agent i. We say that a randomised matching m stochastically dominates a randomised matching m' for agent i if:

$$\omega(\succ_i, j, m_i) \geq \omega(\succ_i, j, m'_i).$$

We say that the randomised matching m is stochastic dominance efficient (or sd-efficient for short) if m is not stochastically dominated by any randomised matching for all agents. Instead of aiming for sd-efficiency, a central planner may aim for the weaker requirement of ex post efficiency. A matching is ex post efficient if it is Pareto optimal, meaning that there exists no matching in which all agents are better off.

A standard way of modelling fairness is via envy-free conditions, which stipulate that no agent should prefer the assignment given to another agent over their own. Like efficiency, envy-freeness has a natural formulation via first-order stochastic dominance. More specifically, we say that a matching is sd-envy-free if, no agent's is assignment is stochastically dominated (according to their own preferences) by another agent's assignment.

Aside from fairness and efficiency, a central planner may also hope to adopt a matching policy which is truthful, or strategyproof. Truthfulness ensures that honest agents are rewarded, which is desirable from an ethical standpoint. Moreover, the efficiency and fairness of a given matching policy is typically examined under the theoretical assumption that agents are truthful. As a result, truthfulness is an important property from a performance standpoint, which ensures theoretical results regarding efficiency and fairness reflect reality. Once again, truthfulness can be formulated via first-order stochastic dominance. Briefly put, a matching policy is sd-strategyproof if the assignment each agent receives when they report truthfully

stochastically dominates any assignment they could receive by misreporting, with respect to their own preferences.

With fairness, efficiency and truthfulness in mind, Bogomolnaia and Moulin (2001) propose the probabilistic serial (PS) mechanism. The PS mechanism is always sd-efficient, sd-envy-free, and (weakly) sd-strategyproof. The PS mechanism first computes a fractional allocation of services over multiple phases, which is then converted into a distribution over integral matchings that the central planner samples from. In short, the PS mechanism constructs a fractional allocation by allowing each agent to "eat" their most preferred service at a constant rate until the service has been entirely consumed. Once a service has been completely consumed, the first phase ends. After this, the agents continue eating, selecting their most preferred service out of those still available.

Aside from the PS mechanism, the random serial dictatorship (RSD) algorithm (Abdulkadiroglu and Sonmez, 1998) is a popular one-sided matching algorithm due to its simplicity. In short, RSD proceeds by first uniformly sampling a permutation, σ, over agents. Following the order of the sampled permutation, each agent is assigned their most preferred service out of those that remain. Unfortunately, RSD is not sd-efficient and only satisfies a weakened version of sd-envy-freeness. However, RSD is both ex post efficient and sd-strategyproof. In general, RSD and the PS mechanism are not directly comparable, in that neither policy stochastically dominates the other in general. However RSD and the PS mechanism converge to the same limit as the supply (i.e. number of copies) of each service goes to infinity (Che and Kojima, 2010). We refer the interested reader to Hosseini et al. (2018), who evaluate and compare the empirical performance of both policies via several experiments.

So far, we have only performance measures and benchmarks that consider ordinal information. When preferences are cardinal, it is sensible to dispense with properties defined in terms of stochastic dominance and study properties or benchmarks that explicitly depend on the utility values each agent assigns to each service. For example, a central planner may want to select a matching m which maximises social welfare:

$$\mathrm{SW}(m) = \sum_{i=1}^{n} \mu_{i,m(i)}$$

where $m(i)$ denotes the service assigned to agent i in the matching m. However, when cardinal preferences are reported ordinally, the central mechanism cannot hope to select a matching (randomised or not) that maximises social welfare. This is because there may be many utility profiles which correspond to a single preference profile. Instead, the central planner can only hope to minimise distortion, which corresponds to the worst-case social welfare of a matching given the preference profiles submitted. Distortion was first studied by Procaccia and Rosenschein (2006) in the context of

computational social choice, and characterizes the unavoidable penalty that comes with ordinal reporting of cardinal preferences.

When cardinal preferences are unit-sum or unit-range, Filos-Ratsikas et al. (2014) show that RSD achieves $\mathcal{O}(\sqrt{n})$ distortion, and that no matching policy is asymptotically better. Similarly, Adamczyk et al. (2014) show that RSD is a 3-approximation for dichotomous preferences, and that RSD is asymptotically optimal in settings where preferences are normalized to lie in the interval $[0, 1]$. Likewise, Christodoulou et al. (2016) show that both RSD and PS exhibit $\mathcal{O}(\sqrt{n})$ price of anarchy, and show that any deterministic matching policy achieves price of anarchy of order $\Omega(n^2)$.

In contrast to RSD, the PS mechanism is not truthful when agents have cardinal preferences. That is, an agent may be able to improve their expected utility under the PS mechanism by misreporting thier preferences. Whilst, not completely truthful, the PS mechanism does satisfy a relaxed version of strategyproofness know as incentive ratio. More specifically, the PS mechanism has a $3/2$ incentive ratio (Wang et al., 2020), which means that no agent can improve their own utility by a multiplicative factor greater than $3/2$ by misreporting. In addition, Kojima and Manea (2010) show that the PS mechanism is (weakly) truthful when the supply of each service is sufficiently large.

Note that social welfare is not a fair benchmark. This is because policies which assign low utilities to one agent and large utilities to another agent can attain the same social welfare as a policy which assigns roughly the same utility to everyone. As a result, much of the matching literature has focused on alternate benchmarks that better trade-off fairness with efficiency, especially for matching settings in which preferences are reported cardinally. For instance, instead of maximising social welfare, a central planner may instead estimate the Nash bargaining solution (Nash Jr, 1950), which corresponds to the product of each agents' gain in expected utility relative to a baseline matching. Note that, after subtracting the utility of the baseline matching from each agent's utilities, finding the Nash bargaining solution is equivalent to maximising the Nash social welfare (Caragiannis et al., 2019). Any randomised matching which maximises the NSW is proportionally fair. That is, no alternative matching can improve the welfare of one agent by a multiplicative factor, without reducing the utility of another agent by a greater factor.

Moreover, in general assignment problems without unit-demand constraints, assignments that maximise Nash social welfare are envy-free up to one service and serve as good approximations of other fairness properties, such as the maximin share guarantee (Caragiannis et al., 2019). In the general assignment setting with divisible services, Cole et al. (2013) devise the partial allocation (PA) algorithm which is truthful and ensures each agent receives a $1/e$-approximation of its welfare under the NSW-maximising allocation. Unfortunately, to guarantee truthfulness, the PA

mechanism may leave some services unassigned. In the case of one-sided matching, Abebe et al. (2020) address this issue by proposing the randomised partial improvement (RPI) mechanism, which returns a $\mathcal{O}(2^{2\sqrt{\log(n)}})$ per-agent approximation of the Nash bargaining solution and ensures that all services are allocated.

Competitive equilibrium from equal incomes (CEEI) (Hylland and Zeckhauser, 1979) is an alternative solution concept which also aims to trade-off fairness with efficiency. CEEI models the allocation of services via the following the hypothetical process. First, each agent receives a single unit of fiat currency. Then, each agent is permitted to buy fractions of each service using their fiat currency. A CEEI is simply the allocation of services at a competitive equilibrium in the resulting market. Any allocation which is a CEEI is envy-free in the sense that no agent prefers the allocation prescribed to another agent over its own. Unfortunately, CEEI is not a strategyproof solution concept. That is, agents may be incentivised to misreport their preferences in order to alter the competitive equilibrium of the induced market and achieve a better payoff. However, CEEI is strategyproof in the large. In other words, given enough agents with the same preferences, computing a CEEI constitutes a truthful algorithm. Note that with no capacity constraints on agents, computing CEEI corresponds to finding a randomised assignment which maximises NSW. However, under matching constraints, CEEI and NSW are different solution concepts. Unfortunately, the problem of computing an exact CEEI is not in PPAD, and the problem of approximating a CEEI is PPAD-hard (Chen et al., 2022). In order to circumvent this issue, Alaei et al. (2017) study a more tractable setting, and provide a polynomial time algorithm for computing CEEI in matching settings when the number of unique preference orderings is constant.

Generally speaking, the choice of underlying preference structure and reporting scheme for a one-sided matching setting depends on the real world problem being modeled. Note that many large-scale decision-making problems involve humans in the loop. For example, in the case of freelance employment, the suitability of a given contractor is typically determined by an employee of the contracting company. In many practical contexts, humans can find it hard to ascribe an exact numerical value which expresses how much they value a given service. In contrast, humans often find it easy to select the service they prefer from a selection. Such an observation motivates a focus on one-sided matching settings in which preferences are reported ordinally. In addition, ordinal reporting is more efficient than cardinal reporting in the sense of communication bandwidth. Moreover, matching policies which rely on ordinal preference reports can be trivially extended to settings with cardinal reporting. With these concerns in mind, the repeated matching setting we curate in Chapter 5 assumes that agents have cardinal preferences but report ordinally.

Note that the motivating examples considered in Section 1.3 take place repeatedly over many time steps. For example, consider the problem of assigning freelance contractors to employers. Companies may will eventually have new contracts that need to be fulfilled and as a result must be reassigned freelance contractors in the future. Similarly, consider the cloud computing problem discussed in Chapter 1. After a certain amount of time, clients are likely to return with similar tasks that need to be completed in the future. As a result, a server administrator will be required to produce multiple matchings through time as new jobs are submitted. In addition, agents are often initially unaware of their preferences, and must learn them over time through repeatedly matching with services. For instance, a company may need to hire with a freelance contractor multiple times before they decide if they are a good fit for a company. Similarly, a client may need to run multiple jobs on a single piece of server hardware before they have a good understanding of its performance and cost effectiveness. Therefore, the matching setting we propose in Chapter 5 is repeated, in order to capture the phenomena described above.

2.3.2 Repeated Matching Problems

A natural way to design algorithms for repeated matching settings is to directly apply algorithms for one-sided matching on each time step. However, if the preferences of individual agents change over time many desirable properties of established matching algorithms do not carry over to the sequential setting. For example, Hosseini et al. (2015) investigate the performance of RSD in a repeated ordinal matching setting, where the preferences of each agent evolve stochastically depending on the matchings selected by the central planner, and introduce a sequential version of sd-strategyproofness, known as global sd-strategyproofness, or gsd-strategyproofness for short. In particular, the authors show that RSD is not gsd-strategyproof and propose a modified version of RSD, which is sd-efficient and gsd-strategyproof. A similar model is studied in a social choice context by (Parkes and Procaccia, 2013), who model a sequential social choice problem via a Markov decision process and show that optimal policies, satisfying a range of axiomatic properties, can be computed in polynomial time when the number of agent types is constant.

Note that in both the settings discussed above, each agent is fully aware of their own preferences at the start of each time step. Therefore, these settings are insufficient when it comes to modelling agents that must learn their preferences over time (Requirement 2c). In contrast, within the multi-agent multi-armed bandit community, many repeated matching settings in which agents are initially unaware of their preferences have been proposed since the initial work of Das and Kamenica (2005). For example, Liu et al. (2020) consider a two-sided sequential matching setting in which each agent must learn about their preferences via bandit feedback. The authors

first propose an ETC algorithm which achieves sublinear regret with respect to the optimistic Gale-Shapley matching (Gale and Shapley, 1962). More specifically, the ETC algorithm proposed consists of assigning services in a round robin fashion for a fixed number of rounds, so that each agent may learn their preferences. After this initial exploration phase, the central planner simply computes the optimistic Gale-Shapley matching on each time step assuming that each agent will submit the preference ordering induced by their empirical mean rewards. In addition, Liu et al. (2020) propose a UCB algorithm which achieves sublinear regret with respect to the pessimistic Gale-Shapley matching.

As the proposed ETC algorithm gives agents no say over their allocated service in the exploration phase, and performs Gale-Shapley matching repeatedly in the exploitation phase, it inherits all the truthfulness properties of the Gale-Shapley algorithm. The situation is slightly more unclear in the case of the UCB algorithm. However, the authors show that no agent can improve their regret with respect to the optimistic Gale-Shapley matching by deviating from the preference ordering induced by their upper confidence indexes for each service. Cen and Shah (2022) investigate a similar setting to that of Liu et al. (2020), but introduce monetary costs and transfers. More specifically, it is shown that there exist transfer and cost rules which ensure repeated application of the Gale-Shapley algorithm results in $\mathcal{O}(\log(T))$ regret for each agent with respect to the optimistic Gale-Shapley matching.

Closely related are decentralised multi-agent bandit settings. In decentralised settings, agents are ranked, with the first agent in the ranking having priority over others. If an agent chooses an arm selected by a higher ranked agent, then it is blocked and receives zero reward. As a result, each agent must be pragmatic, and only pulls arm which are not desired by higher ranking agents and must communicate effectively through their actions to prevent avoidable collisions. Sankararaman et al. (2021) develop a variant of UCB for this setting, referred to as UCB-D3. In particular, it is shown that each agent may only benefit by a small additive factor if they deviate from UCB-D3. Chawla et al. (2020) investigate a more general version of the setting introduced by Sankararaman et al. (2021), in which agents are able to pass messages to each other between pulls, whilst obeying a communication budget, and develop a UCB style algorithm with sublinear regret guarantees. Meanwhile, Liu et al. (2021) consider a two-sided matching setting where the ranking of agents is unique to each arm, and develop a decentralised variant of UCB which achieves sublinear regret with respect to the pessimistic Gale-Shapley matching and requires no explicit communication between agents. Basu et al. (2021b) propose an improved algorithm for this setting, extending from UCB-D3, and achieve logarithmic regret with respect to the optimistic Gale-Shapley matching.

Note that all the motivating examples discussed with respect to Problem Domain 3 in Chapter 1 involve a central planner, who has access to the reports of each agent. For

example, the allocation of computer hardware to clients in a cloud computing setting may be performed by a server administrator who knows the needs of all clients. In the case of contract work, matching is often performed by an intermediary who plays the role of a recruiter. As a result, the sequential matching setting we propose in Chapter 5 is centralised, and does not suffer from issues of collision and communication present in decentralised models.

Lastly, there has been a long line of work studying online versions of weighted bipartite matching and stable matching where matching entities (i.e. agents and services) arrive dynamically over time (Karp et al., 1990; Kalyanasundaram and Pruhs, 1993; Karande et al., 2011; Khuller et al., 1994). Within these settings, agents and services often need to be matched quickly upon arrival, and are no longer relevant once matched. As a result, such settings often do not capture the learning experience of individuals and organizations when interacting with matching platforms, and hence are generally unsuitable for addressing Requirement 3c.

In reality, agents and services often persist through time and need to be matched repeatedly. For example, a freelancer may maintain a relationship with a specific recruiter over the course of many contracts. Likewise, a company may return to a recruiter as new openings need to filled. With this concern in mind, the matching setting we develop Chapter 5 is similar in vein to the bandit models discussed above, wherein a central planner is tasked with selecting a matching between persistent agents and services on each time step, instead of matching agents and services on the fly as they arrive, in order to construct a single matching over the entire time horizon.

Note that none of the settings discussed in this section model resource availability. Similar to Problem Domain 2, services often become unavailable for a period when matched. Moreover, the length of time a service is unavailable often depends on the agent it was assigned to. For example, once a piece of cloud computing hardware is assigned to a particular client, it is unavailable to others until the client's job is complete. Such use cases motivate the development of repeated matching settings where services can be blocked when assigned (Requirements 3a and 3b).

With this concern in mind, the sequential matching setting we propose in Chapter 5 explicitly models the blocking of matched services. As in repeated matching settings with dynamically evolving preferences, we will see that trivial applications of standard matching algorithms, such as RSD, do not suffice. Moreover, we will see that standard performance benchmarks, such as policies which repeatedly select the same matching on each time step cease to be reasonable measures of performance. As services may become blocked when assigned, the central planner may not be able to construct the same matching on every time step. In other words, the central planner must instead adopt a performance benchmark which accounts for the blocking of services. Hence, attaining performance guaranatees in terms of efficiency

(Requirement 3a) and truthfulness (Requirement 3b) presents a significant technical challenge in the presence of blocking.

Chapter 3

Stackelberg Prediction Games for Linear Regression

In this chapter, we present a learning framework to address Problem Domain 1. More specifically, we consider a Stackelberg prediction game (SPG) setting for linear regression. We formally introduce our framework in Section 3.1. Whilst doing so, we illustrate our framework's ability to model a wide range of agent preferences, with the goal of addressing Requirement 1a. Then, we investigate a natural approach to learning called Stackelberg empirical risk minimisation, which, as the name suggests, serves as an analog to empirical risk minimisation in the classical supervised machine learning setting. We first analyse Stackelberg empirical risk minimisation from an optimisation perspective, with a particular focus on the subclass of SPGs where both agents and the learner adopt a square loss function. Under this assumption, we develop a polynomial time algorithm for Stackelberg empirical risk minimisation, called `SDP-BISECT`, partly addressing Requirement 1b. Sections 3.3 through 3.6 are dedicated to its analysis. After briefly discussing extensions of `SDP-BISECT` based on kernel methods (Section 3.7), we showcase the empirical performance of `SDP-BISECT` on several benchmark datasets (Section 3.8). Following this, we consider Stackelberg empirical risk minimisation from a statistical perspective, identifying when it is possible to obtain meaningful generalisation guarantees (addressing Requirement 1c). Following this, we summarise the semidefinite programming reformulation of STERM, which is both heavily inspired by and improves upon `SDP-BISECT` (Section 3.10). Then, in Section 3.11, we discuss how γ, a hyperparameter in our model that characterises the manipulation power of each agent, may be selected in practice. Before concluding, we discuss some related open problems in Section 3.12.

3.1 Model

To begin, we formally describe the problem setting which will form the basis of this chapter. We consider a Stackelberg prediction game (Brückner and Scheffer, 2011), in which a learner must select a linear hypothesis $\mathbf{w} \in \mathbb{R}^n$ for the purpose of regression. It is assumed that data of interest is sampled from a fixed data distribution $\mathcal{D}$, which the learner does not have access to. Note that so far, the assumptions introduced are standard and align with the classical PAC learning setting discussed in Section 2.1.1. It is from this point onwards that we diverge from the classical PAC learning setting.

We assume that data points are sampled from $\mathcal{D}$ by data providing agents, who may have their own incentives and goals. More formally, we assume that each data point sampled from $\mathcal{D}$ is of the form $(\mathbf{x}, y, z)$, where $\mathbf{x} \in \mathbf{R}^n$ is an input example, $y \in \mathbb{R}$ is the corresponding label of interest to the learner, and $z \in \mathbb{R}$ is a label indicating the preferences of the agent who produced the sample. For example, consider the insurance quotation problem discussed at numerous points throughout this thesis. A single data point may correspond to a customer (the agent). In this case, $\mathbf{x}$ may correspond to personal statistics describing the customer, y may correspond to an insurance quote which is financially optimal from the point of view of the insurer (the learner), whilst z may describe the insurance quote desired by the customer.

Additionally, we assume that the learner has access to a training dataset, $S = \{(\mathbf{x}_i, y_i, z_i)\}_{i=1}^m$, consisting of input examples, and their corresponding labels, sampled directly from $\mathcal{D}$. In practice, such a dataset may be obtained through a costly verification process. For example, an insurance company may audit a small portion of its customer base in order to validate the personal information they have submitted. Moreover, the insurance company may survey customers to identify their preferred labellings. Whilst such verification processes may be too expensive to apply to every single data point, the learner may be able to construct a small, verified dataset for the purpose of training.

After the learner has selected a hypothesis, agents continue to produce input examples from the distribution $\mathcal{D}$. With full knowledge of the linear predictor chosen by the learner, each agent is allowed to modify the features of each input example. However, each agent pays a cost, $c(\mathbf{x}, \tilde{\mathbf{x}})$, for modifying an input example $\mathbf{x}$ into $\tilde{\mathbf{x}}$. The cost function, $c : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_+$, reflects the effort invested by the agent to perform such a modification. For example, in the insurance setting, such a cost function may correspond to the cost of producing falsified documentation or the legal risk associated with lying. The goal of each agent is to manipulate the learner into predicting as close to the agent's preferred labelling as possible, whilst also ensuring the cost of manipulation is low.

More formally, given the sample $(\mathbf{x}, y, z)$ the goal of an agent is to solve the following optimisation problem:

$$\mathbf{x}^\star \in \arg\min_{\tilde{\mathbf{x}}} \ \ell_{+1}(\mathbf{w}^\top \tilde{\mathbf{x}}, z) + \gamma c\left(\mathbf{x}, \tilde{\mathbf{x}}\right)$$

where $\mathbf{w}$ is the linear predictor chosen by the learner, $\ell_{+1} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ is a loss function measuring the accuracy of the labelling output by the learner compared to the target label z, and $\gamma > 0$ is scalar characterising the trade-off each agent makes between producing low cost modifications, and bringing the learner's prediction as close to z as possible.

In contrast, the goal of the learner is to select a risk minimising predictor under the assumption that agents will submit optimal modifications:

$$\mathbf{w}^\star \in \arg\min_{\mathbf{w}} \mathbb{E}\left[\ell_{-1}(\mathbf{w}^\top \mathbf{x}^\star, y)\right] \tag{3.1}$$

where ℓ_{-1} is the loss function of the learner and expectation is taken with respect to the distribution $\mathcal{D}$. In other words, the learner is interested in reliably predicting the target label y, under the assumption that agents will try to manipulate them into predicting z. From now on, we refer to the optimisation objective in Problem (3.1) as the Stackelberg risk, due to its correspondence to the standard risk examined in classical supervised learning settings.

Observe that Problem (3.1) is not well-defined, as a given agent may have multiple best-response modifications to a learner's chosen predictor. Hence, from now on, we assume for any given sample $(\mathbf{x}, y, z)$ and any linear predictor $\mathbf{w}$, that $\mathbf{x}^\star$ is unique. This avoids the need to distinguish between optimistic and pessimistic versions of Problem (3.1) from the bilevel optimisation perspective. All the special cases of Problem (3.1) we examine in this chapter satisfy this assumption. More generally, whenever both the loss and cost functions of the data providers are convex, strict convexity of either function is sufficient to guarantee that each data provider has a unique best-response.

Note that, by changing the label z associated with a given data point, a wide variety of agent preferences can be expressed. For example, perhaps every agent is interested in attaining a specific score $s \in \mathbb{R}$, in the same way that all agents are interested in attaining a specific classification in strategic classification. This can be easily modeled by setting $z = s$ across the entire distribution $\mathcal{D}$. Similarly, by varying γ, the learner can account for agents of different capability. In other words, by decreasing γ, the learner assumes that agents are more powerful and can produce a given modification at lower cost. On the other hand, by increasing γ, the learner assumes that agents are less powerful, and pay a higher cost for producing a given modification. As a result, the model proposed is highly flexible, and addresses Requirement 1a.

Of course, the learner cannot hope to solve Problem (3.1) directly. However, the learner can use the training dataset to minimise an empirical version of their risk:

$$\begin{aligned} \arg\min_{\mathbf{w}\in\mathbb{R}^n} \quad & \frac{1}{m}\sum_{i=1}^{m} \ell_{-1}(\mathbf{w}^\top \mathbf{x}_i^\star, y_i) \\ \text{s.t.} \quad & \mathbf{x}_i^\star \in \arg\min_{\tilde{\mathbf{x}}_i} \ \ell_{+1}(\mathbf{w}^\top \tilde{\mathbf{x}}_i, z_i) + \gamma c\left(\mathbf{x}_i, \tilde{\mathbf{x}}_i\right) \quad \forall i \in [m]. \end{aligned} \tag{3.2}$$

We refer to Problem (3.2) as Stackelberg empirical risk minimisation (STERM), due to its clear connections with ERM in the classical supervised learning setting. The remainder of this chapter is dedicated to investigating STERM from both an optimisation and statistical perspective. From the optimisation standpoint, it is important that STERM can be solved, or approximated, efficiently (Requirement 1b). From the statistical standpoint, it is important that the empirical performance of a linear predictor reflects its Stackelberg risk as the size of the training dataset increases (Requirement 1c).

3.2 Square Losses

We now consider a special case of model above in which both agents and the learner adopt a square loss function. Additionally, we assume that the cost function associated with each agent is the squared Euclidean distance. More formally, it is assumed that:

$$\begin{aligned} \ell_{-1}(\tilde{y}, y) &= (\tilde{y} - y)^2 \\ \ell_{+1}(\tilde{z}, z) &= (\tilde{z} - z)^2 \\ c(\mathbf{x}, \tilde{\mathbf{x}}) &= \|\mathbf{x} - \tilde{\mathbf{x}}\|^2. \end{aligned}$$

Note that this special case is of interest for several reasons. First of all, the square loss is perhaps the most popular loss function for regression problems, with its use motivated by classical results such as the Gauss-Markov Theorem. Secondly, square loss functions typically have special properties from an optimisation perspective, often yielding closed form solutions. As a result, one may expect to develop specialised algorithms for SPGs involving square losses. Moreover, the cost function adopted can be interpreted as a form of Tikhonov regularisation, limiting the complexity of the modification submitted by each agent. By decreasing γ, the learner assumes that each agent i may submit more complex modifications that may stray further from the originally sampled input example in terms of Euclidean distance.

Substituting the aforementioned functions into Problem (3.2) yields the following bilevel optimisation problem:

$$\begin{aligned} \arg\min_{\mathbf{w}\in\mathbb{R}^n} \quad & \frac{1}{m}\sum_{i=1}^{m}(\mathbf{w}^\top \mathbf{x}_i^\star - y_i)^2 \\ \text{s.t.} \quad & \mathbf{x}_i^\star \in \arg\min_{\tilde{\mathbf{x}}_i} (\mathbf{w}^\top \tilde{\mathbf{x}}_i - z_i)^2 + \gamma\|\mathbf{x}_i - \tilde{\mathbf{x}}_i\|^2 \quad \forall i \in [m]. \end{aligned} \tag{3.3}$$

As is standard, we may construct a matrix, $X \in \mathbb{R}^{m\times n}$, whose ith row corresponds to input example $\mathbf{x}_i$ in the sample S. Likewise, we may construct the vectors $\mathbf{y}$ and $\mathbf{z}$, whose ith elements correspond to output labels y_i and z_i respectively. This allows us to reformulate Problem (3.3) as follows:

$$\begin{aligned} \arg\min_{\mathbf{w}\in\mathbb{R}^n} \quad & \|X^\star\mathbf{w} - \mathbf{y}\|^2 \\ \text{s.t.} \quad & X^\star = \arg\min_{\tilde{X}\in\mathbb{R}^{m\times n}} \left\|\tilde{X}\mathbf{w} - \mathbf{z}\right\|^2 + \gamma\left\|\tilde{X} - X\right\|_F^2 \end{aligned} \tag{3.4}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. In the sections that follow, we will study Problem (3.4) in detail and present a polynomial time algorithm that converges to global optima. Note that this is a marked improvement upon the generic algorithm for SPGs proposed by Brückner and Scheffer (2011), which only guarantees convergence to stationary points. In other words, Problem (3.4) represents a tractable special case, in which it is possible to identify globally optimal hypotheses, rather than hypotheses which are only locally optimal.

Before introducing our algorithm formally, we first give a brief overview of how our algorithm is derived and the theoretical tools employed. To begin, using standard results in linear algebra, we first reformulate Problem (3.4) into a fractional programming problem (Lemma 3.1). After this, we show that such fractional programming problems can be solved via bisection search (Section 3.4), by leveraging the classical results of Dinkelbach (1967). Unfortunately, each step of this bisection search involves solving a nonlinear optimisation problem. We show that this subproblem can be reformulated as a semidefinite program (SDP) via a special application of the S-lemma, an equivalence theorem which states when one quadratic inequality is the consequence of another set of quadratic inequalities. Summarising, the algorithm we propose solves a fractional program equivalent to Problem (3.4) via bisection search, wherein each step consists of solving an SDP.

3.3 Problem Reformulation

To begin our analysis, we first reformulate Problem (3.4) as a fractional programming problem. In doing so, we obtain the following intermediate form of Problem (3.4).

Lemma 3.1. *Problem* (3.4) *is equivalent to the following nonlinear optimisation problem:*

$$\arg\min_{\mathbf{w}\in\mathbb{R}^n} \quad \left\| \frac{\frac{1}{\gamma}\mathbf{z}\|\mathbf{w}\|^2 + X\mathbf{w}}{1+\frac{1}{\gamma}\|\mathbf{w}\|^2} - \mathbf{y} \right\|^2 \tag{3.5}$$

Proof. Observe that the lower-level minimisation problem is unconstrained and strictly convex. As a result, by Fermat's Theorem, the lower-level minimisation problem can be solved by finding a point at which the gradient with respect to $\tilde{X}$ is zero. This yields a closed form solution for $X^\star$ which can be used to rewrite the constraint in Problem (3.4) as follows:

$$\begin{aligned} \arg\min_{\mathbf{w}\in\mathbb{R}^n} \quad & \|X^\star\mathbf{w} - \mathbf{y}\|^2 \\ \text{s.t.} \quad & X^\star = (\mathbf{z}\mathbf{w}^T + \gamma X)(\mathbf{w}\mathbf{w}^T + \gamma I)^{-1} \end{aligned}$$

Substituting the right-hand side of the constraint directly into the objective and applying the Sherman-Morrison formula (Boyd and Vandenberghe, 2004) to the matrix $\mathbf{w}\mathbf{w}^T + \gamma I$ achieves the desired form. □

Note that Problem (3.5) clearly illustrates the effect of each agent's actions on the predictions of the learner. In particular, it is clear that the prediction made by the learner on a given data point is a convex combination of the agent's preferred labelling, $z \in \mathbb{R}$, and the prediction the learner would have made if given the true input example without modification from the agent. Moreover, the weighting of this convex combination is controlled by the Euclidean norm of the predictor chosen and γ. As the learner chooses a predictor larger in Euclidean norm, its predictions will slowly be pushed towards the labels desired by each agent. Similarly, as γ grows, the closer the prediction of the learner gets to their prediction on the unmodified input.

By moving $\mathbf{y}$ into the fraction, we can express Problem (3.5) as the following fractional program:

$$\arg\min_{\mathbf{w}\in\mathbb{R}^n} \quad \frac{\left\| \frac{1}{\gamma}z\|\mathbf{w}\|^2 + X\mathbf{w} - y - \frac{1}{\gamma}\|\mathbf{w}\|^2 y \right\|^2}{(1+\frac{1}{\gamma}\|\mathbf{w}\|^2)^2}$$

To get rid of higher degree terms, we can introduce a new variable $\alpha \in \mathbb{R}$ and set it equal to $\|\mathbf{w}\|^2$:

$$\arg\min_{\mathbf{w},\alpha} \quad \frac{\left\| \frac{\alpha}{\gamma}z + X\mathbf{w} - \mathbf{y} - \frac{\alpha}{\gamma}\mathbf{z} \right\|^2}{(1+\frac{\alpha}{\gamma})^2} \quad \text{s.t.} \quad \alpha = \|\mathbf{w}\|^2 \tag{3.6}$$

Recall that we started with a quadratic bilevel optimisation problem. We obtained a single-level reformulation through replacing the internal optimisation problem faced

by each data provider with its Karush-Kuhn-Tucker (KKT) conditions. This is a standard approach within the bilevel optimisation literature (Dempe, 2002; Strekalovsky and Orlov, 2020). Additionally, observe that the lower-level optimisation problem faced by each data provider is unconstrained. Hence, the single-level reformulation we obtain contains no bilinear constraints corresponding to complementarity constraints within a KKT system. Such constraints are the primary source of nonconvexity in single-level reformulations of quadratic bilevel optimisation problems (Strekalovsky and Orlov, 2020).

As a result, one may hope to find an algorithm for global optimisation that is more efficient than standard algorithms for bilevel optimisation, which typically rely on methods such as branch-and-bound that scale poorly as the number of variables increases (Strekalovsky and Orlov, 2020; Muu and Quy, 2003; Audet et al., 2007). This is especially important in big data contexts, where learners may be regressing on more than a thousand features at a time. Unfortunately, established algorithms for bilevel optimisation typically only scale to the order of 50-500 variables (Audet et al., 2007; Gruzdeva and Petrova, 2010; de Sabóia et al., 2004).

Returning to our own analysis, observe that Problem (3.6) takes the form of a quadratic fractional program with a quadratic equality constraint. In what follows, we will further reformulate Problem (3.6) using classical techniques from the fractional programming literature.

3.4 Dinkelbach's Lemmas for Fractional Programming

Before proceeding, we first review several results from the fractional programming literature which our analysis will leverage. Consider a fractional program of the following form:

$$\min_{\mathbf{w}\in\mathcal{W}} \quad \frac{N(\mathbf{w})}{D(\mathbf{w})} \tag{3.7}$$

where both $N : \mathbb{R}^n \to \mathbb{R}$ and $D : \mathbb{R}^n \to \mathbb{R}$ are continuous functions and $\mathcal{W}$ is a compact subset of $\mathbb{R}^n$. With Problem (3.7) in mind, we may consider the following parameterised optimisation problem:

$$\min_{\mathbf{w}\in\mathcal{W}} \quad N(\mathbf{w}) - qD(\mathbf{w}) \tag{3.8}$$

where $q \in \mathbb{R}$ is a parameter. In what follows, we will refer to Problem (3.8) as the Dinkelbach program associated with Problem (3.7). Moreover, we may define a function $F : \mathbb{R} \to \mathbb{R}$ mapping any q to the optimal value of the corresponding Dinkelbach program. That is,

$$F(q) = \min_{\mathbf{w}\in\mathcal{W}} \quad N(\mathbf{w}) - qD(\mathbf{w}).$$

We refer to F as the Dinkelbach function associated with Problem (3.7). Dinkelbach (1967) uncovered several relationships between F and Problem (3.7), which we use in our own analysis. More specifically, the following two results are of interest.

Lemma 3.2 (Dinkelbach (1967)). *The function $F(q)$ is continuous in q and is strictly monotonically decreasing. Moreover if $\mathbf{w} \in \mathcal{W}$ and*

$$q = \frac{N(\mathbf{w})}{D(\mathbf{w})}$$

then $F(q) \leq 0$.

Theorem 3.3 (Dinkelbach (1967)). *$F(q^\star) = 0$ has a unique solution. Furthermore $q^\star$ is the optimal value for the fractional program (3.7) if and only if $F(q^\star) = 0$.*

Observe that Theorem 3.3 implies that solving Problem (3.7) reduces to finding a $q^\star \in \mathbb{R}$ such that $F(q^\star) = 0$. Moreover, Lemma 3.2 shows that the function F is monotone, and thus, its roots can be found via bisection search. Hence, instead of solving Problem (3.6) directly, we may instead consider the Dinkelbach function associated with Problem (3.6):

$$F(q) = \arg\min_{\mathbf{w},\alpha} \quad \left\| \frac{\alpha}{\gamma} z + X\mathbf{w} - \mathbf{y} - \frac{\alpha}{\gamma}\mathbf{y} \right\|^2 - q\left(1 + \frac{\alpha}{\gamma}\right)^2 \quad \text{s.t.} \quad \alpha = \|\mathbf{w}\|^2 \tag{3.9}$$

and apply bisection search to find its root. However, evaluating $F(q)$ involves solving a quadratically constrained quadratic program with a single constraint (QC1QP). Thankfully, QC1QPs exhibit hidden convexity properties which can be exploited via the application of an S-lemma, a theorem of the alternative for quadratic polynomials analogous to Farkas Lemma in conic optimisation. In the following section, we detail how the S-lemma can be used to reformulate the problem of evaluating $F(q)$ as an SDP, before putting everything together to design a practical algorithm.

3.5 Applying the S-Lemma

To start this section, we first review and introduce the S-procedure, a common technique for reformulating quadratically constrained quadratic programs into a more amenable form. In particular, we review the S-Lemma with equality, recently proposed by Xia et al. (2016). After this, we show that the S-lemma can be applied to reformulate $F(q)$, described in Equation (3.9), as an SDP.

Consider a system of two quadratic conditions:

$$\begin{aligned} f(\mathbf{w}) &= \mathbf{w}^\top A\mathbf{w} + 2\mathbf{a}^\top\mathbf{w} + c \geq 0 \\ h(\mathbf{w}) &= \mathbf{w}^\top B\mathbf{w} + 2\mathbf{b}^\top\mathbf{w} + d = 0 \end{aligned} \tag{3.10}$$

where $A, B \in \mathbb{R}^{n\times n}$ are symmetric matrices, $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ and $c, d \in \mathbb{R}$. Additionally, assume that for h, the dual Slater condition holds:

Assumption 1 (Dual Slater Condition). There exists $\mathbf{w}_1 \in \mathbb{R}^n$ and $\mathbf{w}_2 \in \mathbb{R}^n$ such that $h(\mathbf{w}_1) < 0$ and $h(\mathbf{w}_2) > 0$.

The S-Lemma with equality characterises when such a system of quadratic inequalities is feasible.

Theorem 3.4 (Xia et al. (2016))**.** *Consider a system of quadratic inequalities in the form of* (3.10)*. In addition, assume that the dual Slater condition holds, and that $B \neq 0$. Then the following statements are equivalent:*

(i) $h(\mathbf{w}) = 0 \Rightarrow f(\mathbf{w}) \geq 0 \quad \forall \mathbf{w} \in \mathbb{R}^n$.

(ii) There exists a number $\lambda \in \mathbb{R}$ such that $f(\mathbf{w}) + \lambda h(\mathbf{w}) \geq 0$ for all $\mathbf{w} \in \mathbb{R}^n$.

Note that the S-lemma described above is just one of many, each tailored for pairs of quadratic inequalities satisfying certain properties. S-lemmas find applications in many branches of mathematics, including control theory and optimisation. In particular, the main use of S-lemmas is the reformulation of quadratically constrained quadratic programs into SDPs. In particular, the following lemma shows how the QC1QP defined by $F(q)$ can be reformulated into an SDP by leveraging the S-Lemma with equality.

Lemma 3.5. *For any $q \in \mathbb{R}$, evaluating $F(q)$, as defined in* (3.9)*, is equivalent to solving the following semidefinite program:*

$$\max_{\tau\in\mathbb{R},\lambda\in\mathbb{R}} \quad \tau \quad s.t. \quad \begin{bmatrix} A+\lambda B & \mathbf{a}+\lambda\mathbf{b} \\ \mathbf{a}^\top+\lambda\mathbf{b}^\top & c-\tau \end{bmatrix} \succeq 0 \tag{3.11}$$

where:

$$\mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{a} = \begin{bmatrix} -X^\top\mathbf{y} \\ -\frac{1}{\gamma}(\mathbf{z}-\mathbf{y})^\top\mathbf{y} - \frac{q}{\gamma} \end{bmatrix}, \quad B = \begin{bmatrix} -I & 0 \\ 0 & 0 \end{bmatrix}, \quad c = \|\mathbf{y}\|^2 - q$$

and

$$A = \begin{bmatrix} X^\top X & \frac{1}{\gamma}X^\top(\mathbf{z}-\mathbf{y}) \\ \frac{1}{\gamma}X^\top(\mathbf{z}-\mathbf{y}) & \frac{1}{\gamma^2}\|\mathbf{z}-\mathbf{y}\|^2 - \frac{q}{\gamma^2} \end{bmatrix}.$$

Proof. Firstly, note that $F(q)$ may be rewritten as follows:

$$\begin{aligned} \min_{\mathbf{u}\in\mathbb{R}^{n+1}} \quad & f(\mathbf{u}) = \mathbf{u}^\top A\mathbf{u} + 2\mathbf{a}^\top\mathbf{u} + c \\ \text{s.t.} \quad & h(\mathbf{u}) = \mathbf{u}^\top B\hat{\mathbf{w}} + 2\mathbf{b}^\top\mathbf{u} = 0. \end{aligned} \tag{3.12}$$

In the next steps of the proof, we perform the S-procedure, using the S-lemma with equality. To begin, note that Problem (3.12) may be reformulated further:

$$\max_{\tau \in \mathbb{R}} \quad \tau \quad \text{s.t.} \quad h(\mathbf{u}) = 0 \Rightarrow f(\mathbf{u}) - \tau \geq 0 \quad \forall \mathbf{u} \in \mathbb{R}^{n+1}.$$

Then, since $B \neq 0$, we can apply the S-lemma with equality (Theorem 3.4) to rewrite the above optimisation problem:

$$\max_{\tau \in \mathbb{R}, \lambda \in \mathbb{R}} \quad \tau \quad \text{s.t.} \quad f(\mathbf{u}) - \tau + \lambda h(\mathbf{u}) \geq 0 \quad \forall \mathbf{u} \in \mathbb{R}^{n+1}. \tag{3.13}$$

Furthermore, note that the constraint in Problem (3.13) can be expressed like so:

$$\begin{bmatrix} \mathbf{u} \\ 1 \end{bmatrix}^\top \begin{bmatrix} A + \lambda B & \mathbf{a} + \lambda \mathbf{b} \\ \mathbf{a}^\top + \lambda \mathbf{b}^\top & c - \tau \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ 1 \end{bmatrix} \geq 0 \quad \forall \mathbf{u} \in \mathbb{R}^{n+1} \tag{3.14}$$

In what follows, we will refer to the matrix in the above constraint by M. It is easy to see that the constraint above is equivalent to a positive semidefinite constraint on M. For the sake of contradiction, suppose that $\exists \mathbf{v} \in \mathbb{R}^{n+2}$ such that $\mathbf{v}^\top M \mathbf{v} < 0$ and (3.14) holds. If the final coordinate of $\mathbf{v}$ is nonzero, we can simply rescale $\mathbf{v}$ by this coordinate to achieve a contradiction. If the final coordinate of $\mathbf{v}$ is nonzero, then we can use the continuity of quadratic forms to argue that there must exist a $\tilde{\mathbf{v}}$ whose final coordinate is not equal to zero such that $\tilde{\mathbf{v}}^\top M \tilde{\mathbf{v}} < 0$, bringing us back to the nonzero case. Thus, we can replace the constraint (3.14) as follows:

$$\max_{\tau \in \mathbb{R}, \lambda \in \mathbb{R}} \quad \tau \quad \text{s.t.} \quad \begin{bmatrix} A + \lambda B & \mathbf{a} + \lambda \mathbf{b} \\ \mathbf{a}^\top + \lambda \mathbf{b}^\top & c - \tau \end{bmatrix} \succeq 0$$

and the proof is complete. □

Lemma 3.5 shows that evaluating $F(q)$ corresponds to solving an SDP. Fortunately, SDPs can be solved in polynomial time via interior point methods (Boyd and Vandenberghe, 2004). Assuming that strong conic duality holds, a vector $\mathbf{w}$ which attains $F(q)$ on Problem (3.6) can be found by taking a rank-1 decomposition of dual variables. A description of the dual program may be found in Appendix A. Moreover, as each SDP has a rank-1 solution, standard low-rank approximation schemes for SDPs can be applied in order to cope with data of high dimension (Burer and Monteiro, 2003, 2005). In addition, a variety of first-order methods tailored to SDPs can be applied in scenarios where evaluating the Hessian is prohibitively expensive (Sra et al., 2011).

3.6 A Polynomial Time Algorithm For Square Losses

Finally, we introduce our proposed algorithm, SDP-BISECT, for solving Problem (3.4). The pseudocode for SDP-BISECT is detailed in Algorithm 1. As previously mentioned SDP-BISECT aims to solve fractional program (3.6), which is equivalent to Problem (3.4). More specifically we consider the Dinkelbach function F associated with Problem (3.6). By Theorem 3.3, $F(q^\star) = 0$ implies that the linear predictor $\mathbf{w}^\star$ corresponding to $F(q^\star)$ is a global solution to Problem (3.6) and thus Problem (3.4). Additionally, we know from Lemma 3.2 that F is a concave monotonically decreasing continuous function. As a result, given $q_1, q_2 \in \mathbb{R}$, for which $F(q_1) \leq 0$ and $F(q_2) \geq 0$, we can employ bisection search to find $q^\star$ and hence $\mathbf{w}^\star$. The following theorem characterises the convergence rate of SDP-BISECT.

Algorithm 1: The SDP-BISECT algorithm

Input : data matrix X, learner's labels $\mathbf{y}$, data provider's labels $\mathbf{z}$, tolerance ϵ

Initialize $q_1 = 0$
Initialize $q_2 = \mathbf{y}^\top \mathbf{y}$
Initialize $\mathbf{w} = \mathbf{0}$
while $q_2 - q_1 > \epsilon$ **do**
  $q = (q_1 + q_2)/2$
  Evaluate $F(q)$ by solving the SDP detailed in Lemma 3.5
  Set $\mathbf{v}$ equal to the rank-1 decomposition of dual variables associated with $F(q)$
  if $F(q) \geq 0$ **then**
    $q_1 = q$
  else
    $q_2 = q$
    $\mathbf{w} = \mathbf{v}[1 : n]$
end
return $\mathbf{w}, q_2$

Theorem 3.6. *Algorithm 1 takes at most* $\log_2(2\|y\|^2/\epsilon)$ *iterations to return a* $q \in \mathbb{R}$ *such that* $q - q^* \leq \epsilon$

Proof. For bisection search to converge to a $q^\star$ such that $F(q^\star) = 0$ we require two points. One point, $q_1 \in \mathbb{R}$, for which $F(q_1) \geq 0$, and another point, $q_2 \in \mathbb{R}$, for which $F(q_2) \leq 0$. We begin by claiming that setting $q_1 = 0$ and $q_2 = \|\mathbf{y}\|^2$ satisfies these conditions.

Finding a q_2 such that $F(q_2) \leq 0$ is simple. Lemma 3.2 tells us that we can choose any feasible point for Problem (3.6) and simply set q_2 to the value of the objective at the chosen point. Thus, we choose the zero vector which leads to an objective value of $\|\mathbf{y}\|^2$.

To find a q_2 such that $F(q) \geq 0$, we employ both Lemma 3.2 and Theorem 3.3. Since $F(q^\star) = 0$ and F is strictly monotonically decreasing, any value which lower bounds

$q^\star$ will map to a nonnegative value when passed to F. Since the objective of Problem (3.6) is always nonnegative, we can select $q_2 = 0$ as a lower bound.

Since $F(q_1)$ is nonnegative, $F(q_2)$ is nonpositive and F is continuous, $q^\star$ must lie in the interval $[q_1, q_2]$ by the intermediate value theorem. Thus $q = (q_1 + q_2)/2$, the mid-point of $[q_1, q_2]$, is at most $|q_1 - q_2|/2$ in distance from $q^\star$. Note that, in Algorithm 1, q_1 is initialised at 0 and q_2 is initialised at $\|y\|^2$. Thus, initially $|q_1 - q_2| = \|\mathbf{y}\|^2$.

At each iteration, the interval $[q_1, q_2]$ is updated and halved in length. Therefore, after $\log_2(2\|y\|^2/\epsilon)$ iterations, $|q - q^\star| \leq \epsilon/2$. After the same number of iterations $q_2 - q = \epsilon/2$. Therefore, $q_2 - q^\star \leq \epsilon$. As q_2 is returned by Algorithm 1, we have proved the result. □

Theorem 3.6 essentially states that the `SDP-BISECT` algorithm converges linearly to a global solution of Problem (3.4). In contrast, existing methods can only guarantee convergence to stationary points which satisfy the Karush-Kuhn-Tucker (KKT) conditions. Additionally, Theorem 3.6 indicates that the subclass of SPGs described by Problem (3.3) are easy relative to the general case.

3.7 Extensions to Kernel Methods

In this section we briefly describe a version of Algorithm 1 based on kernel methods. In some instances, we may wish to apply a high dimensional feature mapping, ϕ, to data points before making a prediction. Typically, it is assumed that the feature mapping ϕ maps each element of the input space, $\mathcal{X}$, to an element of a reproducing Hilbert kernel space (RKHS), $\mathcal{F}$, with corresponding kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. In the standard least squares linear regression setting, where data is sampled cleanly without manipulation, this leaves us with the following optimisation problem:

$$\arg\min_{\mathbf{w}\in\mathbb{R}^n} \sum_{i=1}^{m} (\mathbf{w}^T \phi(\mathbf{x}_i) - y_i)^2$$

By leveraging the representer theorem (Schölkopf et al., 2001), it can be shown that there exists an optimal solution $\mathbf{w}^\star \in \mathcal{F}$ with the following form:

$$\mathbf{w}^\star = \sum_{i=1}^{m} \beta_i \phi(\mathbf{x}_i) \tag{3.15}$$

where $\beta_i \in \mathbb{R}$ for all $i \in [m]$. Moreover, the coefficients β_i can be characterised by the Gram matrix $K \in \mathbb{R}^{m\times m}$, where $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. In contrast, for SPGs under square losses, assuming that the feature map ϕ is surjective, we are left with the following

optimisation problem:

$$\arg\min_{\mathbf{w}\in\mathbb{R}^n} \quad \sum_{i=1}^{m}\left(\frac{\frac{1}{\gamma}z_i\|\mathbf{w}\|^2+\omega^\top\phi(\mathbf{x}_i)}{1+\frac{1}{\gamma}\|\mathbf{w}\|^2}-y_i\right)^2.$$

In this case, the representer theorem cannot be applied, as the prediction made by a given predictor $\mathbf{w}$ depends on its inner product with each mapped data point *and* its own norm, rather than just the former. However, if we optimise over predictors of the form described in equation (3.15), then we can obtain a new version of Algorithm 1 which returns the optimal vector of coefficients, $\beta=(\beta_1,\dots\beta_m)^\top$, wherein the following terms are redefined:

$$\mathbf{b}=\begin{bmatrix}0\\1\end{bmatrix},\quad \mathbf{a}=\begin{bmatrix}-K^T\mathbf{y}\\-\frac{1}{\gamma}(\mathbf{z}-\mathbf{y})^\top\mathbf{y}-\frac{q}{\gamma}\end{bmatrix},\quad A=\begin{bmatrix}K^TK & \frac{1}{\gamma}K(\mathbf{z}-\mathbf{y})\\ \frac{1}{\gamma}K^T(\mathbf{z}-\mathbf{y}) & \frac{1}{\gamma^2}\|\mathbf{z}-\mathbf{y}\|^2-\frac{q}{\gamma^2}\end{bmatrix}$$

Whilst it cannot be guaranteed that the predictor output by this algorithm will be optimal, we can guarantee that the predictor is better than any other predictor in the span of mapped training data points, which by the representer theorem, contains the optimal predictor for the classical clean data setting.

3.8 Empirical Evaluation

Given the theoretical performance analysis of our proposed algorithm, we now demonstrate that it is significantly more accurate in practice compared to state of the art approaches. More specifically, we evaluate Algorithm 1 on two real world datasets and compare it against both ridge regression, which is the optimal approach under the assumption that data providers are completely adversarial, and to the single level nonconvex reformulation of the SPG originally proposed by Brückner and Scheffer (2011). The first of these datasets is the medical personal costs dataset (Choi, 2018), which we use to model the problem faced by an insurer when providing quotes to customers. Additionally, we also consider the red wine dataset Cortez et al. (2009), where we assume that agents take the role of wine producers aiming for a specific quality rating. In what follows, we describe our findings with respect to the medical personal costs dataset in detail. Our experimental results regarding the red wine dataset, as well as our results analysing the runtime of `SDP-BISECT` are deferred to the appendices.

The medical personal costs dataset consists of 1338 instances each with 7 features (Choi, 2018). Each feature details information regarding an individual. Some are continuous, such as the individual's age and body mass index, while others are categorical, such as region and smoking status. So that we can perform linear regression on all features, categorical features are transformed into one-hot vectors,

increasing the total number of features from 7 to 13. The response variable is the individual's medical costs billed by their health insurance.

We consider a scenario in which insurers would like to predict the medical costs of a new customer in order to provide a reasonable insurance quote. We assume that individuals may provide fake data in the hope of receiving a lower quote. Similar to the experimental design of Tong et al. (2018), we define each data provider's desired outcome as $\mathbf{z}_i = \mathbf{y}_i + \delta_i$, where δ_i denotes the change in medical billing that each individual is striving for. In particular, we consider two types of provider, $\mathcal{A}_{\text{modest}}$, who wishes to reduce their predicted medical billing by \$100 ($\delta_{\text{modest}} = -100.0$), and $\mathcal{A}_{\text{severe}}$, who wishes to reduce their predicted medical billing by \$300 ($\delta_{\text{severe}} = -300.0$). Since medical charges range from \$1000 to \$63,000, we numerically scale the data labels by dividing them by 100 before passing them to each algorithm.

In order to evaluate Algorithm 1, we perform 10-fold cross validation and compare its performance to ridge regression for $\gamma \in [1 \times 10^{-5}, 1]$. For each value of γ, we compute the regularisation parameter for ridge regression via grid search on 8 logarithmically spaced points in the interval $[1 \times 10^{-5}, 1000]$ during cross validation. For ridge regression, and the SDPs in Algorithm 1, we use the `SDPT3` solver (Toh et al., 1999) to find global solutions. We also compare Algorithm 1 to the nonconvex single level reformulation of the SPG originally proposed by Brückner and Scheffer (2011). We employ the interior point method from the `MATLAB` optimisation toolbox to find a stationary point for this problem reformulation. The same error tolerances are used for both Algorithm 1 and the interior point method we use to solve nonconvex problem reformulation proposed by Brückner and Scheffer (2011). Figure 3.1 shows

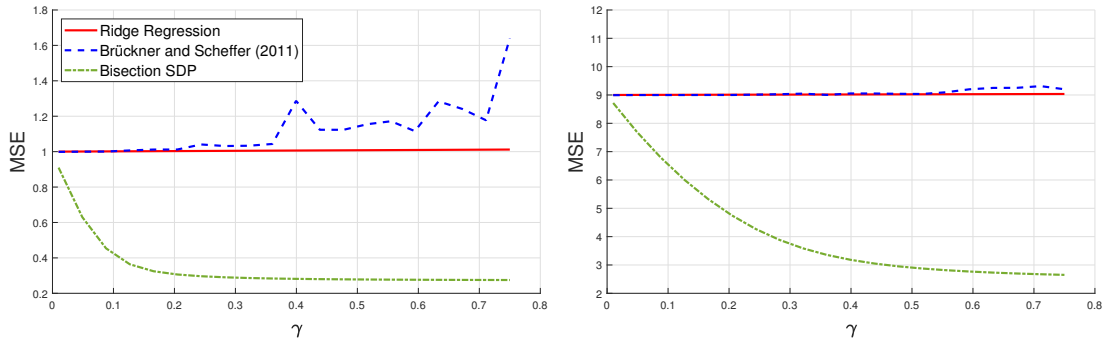

FIGURE 3.1: A performance comparison between different algorithms ran on the medical personal costs dataset. The left plot compares the average MSE of each algorithm during 10-fold cross validation where data was generated by $\mathcal{A}_{\text{modest}}$, whilst the right plot shows the average MSE where data is generated by $\mathcal{A}_{\text{severe}}$

the average mean squared error (MSE) achieved by each algorithm on the medical personal costs dataset. Firstly, observe that Algorithm 1 outperforms both ridge regression and the nonconvex problem reformulation for every value of γ. Also note

that the MSE of Algorithm 1 is very stable, whilst the interior point solution seems to behave erratically for higher values of γ.

For values of $\gamma > 0.4$, we observe that, for modest data providers, our algorithm is at least \$45 more accurate than ridge regression on average. For severe data providers, we observe an even greater difference. For example, when $\gamma > 0.5$, the predictions made by our algorithm are at least \$120 more accurate than ridge regression on average. As one would expect, the benefits of explicitly modelling the goals of data providers becomes more beneficial as γ increases, as the data provider's capability for manipulation becomes more limited.

3.9 Bounding Rademacher for Square Losses

We now shift our focus from optimisation and consider the statistical properties of STERM. From Lemma 3.1, it is obvious that given a sample $(\mathbf{x}, y, z)$ the prediction computed by a predictor $\mathbf{w}$, after the agent has modified the input, is given by

$$h_{\mathbf{w}}(\mathbf{x}, z) = \frac{\mathbf{w}^\top \mathbf{x} + \frac{1}{\gamma}\|\mathbf{w}\|^2 z}{1 + \frac{1}{\gamma}\|\mathbf{w}\|^2}. \tag{3.16}$$

Hence, the problem of minimising Stackelberg risk for square losses may be reformulated as a standard risk minimisation problem:

$$\min_{h \in \mathcal{H}} \mathbb{E}[(h(\mathbf{x}, z) - y)^2]$$

where $\mathcal{H} = \{h_{\mathbf{w}} : \mathbf{w} \in \mathbb{R}^n\}$. In particular, note that for any $\mathbf{w} \in \mathbb{R}^n$, $h_{\mathbf{w}}$ is linear in both $\mathbf{x}$ and z. Thus, $\mathcal{H}$ is a subset of the linear predictors of dimension $n + 1$. Additionally, the following lemma shows that $\mathcal{H}$ is bounded in terms of L_2-norm.

Lemma 3.7. *For all $\mathbf{w} \in \mathbb{R}^n$, the following inequality holds:*

$$\|h_{\mathbf{w}}\|_2 \leq \begin{cases} 1 & \text{if } \gamma \leq 2 \\ \frac{\gamma}{2\sqrt{\gamma - 1}} & \text{if } \gamma > 2 \end{cases}$$

Proof. Note that

$$\begin{aligned}\|h_{\mathbf{w}}\|_2^2 &= \frac{1}{(1+\frac{1}{\gamma}\|\mathbf{w}\|^2)^2}\begin{bmatrix}\mathbf{w}\\ \frac{1}{\gamma}\|\mathbf{w}\|^2\end{bmatrix}^\top\begin{bmatrix}\mathbf{w}\\ \frac{1}{\gamma}\|\mathbf{w}\|^2\end{bmatrix}\\ &= \frac{\|\mathbf{w}\|^2+\frac{1}{\gamma^2}\|\mathbf{w}\|^4}{(1+\frac{1}{\gamma}\|\mathbf{w}\|^2)^2}\\ &= \frac{\gamma^2\|\mathbf{w}\|^2+\|\mathbf{w}\|^4}{(\gamma+\|\mathbf{w}\|^2)^2}\end{aligned}$$

Peforming the substituion $a = \|\mathbf{w}\|^2$, we have:

$$\sup_{\mathbf{w}}\|h_{\mathbf{w}}\|_2^2 = \sup_{a\in\mathbb{R}_+}\frac{\gamma^2 a + a^2}{(\gamma+a)^2} = \sup_{a\in\mathbb{R}_+} g(a) \tag{3.17}$$

where $g(a) = \frac{\gamma^2 a+a^2}{(\gamma+a)^2}$. Note that:

$$g'(a) = \frac{(2\gamma-\gamma^2)a^2 + 2\gamma^2 + \gamma^4}{(\gamma+a)^4}$$

As a result, if $\gamma \le 2$, then $g(a)$ is monotonically increasing on $\mathbb{R}_+$, and hence:

$$\sup_{a\in\mathbb{R}_+} g(a) = \lim_{a\to\infty} g(a) = 1$$

proving the first case. When $\gamma > 2$, through standard calculus, we observe that $f(a)$ attains a maximum at $\frac{\gamma^2}{4(\gamma-1)}$. Taking square roots proves the second case. □

An immediate corollary of Lemma 3.7 is that $\mathcal{H}$ has bounded Rademacher complexity.

Corollary 3.8. *Let $\mathcal{H} = \{h_{\mathbf{w}} : \mathbf{w}\in\mathbb{R}^n\}$, where $h_{\mathbf{w}}$ is defined as in Equation* (3.16)*. Moreover, assume that $\|\mathbf{x}\|_2 \le C$ for all $\mathbf{x}$ in the support of $\mathcal{D}$ and that $z\in[-r,r]$ for all z that lie in the support of $\mathcal{D}$. Then,*

$$\mathcal{R}_m(\mathcal{H}) \le \begin{cases}\sqrt{\frac{C^2+r^2}{m}} & \text{if } \gamma < 2\\ \frac{\gamma}{2\sqrt{\gamma-1}}\sqrt{\frac{C^2+r^2}{m}} & \text{if } \gamma \ge 2\end{cases}$$

Proof. Observe that

$$\left\|\begin{bmatrix}\mathbf{x}_i\\ z_i\end{bmatrix}\right\|_2^2 \le C^2 + r^2. \tag{3.18}$$

Assume that $\gamma < 2$. Then,

$$\begin{aligned}\mathcal{R}_m(\mathcal{H}) &= \mathbb{E}\left[\sup_{\mathbf{w}\in\mathbf{R}^n} \frac{1}{m}\sum_{i=1}^{m} h_{\mathbf{w}}(\mathbf{x}_i, z_i)\sigma_i\right] \\ &\leq \mathbb{E}\left[\sup_{\mathbf{v}:\|\mathbf{v}\|_2\leq 1} \frac{1}{m}\sum_{i=1}^{m} \mathbf{v}^\top \begin{bmatrix}\mathbf{x}_i \\ z_i\end{bmatrix}\sigma_i\right] \\ &\leq \sqrt{\frac{C^2+r^2}{m}}\end{aligned}$$

where the first inequality follows from Lemma 3.7, and the second inequality follows from Theorem 2.3 and inequality (3.18). Now, assume that $\gamma \geq 2$. The same reasoning applies in this case, except that $\frac{\gamma}{2\sqrt{\gamma-1}}$ is used in place of 1. □

Corollary 3.8 implies that data providing agents limit the expressibility of the hypothesis class available to the learner. In particular, the smaller γ, the less expressibility the learner has. This is what one would intuitively expect. When γ is small, each agent can make large modifications to their input data at small cost. Thus, no matter the predictor chosen by the learner, all predictions will be driven towards the labels desired by each agent. In contrast, when γ is large, each agent can only make large modfications to the input data when the weight vector chosen by the learner is also large. Therefore, weight vectors of small norm are relatively unaffected by the actions of each agent. As a result, the learner can still produce a wide variety of labellings by choosing from weight vectors which are small in norm.

3.10 An Improved Scheme

Note that our proposed algorithm, `SDP-BISECT`, consists of bisection search wherein each step consists of solving a semidefinite program. One may ask, given the similarities between the SDPs solved at each iteration, if bisection search is necessary, or whether one may find an optimal predictor by solving a single SDP. This question was first answered in the affirmative by Wang et al. (2021) who showed that Problem (3.4) may be reformulated directly as a single SDP, largely inspired by the analysis of Algorithm 1. In what follows, we present our own proof of this result. In contrast to the proof of Wang et al. (2021), our proof relies directly on the lemmas of Dinkelbach.

Theorem 3.9. *Let $(q^\star, \lambda^\star)$ form an optimal solution to the following semidefinite program:*

$$\max_{q\in\mathbb{R},\lambda\in\mathbb{R}} \quad q \quad s.t. \quad \begin{bmatrix} A+\lambda B - qC & \mathbf{a}+\lambda\mathbf{b}-q\mathbf{c} \\ \mathbf{a}^\top+\lambda\mathbf{b}^\top-q\mathbf{c}^\top & c-q\end{bmatrix} \succeq 0 \tag{3.19}$$

where,

$$\mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{a} = \begin{bmatrix} -X^\top \mathbf{y} \\ -\frac{1}{\gamma}(\mathbf{z}-\mathbf{y})^\top \mathbf{y} - \frac{q}{\gamma} \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} 0 \\ \frac{1}{\gamma} \end{bmatrix}, \quad c = \|\mathbf{y}\|^2$$

and

$$A = \begin{bmatrix} X^\top X & \frac{1}{\gamma} X^\top (\mathbf{z}-\mathbf{y}) \\ \frac{1}{\gamma} X^\top (\mathbf{z}-\mathbf{y}) & \frac{1}{\gamma^2}\|\mathbf{z}-\mathbf{y}\|^2 - \frac{q}{\gamma^2} \end{bmatrix}, \quad B = \begin{bmatrix} -I & 0 \\ 0 & 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{\gamma^2} \end{bmatrix}$$

then $q^\star$ is the optimal value for Problem (3.4).

Proof. Recall that $q^\star$ is an optimal solution to Problem (3.4) if and only if $F(q^\star) = 0$, where F is the Dinkelbach program associated with Problem (3.4). From Lemma 3.5 it follows that if $F(q) = \tau \geq 0$ then

$$\begin{bmatrix} A + \lambda B - qC & \mathbf{a} + \lambda \mathbf{b} - q\mathbf{c} \\ \mathbf{a}^\top + \lambda \mathbf{b}^\top - q\mathbf{c}^\top & c - q - \tau \end{bmatrix} \succeq 0.$$

As a result, it follows that

$$\begin{bmatrix} A + \lambda B - qC & \mathbf{a} + \lambda \mathbf{b} - q\mathbf{c} \\ \mathbf{a}^\top + \lambda \mathbf{b}^\top - q\mathbf{c}^\top & c - q \end{bmatrix} \succeq 0$$

for all q such that $F(q) \geq 0$. According to Lemma 3.2, F is monotonically decreasing. Thus, it follows immediately that $q^\star$ is the solution to the following the optimisation problem

$$\max_{q \in \mathbb{R}, \lambda \in \mathbb{R}} \quad q \quad \text{s.t.} \quad \begin{bmatrix} A + \lambda B - qC & \mathbf{a} + \lambda \mathbf{b} - q\mathbf{c} \\ \mathbf{a}^\top + \lambda \mathbf{b}^\top - q\mathbf{c}^\top & c - q \end{bmatrix} \succeq 0$$

as claimed by the theorem. □

3.11 Setting γ

So far, we have assumed that γ is given. In practice, the learner must select an appropriate value for γ which characterises the cost of manipulation for data providers. In cases where the training dataset is obtained through auditing of past interactions, the learner may have access to an augmented dataset in which each sample is a tuple of the form $(\mathbf{w}, \mathbf{x}, \hat{\mathbf{x}}, y, z)$, where $\mathbf{w}$ denotes the linear predictor in use when the data provider reported their information and $\hat{\mathbf{x}}$ denotes their misreport.

Observe that, in this case, the learner may reverse engineer a γ parameter for each individual agent. More specifically, the corresponding parameter γ_i for agent i's

misreport can be described as follows:

$$\gamma_i = \frac{(z_i - \mathbf{w}^\top \hat{\mathbf{x}}_i)\|\mathbf{w}\|^2}{\mathbf{w}^\top(\hat{\mathbf{x}}_i - \mathbf{x}_i)}$$

Note that γ_i may differ across data points. In this case, we may specify a generalisation of Problem 3.4 in which γ is tailored to each individual agent. Following a similar analysis to Lemma 3.1, we end at the following optimisation problem:

$$\arg\min_{\mathbf{w}\in\mathbb{R}^n} \quad \sum_{i=1}^{m} \left(\frac{\frac{1}{\gamma_i} z_i \|\mathbf{w}\|^2 + \mathbf{w}^\top \mathbf{x}_i}{1 + \frac{1}{\gamma_i}\|\mathbf{w}\|^2} - y_i \right)^2 . \tag{3.20}$$

Note that Problem (3.20) is a sum-of-ratios problem. In general, sum-of-ratios problems are significantly more challenging than conventional fractional programs (Schaible and Shi, 2003). Though solving Problem 3.20 is outside the scope of this chapter, we proceed to highlight several promising approaches that may yield efficient algorithms.

For example, Gruzdeva and Strekalovsky (2018) provide a generalisation of Dinkelbach's results to sum-of-ratios problems, identifying sufficient conditions for optimality in terms of the Dinkelbach functions for each fractional term of the objective. Motivated by this result, the authors propose an algorithm for global optimisation consisting of a modified bisection search algorithm wherein a nonlinear optimisation problem is solved at each iteration. Unfortunately, the algorithm proposed has poor empirical performance in practice, taking many iterations of bisection search to converge. However, special structural properties of Problem (3.20) may allow a specialised version of this algorithm to be designed that converges rapidly.

Aside from approaches inspired by the insights of Dinkelbach, branch-and-bound methods are a popular family of techniques for solving nonlinear sum-of-ratios problems. For instance, Jiao et al. (2013) proposes a branch-and-bound algorithm for global optimisation which relies on a parametric linear programming relaxation of Problem (3.20) to produce lower bounds on the objective function. Similar approaches, based on linear-fractional and linear programming relaxations, have been adopted by several authorsn (Qu et al., 2007; Ji et al., 2012). Unfortunately, branch-and-bound algorithms are typically impractical for large scale machine learning problems, where the dimension of input data is high. As a result, we conjecture that such algorithms may not be readily applicable to our setting.

Lastly, we highlight an approach taken by Wang et al. (2022) to solve Problem (3.4). More specifically, Wang et al. (2022) propose a spherically constrained least squares reformulation of Problem (3.4) where each feasible vector of parameters is mapped to the unit sphere. The reformulation proposed takes the form of a generalised

trust-region subproblem which can be readily solved. Once a solution is computed one may simply take the inverse map of solution parameters to find a solution to the original problem. The mapping proposed by Wang et al. (2022) is given below:

$$\tilde{\mathbf{w}} = \frac{2\gamma}{(\alpha+1)}\mathbf{w} \quad \text{and} \quad \tilde{\alpha} = \frac{\alpha-1}{\alpha+1}.$$

Note that $\tilde{\alpha}$ is specifically tuned to eliminate the denominator in Problem (3.4), so that the transformed objective is no longer fractional. Additionally, note that the reparameterisation implicitly depends on γ. When γ is free to vary across data providers, the mapping proposed by Wang et al. (2022) is no longer valid. However, we conjecture that a generalised reformulation may be possible.

3.12 Open Problems

We now take a moment to describe several related problems of both theoretical and practical interest. So far, we have assumed that the cost function associated with each agent is fixed. In practice, alternative cost functions may be better modelling choice for many real world scenarios. For example, one can consider the subclass of SPGs in which square Euclidean distance replaced by the L_1, or taxicab, distance:

$$\begin{aligned} \ell_{-1}(\tilde{y}, y) &= (\tilde{y} - y)^2 \\ \ell_{+1}(\tilde{z}, z) &= (\tilde{z} - z)^2 \\ c(\mathbf{x}, \tilde{\mathbf{x}}) &= \|\mathbf{x} - \tilde{\mathbf{x}}\|_1. \end{aligned}$$

This results in the following STERM problem:

$$\begin{aligned} \arg\min_{\mathbf{w}\in\mathbb{R}^n} \quad & \frac{1}{m}\sum_{i=1}^{m}(\mathbf{w}^\top \mathbf{x}_i^\star - y_i)^2 \\ \text{s.t.} \quad & \mathbf{x}_i^\star \in \arg\min_{\tilde{\mathbf{x}}_i} (\mathbf{w}^\top \tilde{\mathbf{x}}_i - z_i)^2 + \gamma\|\mathbf{x}_i - \tilde{\mathbf{x}}_i\|_1 \quad \forall i \in [m]. \end{aligned} \tag{3.21}$$

Note that such a problem setting has a natural interpretation. Under the L_1-distance, agents are forced to pick sparse modifications. This mirrors many real world situations in which agents only have the time or resources to modify certain elements of the data they submit, rather than all of them simultaneously. As in the case of the squared Euclidean distance, the lower-level optimisation problem faced by each agent can be solved directly, using standard techniques from calculus, leading to the

following closed-form solution for $\mathbf{x}_i^\star$:

$$\mathbf{x}_i^\star = \begin{cases} \mathbf{x}_i & \text{if } \mathbf{x}_i = 0 \text{ or } \|\mathbf{w}\|_\infty |z_i - \mathbf{w}^\top \mathbf{x}_i| \leq \gamma/2 \\ \mathbf{x}_i + \|\mathbf{w}\|_\infty (z_i - \mathbf{w}^\top \mathbf{x}_i) - \frac{\gamma}{2\|\mathbf{w}\|_\infty} & \text{if } \|\mathbf{w}\|_\infty (z_i - \mathbf{w}^\top \mathbf{x}_i) > \gamma/2 \\ \mathbf{x}_i + \|\mathbf{w}\|_\infty (z_i - \mathbf{w}^\top \mathbf{x}_i) + \frac{\gamma}{2\|\mathbf{w}\|_\infty} & \text{if } \|\mathbf{w}\|_\infty (z_i - \mathbf{w}^\top \mathbf{x}_i) < -\gamma/2. \end{cases}$$

Thus, given a sample $(\mathbf{x}, y, z)$, the prediction made by the learner, after an agent has modified their input example, is given by:

$$h_{\mathbf{w}}(\mathbf{x}, z) = \begin{cases} \mathbf{w}^\top \mathbf{x} & \text{if } \mathbf{w} = 0 \text{ or } \|\mathbf{w}\|_\infty |z - \mathbf{w}^\top \mathbf{x}| \leq \gamma/2 \\ z - \frac{\gamma}{2\|\mathbf{w}\|_\infty} & \text{if } \|\mathbf{w}\|_\infty (z_i - \mathbf{w}^\top \mathbf{x}_i) > \gamma/2 \\ z + \frac{\gamma}{2\|\mathbf{w}\|_\infty} & \text{if } \|\mathbf{w}\|_\infty (z_i - \mathbf{w}^\top \mathbf{x}_i) < \gamma/2. \end{cases}$$

In other words, the learner is implicitly tasked with performing empirical risk minimisation over a set of piecewise linear functions. Obtaining both optimisation and generalisation guarantees for STERM in this setting forms an interesting open problem. Moreover, Problem (3.21) is interesting from the perspective of incentive-compatibility. In the case of the squared Euclidean distance, the only incentive-compatible predictor is the zero vector (this follows trivially from Lemma 3.1). However, when the L_1-distance is employed, a linear predictor is incentive-compatible if and only if:

$$\|\mathbf{w}\|_\infty |z - \mathbf{w}^\top \mathbf{x}| \leq \gamma/2 \quad \forall (\mathbf{x}, y, z) \in \text{supp}(\mathcal{D}) \tag{3.22}$$

where $\text{supp}(\mathcal{D})$ denotes the support of $\mathcal{D}$. As a result, a nontrivial set of linear predictors may be incentive-compatible under the L_1-distance, depending on additional structural assumptions placed on the labels of data providing agents. Thus, the development of learning algorithms for Problem (3.21), which only select from incentive-compatible hypotheses, presents a candidate direction for future work.

3.13 Conclusion

Summarising, we have outlined a new subclass of SPGs, in which each data point comes with two labels; one corresponding to the prediction preferred by the learner, and one corresponding to the prediction preferred by the data providing agent. In particular, we investigate a version of this SPG in which both agents and the learner adopt a square loss function. Under this assumption, we show that the Stackelberg empirical risk can be minimised via a combination of bisection search and semidefinite programming, culminating in the `SDP-BISECT` algorithm, whose performance we illustrate on a number of benchmark datasets. After this, we

investigated the statistical properties of Stackelberg risk minimisation and, in the case of square losses, show that minimising the Stackelberg risk is equivalent to minimising the standard risk with respect to the learner's labels on a subset of bounded linear predictors. This enabled us to leverage standard results in statistical learning theory to provide a generalisation guarantee for Stackelberg empirical risk minimisation in terms of Rademacher complexity.

Since our initial work, there has been a flurry of research regarding SPGs with square losses and square costs. As covered in Section 3.10, Wang et al. (2021) illustrated that STERM can be directly formulated as an SDP, avoiding the need to employ bisection search. Moreover, Wang et al. (2021) provide a second-order cone reformulation of STERM under square losses and square costs. This is significant, as interior point methods for second order cone programming are far more efficient than interior point methods for semidefinite programming.

Recently, Wang et al. (2022) have proposed a spherically constrained least squares reformulation for SPGs with square losses and square costs. This enables efficient algorithms proposed for the generalised trust region subproblem to be applied. However, such algorithms typically involve a minimum eigenvalue computation that dominates runtime. Instead, Wang et al. (2022) propose two algorithms, based on Krylov subspace and Riemannian trust region methods repectively, which improve efficiency by avoiding the explicit approximation of any eigenvalues. Whilst improving significantly upon `SDP-BISECT` since its original publication, these methods are still computationally inefficient compared to optimisation methods for ERM in the classical supervised learning setting. In other words, our results, alongside those of other authors, highlight an unavoidable computational penalty that comes with considering the actions of data providing agents.

More generally, one may hope that similar theoretical guarantees exist for related SPGs that impose a different cost function on the agents. For example, one can consider the subclass of SPGs in which the squared Euclidean distance is replaced by the L_1-distance, as discussed in Section 3.12. Recall that this leads to a completely different optimisation problem. Hence, it seems unlikely that any of our analysis throughout this chapter, which heavily exploits the hidden convexity of Problem (3.4), would be directly applicable. Put differently, the tractability of SPGs under different costs functions is a significant open question that has yet to be investigated.

Aside, from changing the cost function, one may also consider SPGs with different loss functions. For example, in many real world settings, agents are satisfied with a sufficient approximation of their preferred labelling. In this case, the preferences of each agent are better modeled by the ϵ-insensitive loss, rather than the square loss. Thus, the development of algorithms for SPGs with different loss functions is well-motivated and presents a significant technical challenge.

Chapter 4

Adversarial Blocking Bandits

In this chapter, we aim to address Problem Domain 2. More specifically, we introduce a new sequential decision making problem, called MAXREWARD, which incorporates both nonstationary rewards and blocking (as per Requirements 2a and 2b). In the MAXREWARD problem, a decision maker is tasked with sequentially pulling arms across a time horizon with the goal of maximising their cumulative reward. We assume that the rewards associated with each arm may vary throughout time, but are forced to obey a path variation budget constraint (Besbes et al., 2014). Additionally, when an arm is pulled, we assume it may be blocked for some duration. Unlike the setting of Basu et al. (2019), we assume that the blocking duration associated with each arm may change *arbitrarily* across the time horizon. In other words, the MAXREWARD problem accounts for the nonstationary nature of rewards observed in real world problems, whilst also modelling resource unavailability via blocking. The MAXREWARD problem is described fully in Section 4.1.

We consider three feedback models for the MAXREWARD problem. First, we consider the offline MAXREWARD problem, in which the decision maker has full knowledge regarding the rewards and blocking duration of each arm on every time step before the start of the time horizon. In particular, we show that computing an optimal solution to the offline MAXREWARD problem is strongly NP-hard (Section 4.2). Then, we investigate the online MAXREWARD problem (Section 4.3), wherein the decision maker is only aware of the current reward and blocking duration of each arm. Note that the aforementioned hardness result for the offline MAXREWARD setting implies that we cannot design an algorithm which is both optimal and computationally efficient for online MAXREWARD. Instead, we must design a computationally efficient algorithm with good approximation guarantees. With this goal in mind, we propose the best greedy available arm (`Greedy-BAA`) algorithm, which greedily pulls the arm with the highest current reward out of those available. In particular, we show that `Greedy-BAA` has bounded approximation ratio with respect to the optimal arm pulling policy.

Observe that both the offline and online MAXREWARD problems assume the decision maker has knowledge regarding the future rewards and blocking durations of each arm. This is typically unrealistic, and decision makers must often learn the value of each arm (and its corresponding real world resource or action) through trial and error. With this in mind, we investigate a bandit version of the MAXREWARD problem in which the decision maker is only made aware of the current reward and blocking duration of an arm if they pull it. We refer to this setting as the *adversarial blocking bandit problem* (see Section 4.4 for details). Within this feedback model, we first assume that the decision maker is aware of the path variation budget. Under this assumption, we propose the repeating greedy algorithm (`RGA`), motivated by the repeated Exp3 algorithm proposed by Besbes et al. (2014) for nonstationary bandits. Inspired by the approach of Basu et al. (2019), we adopt `Greedy-BAA` as a benchmark policy for the purpose of establishing finite-time regret guarantees. In particular, we establish a $\mathcal{O}(\sqrt{T(2\tilde{D}+K)B_T})$ finite-time regret guarantee for `RGA`, where B_T is the path variation budget and $\tilde{D}$ is the maximum blocking delay.

After this, we relax our assumption regarding the decision maker's knowledge of the path variation budget. We propose `RGA-META`, which uses Exp3 as a meta-bandit algorithm to learn an appropriate path variation budget and runs `RGA` as a subroutine. More specifically, `RGA-META` splits the time horizon into batches, and runs a new version of `RGA` within each batch. Before each batch, the Exp3 algorithm is used to select a path variation budget, which is given to `RGA` as input. We prove that `RGA-META` achieves a $\mathcal{O}((K+\tilde{D})^{1/4}\tilde{B}^{1/2}T^{3/4})$ regret bound, where $\tilde{B}$ is the maximal path variance within a single batch of the algorithm. In Section 4.5, we discuss lower regret bounds for the adversarial blocking bandit problem. In particular, we show that if the maximal blocking duration $\tilde{D}$ is constant with respect to the length of the time horizon, then there is a matching lower bound for the regret of `RGA`. Finally, in Section 4.6 we illustrate that both `RGA` and `RGA-META` can adapt to a wide variety of variation budgets popular in the literature.

4.1 Model

To begin, we introduce the MAXREWARD problem. In the MAXREWARD problem, a decision maker is tasked with selecting a sequence of actions over a time horizon of length T. At each time step $t \in [T]$, the decision maker must pull one (or none) of a set of K arms. Upon pulling arm $k \in [K]$ on time step t the decision maker receives a reward X_t^k. We denote by X_k the sequence of rewards over the time horizon T associated with arm k:

$$X^k = \{X_t^k\}_{t=1}^T.$$

In addition, we use X to denote the vector sequence describing the reward associated with each arm on every time step:

$$X = \{X^k\}_{k=1}^K.$$

If arm k is pulled on time step t, it becomes blocked, or unavailable, for the next $D_t^k - 1$ time steps. Like reward sequences, we use D^k to denote the sequence of blocking durations associated with arm k over the course of the time horizon.

$$D^k = \{D_t^k\}_{t=1}^T.$$

Similarly, we use D to denote the vector sequence describing the blocking duration associated with each arm on every time step:

$$D = \{D^k\}_{k=1}^K.$$

In our model, the rewards and blocking durations of each arm can change arbitrarily. This is in contrast to the setting of Basu et al. (2019), wherein the blocking duration associated with each arm is fixed. We let $\tilde{D}$ ($\underset{\sim}{D}$) denote the maximum (minimal) possible blocking duration, which is an upper (lower) bound on the largest (smallest) possible blocking duration. Unless explicitly stated otherwise, we will assume that $\underset{\sim}{D} = 1$. We denote by $\mathcal{D}$ the set of all possible vector sequences of blocking durations:

$$\mathcal{D} = \{\underset{\sim}{D}, \ldots, \tilde{D}\}^{K \times T}.$$

Motivated by the nonstationary bandit literature discussed in Section 2.2.2, we assume that there is a path variation budget constraint placed on the reward sequence X. More formally, the path variation associated with a reward sequence X is given by:

$$\sum_{t=1}^{T-1} \sum_{k=1}^{K} \left| X_{t+1}^k - X_t^k \right|.$$

We say that a reward sequence X is admissible if it obeys the following path variation budget constraint:

$$\sum_{t=1}^{T-1} \sum_{k=1}^{K} \left| X_{t+1}^k - X_t^k \right| \leq B_T$$

where B_T is the path variation budget over the time horizon T. We define the temporal uncertainty set $\mathcal{B}$ as the set of admissible reward sequences:

$$\mathcal{B} = \left\{ X \in [0,1]^{K \times T} \ : \ \sum_{t=1}^{T-1} \sum_{k=1}^{K} \left| X_t^k - X_{t+1}^k \right| \leq B_T \right\}$$

By setting $B_T = KT$, one can recover the setting in which reward sequences are free to vary arbitrarily. Enforcing a path variation budget constraint ensures that the rewards

associated with each arm vary in a "smooth" manner rather than rapidly or erratically. This is often the case in real world applications. For example, consider the problem of expert crowsourcing (e.g. Upwork, Outsourcely etc.). In this setting, a job requester can sequentially choose from a pool of workers and allocate short-term projects. The job requester plays the role of a decision maker, and workers correspond to arms. The performance (and thus associated reward) of a worker is typically consistent for long periods, and does not vary erratically from day to day.

However, in some real world settings, the reward associated with a given action may change radically from time step to time step. For example, consider the disaster response scenario discussed throughout this thesis. An emergency vehicle may only be required under a very specific set of circumstances. Although, when required they may be of vital importance. Such a setting is better captured by reward sequences that obey a number of changes budget (Auer et al., 2019), as discussed in Section 2.2.2. Though our current focus is on reward sequences which satisfy a path variation budget constraint, we show in Section 4.6 that our algorithms also work under different budget constraints, such as the number of changes budget (Auer et al., 2019) and maximum variation budget (Besbes et al., 2014).

As one would expect, the goal of the decision maker is to adopt a policy which maximises their cumulative reward over the time horizon. Before proceeding, we first define policies formally. Let U be a random variable defined over a probability space $(\mathbb{U}, \mathcal{U}, \mathbf{P}_u)$. A policy, $\pi = (f_1, \dots, f_T)$, is defined by a set of measurable functions of the following form:

$$
\begin{aligned}
&f_1 : \mathbb{U} \to \mathcal{K} \\
&f_t : [0,1]^{t-1} \times \{1, \dots, \tilde{D}\}^{t-1} \times \mathbb{U} \to \mathcal{K} \quad \text{for } t = 2, 3, \dots
\end{aligned}
$$

We let π_t denote the arm chosen by policy π at time t, which is given by:

$$
\pi_t = \begin{cases} f_1(U) & t = 1 \\ f_t(X_{t-1}^\pi, \dots, X_1^\pi, D_{t-1}^\pi, \dots, D_1^\pi, U) & t = 2, 3, \dots \end{cases}
$$

where X_t^π (D_t^π) denotes the reward (blocking duration) incurred by the policy π at time t. We say that a policy is admissible if it never selects an arm that is blocked. Formally, a policy π is admissible if:

$$
\pi_t \notin \{\pi_j : j + D_j^{\pi_j} - 1 \geq t, \ \forall j \leq t-1\} \quad \forall t \in [T], X \in \mathcal{B}, D \in \mathcal{D}.
$$

We use $\mathcal{P}$ to denote the set of all admissible policies. Additionally, we let $A_t(\pi)$ denote the set of available of arms at time step t under policy π. When it is clear from context, we will use A_t in place of $A_t(\pi)$ for the sake of brevity. The cumulative

reward, $r(\pi)$, of a policy π is given by:

$$r(\pi) = \sum_{t=1}^{T} X_t^{\pi}.$$

Formally, our objective is to find a policy $\pi^\star \in \mathcal{P}$ such that:

$$\pi^* \in \arg\max_{\pi \in \mathcal{P}} \mathbb{E}[r(\pi)]$$

where the expectation is over all possible randomisation coming from the policy π. We refer to this as the MAXREWARD problem. Note that the decision maker always receives bandit feedback. That is, once an arm is pulled, the blocking duration and reward associated with the chosen arm is revealed to the decision maker. In what follows, we will consider three versions of the MAXREWARD problem, in which the decision maker has access to varying levels of side information.

- The offline MAXREWARD problem - In the offline MAXREWARD problem, it is assumed that the decision maker knows the rewards and blocking durations associated with each arm in advance. In other words, the decision maker knows X and D before the start of the time horizon.
- The online MAXREWARD problem - In the online MAXREWARD problem, the decision maker does not know X and D in advance. However, at the start of time step t, both X_t^k and D_t^k are revealed to the decision maker for all values of k before an arm is pulled.
- The MAXREWARD problem - In the MAXREWARD problem, the decision maker only receives bandit feedback. From now on, we refer to this as the *adversarial blocking bandit problem*.

Before moving on, we highlight that the MAXREWARD problem is nonstochastic. That is, rewards and blocking delays are selected by an adversary who, in all cases, must adhere to a variation budget on the realized rewards at each time step. As a result, the MAXREWARD problem is not a stochastic optimisation problem. Moreover, we briefly remark on why three different versions of the MAXREWARD problem are considered. Firstly, a decision maker may have different levels of prior knowledge in different settings. The offline MAXREWARD problem corresponds to scenarios where the decision maker has full knowledge regarding the changing availability and efficacy of actions through time. Meanwhile, the adversarial blocking bandit problem corresponds to scenarios where the decision maker has no prior knowledge. Hence, by considering different versions of the MAXREWARD problem, we may design algorithms tailored to different levels of prior knowledge. Secondly, in cases where a decision maker has limited prior knowledge, it is natural to compare

their chosen policy to the policy they would have adopted with full information. Thus, to understand the efficiency of a policy selected under limited prior knowledge, one must first understand what constitutes an efficient policy in the full information setting. As we will see, computing an optimal solution to the offline MAXREWARD problem is computationally intractable. This motivates us to devise an efficient approximation algorithm for the online MAXREWARD problem which may serve as a meaningful performance benchmark for the MAXREWARD problem.

4.2 Computational Complexity of the Offline MAXREWARD Problem

First, we investigate the offline MAXREWARD problem. In particular, we show that the offline MAXREWARD problem is strongly NP-hard, even with bounded path variation budget. Of course, such a result eliminates the probability of a fully polynomial-time approximation scheme (FPTAS) for all versions of the MAXREWARD problem unless P = NP. To show that the MAXREWARD problem is strongly NP-hard, we reduce from the Boolean satisfiability problem with three literals per clause, better known as 3-SAT. It is well known that 3-SAT is strongly NP-complete (Garey and Johnson, 1979). In a 3-SAT instance, we are given a formula in conjunctive normal form consisting of n clauses built from m variables. Each clause contains at most three literals. A literal is either a variable or its negation. The goal is to determine whether there exists a truth assignment such that the formula is true. That is, to determine whether there exists a truth assignment such that each clause in the formula contains a true literal. For example, consider the following 3-SAT instance, with four variables and three clauses:

$$(v_1 \vee \neg v_2 \vee v_3) \wedge (\neg v_1 \vee v_2 \vee v_3) \wedge (\neg v_1 \vee \neg v_2 \vee \neg v_3).$$

This is a YES instance of the 3-SAT problem, as the assignment

$$(v_1, v_2, v_3) = (\text{True}, \text{True}, \text{False})$$

satisfies the formula.

Theorem 4.1. *Computing an optimal solution for the MAXREWARD problem is strongly NP-hard.*

Proof. Given a 3-SAT instance consisting of m variables, $v_1, \ldots, v_m$, and n clauses, $C_1, \ldots C_n$, we first show how to construct a corresponding instance of the MAXREWARD problem. More specifically, we consider a decision version of the

MAXREWARD problem, in which the goal is find a policy $\pi^\star \in \mathcal{P}$ such that $r(\pi^\star) \geq V$. Our construction is as follows:

- For each variable, v_j, we create two arms, k_j and $k_{\neg j}$.
- For each variable, v_j, we set $X_j^{k_j} = X_j^{k_{\neg j}} = 1$ and $D_j^{k_j} = D_j^{k_{\neg j}} = T_0$ for some T_0 that will be defined later.
- For each clause, C_i, and each literal l in C_i, we set $X_{m+i}^{k_l} = 1$.
- All remaining rewards and blocking durations are set to zero and one respectively.
- We set $T_0 = m + n$, $V = m + n$, and enforce a path variation budget $B_{T_0} = \mathcal{O}(T_0 m)$.

Next, we show that there is a solution to a 3-SAT instance if and only if there is a solution to the correspondig MAXREWARD problem. Our proceed proceeds in two parts. In the first part, we will show that if we have a solution to the 3-SAT problem, then there is a solution to the corresponding MAXREWARD problem (3-SAT $\Rightarrow$ MAXREWARD). Then, we will show that if we have a solution to the MAXREWARD problem, then there exists a solution to the corresponding 3-SAT problem (MAXREWARD $\Rightarrow$ 3-SAT).

First, suppose we have a solution to the 3-SAT problem. It follows that there is an assignment to each variable v_j such that each is clause is true. To construct a solution to the MAXREWARD problem, we design policy in two parts. First, we design a partial policy $\pi_{m+1:T_0}$ for time steps $m + 1$ onward. For each variable v_j that is set to true (false) and for each clause C_i containing v_j ($\neg v_j$), we play arm k_j ($k_{\neg j}$) at time $m + i$. That is, we set $\pi_{m+i} = k_j$ (or $\pi_{m+i} = k_{\neg j}$). If two arms are scheduled to be pulled on the same time step, we can employ an arbitrary tie-breaking rule to decide which arm to pull. From this partial policy, we observe that we obtain a cumulative reward of n, since all of the n clauses are satisfied by at least one literal.

Next, we construct a partial policy $\pi_{1:m}$ for time steps 1 to m which can be combined with the partial policy $\pi_{m+1:T}$ to create an admissible policy. If v_j is true (false), we play arm $k_{\neg j}$ (k_j) on time step j to obtain a reward of 1. That is we set $\pi_j = k_{\neg j}$ (or $\pi_j = k_j$). This partial policy obtains a reward of m as there are m variables. It is easy to see that both partial policies do not conflict with each other and can be readily combined. This is because the subset of arms pulled by both policies are disjoint. Policy $\pi_{1:m}$ only pulls arm j ($\neg j$) if v_j is false (true) and policy $\pi_{m+1:T_0}$ only pulls arm k_j ($k_{\neg j}$) if v_j is true. Clearly the combined policy obtains a cumulative reward of $n + m = V$ and is a valid solution to the MAXREWARD problem. Thus we have proved 3-SAT $\Rightarrow$ MAXREWARD.

Now, suppose we have a solution to the MAXREWARD problem. It follows, that there is a deterministic policy, $\pi^\star$, such that $r(\pi^\star) \geq V = n + m$. Note that for any admissible policy π, $r(\pi) \leq m + n$, as from time periods 1 to m we can obtain a cumulative reward of at most m, and from time steps $m + 1$ to n we can obtain a cumulative reward of at most n. To obtain a reward of m from time steps 1 to m we must play either arm k_j or $k_{\neg j}$ on time step j. To obtain a reward of n from time steps $i = m + 1, \ldots, m + n$, we must play an arm k_l corresponding to one of the literals l in the clause C_{i-m} on each time step i. Thus, if we can construct a truth assignment in which all the literals corresponding to arms pulled in time steps $m + 1$ to $m + n$ are true, then we have found a solution to the 3-SAT problem. Note any literal l whose corresponding arm is pulled in time steps $m + 1$ to $m + n$ cannot equal v_j ($\neg v_j$) if arm k_j ($k_{\neg j}$) was pulled on time step j, as otherwise k_l would be blocked for all time steps $m + 1$ to $m + n$. In other words, we have the following subset relation:

$$\{l : \exists t \geq m + 1, \pi_t^\star = k_l\} \subseteq \{v_j : \pi_j \neq k_j\} \cup \{\neg v_j : \pi_j \neq k_{\neg j}\}$$

Thus, to construct an assignment for the 3-SAT instance, we let v_j be false (true) if arm j ($\neg j$) is played at time j. This ensures that all the literals corresponding to arms pulled from time steps $m + 1$ to $m + n$ are true and thus one of the literals in each clause must be satisfied. This proves MAXREWARD $\Rightarrow$ 3-SAT and completes the theorem. □

Note that, in proof the of Theorem 4.1, a very specific subset of MAXREWARD problem instances are used to derive the hardness result. More specifically, we only consider problem instances in which both T_0 and B_{T_0} are in the range of the maximum blocking duration $\tilde{D} = n + m$. When the time horizon and path variation budget are much larger than the maximum blocking duration, one may hope that the MAXREWARD problem becomes more tractable. However, a simple argument shows that this is not the case. Let q_1 and q_2 be arbitrary integers, and consider a time horizon of length $T = k_1 k_2 T_0$. We proceed by splitting the time horizon into blocks of length $q_1 T_0$. We construct the first T_0 time steps in each block as in the proof of Theorem 4.1. For the remaining $(q_1 - 1)T_0$ time steps, we set the rewards and blocking durations of each arm to 0 and 1 respectively. It is easy to see that the proof of Theorem 4.1 is still valid, but now the time horizon is of length $T = q_1 q_2 T_0 = q_1 q_2 \tilde{D}$ and the total path variation budget is given by $B_T = q_2 \tilde{D}$. By varying q_1 and q_2 we can set up an arbitrary relationship between T, B and $\tilde{D}$. As a result, issues of computational complexity persist, even when both the time horizon and path variation budget is significantly larger than the maximal blocking duration.

Algorithm 2: Greedy best available arm (`Greedy-BAA`)

Input : T, K, X, D - An instance of MAXREWARD
Output: $\pi^+ = (\pi_1^+, \pi_2^+, ..., \pi_T^+) \in \mathcal{P}$ - A greedy solution to MAXREWARD

$\pi^+ = (\emptyset, ..., \emptyset)$;
for $t \leftarrow 1$ **to** T **do**
  Select $\pi_t^+ \in \arg\max_{k \in A_t \cup \emptyset} X_t^k$ # See the preliminary section for definitions
end
return π^+

4.3 A Greedy Algorithm for the Online MAXREWARD Problem

In this section, we consider the online version of MAXREWARD. We devise an online greedy algorithm, called Greedy Best Available Arm (`Greedy-BAA`). `Greedy-BAA` simply pulls the arm with highest reward out of those are available. Algorithm 2 provides a detailed description of `Greedy-BAA`. Below, we show that `Greedy-BAA` provides an approximation guarantee with respect to the optimal policy for offline MAXREWARD that depends on the blocking durations and the variation budget. In what follows, we let $D_{\max}^k$ ($D_{\min}^k$) denote the maximum (minimum) blocking duration associated with arm k for a given instance of the online MAXREWARD problem.

Theorem 4.2. *Let $k^\star = \arg\max_k \frac{D_{\max}^k}{D_{\min}^k}$ denote the arm with the highest max-min blocking duration ratio. Additionally, let π^+ denote the solution returned by `Greedy-BAA`, and $\pi^\star$ denote an optimal solution of the offline MAXREWARD problem. Then,*

$$\left(1 + \frac{D_{\max}^{k^\star}}{D_{\min}^{k^\star}}\right) r(\pi^+) + \frac{D_{\max}^{k^\star}}{D_{\min}^{k^\star}} B_T \geq r(\pi^\star).$$

That is, `Greedy-BAA` has an approximation ratio of $\left(1 + \frac{D_{\max}^{k^\star}}{D_{\min}^{k^\star}}\right)^{-1} \left(1 - \frac{D_{\max}^{k^\star} B_T}{D_{\min}^{k^\star} r(\pi^\star)}\right)$.

Proof. Let $\pi^\star = (\pi_1^\star, ..., \pi_T^\star) \in \arg\max_{\pi \in \mathcal{P}} r(\pi)$ be an optimal solution for the offline MAXREWARD problem. Let $\pi^+ = (\pi_1^+, ..., \pi_T^+) \in \mathcal{P}$ be the policy returned by `Greedy-BAA`.

Consider a time period t where $\pi_t^\star \neq \pi_t^+$. There are two cases in which $\pi_t^\star$ is not selected by `Greedy-BAA`. The first case is when $X_t^{\pi_t^\star} \leq X_t^{\pi_t^+}$. The second case occurs when `Greedy-BAA` has played $\pi_t^\star$ on a recent time step, say $t' < t$, which causes $\pi_t^\star$ to be blocked on time step t. In this case, note that $\pi_{t'}^+ = \pi_t^\star$. Moreover, observe that the difference between the rewards $X_{t'}^j$ and X_t^j for any arm j can be bounded in the

following manner:

$$\begin{aligned}|X^j_{t'} - X^j_t| &= |X^j_{t'} - X^j_{t'+1} + X^j_{t'+1} - X^j_t| \\ &\leq |X^j_{t'} - X^j_{t'+1}| + |X^j_{t'+1} - X^j_t| \\ &\leq \sum_{\bar{t}=t'}^{t-1} |X^j_{\bar{t}} - X^j_{\bar{t}+1}|\end{aligned}$$

where the inequalities follow from applying the triangle inequality repeatedly. Thus,

$$X^j_{t'} + \sum_{\bar{t}=t'}^{t-1} |X^j_{\bar{t}} - X^j_{\bar{t}+1}| \geq X^j_t.$$

Let $\mathrm{Blk}(t', j)$ denote the set of time periods on which j is both optimal ($\pi^\star_t = j$) and blocked under the `Greedy-BAA` algorithm, as a result of pulling arm j on time step t' ($\pi^+_{t'} = j$). Note that

$$\left|\mathrm{Blk}(t', j)\right| \leq \frac{D^j_{\max}}{D^j_{\min}}$$

where $D^j_{\max}$ and $D^j_{\min}$ are the maximum and minimum blocking duration of arm j across all the time periods, respectively. This is because, in the time steps from $t'+1$ to $t' + D^j_{\max}$, arm j can be played at most $\frac{D^j_{\max}}{D^j_{\min}}$ times by any algorithm. Hence, we can derive the following bound on the cumulative reward achieved by arm j over $\mathrm{Blk}(t', j)$:

$$\begin{aligned}\sum_{t \in \mathrm{Blk}(t',j)} X^j_t &\leq \sum_{t \in \mathrm{Blk}(t',j)} \left(X^j_{t'} + \sum_{\bar{t}=t'}^{t-1} |X^j_{\bar{t}} - X^j_{\bar{t}+1}| \right) \\ &\leq \frac{D^j_{\max}}{D^j_{\min}} \left(X^j_{t'} + \sum_{\bar{t}=t'}^{\max(\mathrm{Blk}(t',j))-1} |X^j_{\bar{t}} - X^j_{\bar{t}+1}| \right).\end{aligned}$$

Note that, for any $\bar{t} \neq t'$ such that $\pi^+_{\bar{t}} = \pi^+_{t'} = j$, $\mathrm{Blk}(t', j) \cap \mathrm{Blk}(\bar{t}, j) = \emptyset$. As a result, each arm π^+_t can be used to cover some part of the optimal solution under case 1

and/or case 2 for each time period t. It follows that

$$\begin{aligned}
r(\pi^\star) = \sum_{t=1}^{T} X_t^{\pi_t^\star} &\leq \sum_{t=1}^{T} X_t^{\pi_t^+} + \sum_{t=1}^{T} \frac{D_{\max}^{\pi_t^+}}{D_{\min}^{\pi_t^+}} \left(X_t^{\pi_t^+} + \sum_{\bar{t}=t}^{\max(\text{Blk}(t,\pi_t^+))-1} |X_{\bar{t}}^{\pi_t^+} - X_{\bar{t}+1}^{\pi_t^+}| \right) \\
&\leq \sum_{t=1}^{T} X_t^{\pi_t^+} + \frac{D_{\max}^{j^\star}}{D_{\min}^{j^\star}} \sum_{t=1}^{T} \left(X_t^{\pi_t^+} + \sum_{\bar{t}=t}^{\max(\text{Blk}(t,\pi_t^+))-1} |X_{\bar{t}}^{\pi_t^+} - X_{\bar{t}+1}^{\pi_t^+}| \right) \\
&= \sum_{t=1}^{T} X_t^{\pi_t^+} + \frac{D_{\max}^{j^\star}}{D_{\min}^{j^\star}} \left(\sum_{t=1}^{T} X_t^{\pi_t^+} + \sum_{t=1}^{T} \sum_{\bar{t}=t}^{\max(\text{Blk}(t,\pi_t^+))-1} |X_{\bar{t}}^{\pi_t^+} - X_{\bar{t}+1}^{\pi_t^+}| \right) \\
&\leq \sum_{t=1}^{T} X_t^{\pi_t^+} + \frac{D_{\max}^{j^\star}}{D_{\min}^{j^\star}} \left(\sum_{t=1}^{T} X_t^{\pi_t^+} + \sum_{i\in[K]} \sum_{t=1}^{T-1} |X_t^i - X_{t+1}^i| \right) \\
&\leq \sum_{t=1}^{T} X_t^{\pi_t^+} + \frac{D_{\max}^{j^\star}}{D_{\min}^{j^\star}} \left(\sum_{t=1}^{T} X_t^{\pi_t^+} + B_T \right) \leq \left(1 + \frac{D_{\max}^{j^\star}}{D_{\min}^{j^\star}} \right) r(\pi^+) + \frac{D_{\max}^{j^\star}}{D_{\min}^{j^\star}} B_T,
\end{aligned}$$

where the first inequality follows from applying case 1 and case 2 and the second inequality follows from replacing each blocking duration ratio with $\frac{D_{\max}^{j^\star}}{D_{\min}^{j^\star}}$, where $j^\star$ is the arm with highest max-min blocking duration ratio. The third equality follows by distributing the summations, whilst the fourth inequality follows by first grouping the time periods that each arm i is played in and then applying the sum (which ranges from 1 to $T-1$ in the worst-case). Lastly, the fifth inequality follows by definition of the total path variation budget. Rearranging the terms, we obtain our claimed result. □

Note that as $D_{\min}^{k^\star} \geq \underset{\sim}{D}$ and $D_{\max}^{k^\star} \leq \tilde{D}$, the approximation ratio above can be further bounded above by

$$\left(1 + \frac{\tilde{D}}{\underset{\sim}{D}}\right)^{-1} \left(1 - \frac{\tilde{D} B_T}{\underset{\sim}{D} r(\pi^\star)}\right).$$

When the path variation budget is zero (i.e. reward values are fixed over time) and the blocking durations per arm are homogeneous (i.e. blocking durations per arm are fixed over time), the offline MAXREWARD problem is equivalent to the offline version of the stochastic blocking bandit problem investigated by Basu et al. (2019). In this case, our proof provides an approximation ratio of $1/2$ whereas Basu et al. (2019) provides the approximation ratio $O(1 - 1/e - \mathcal{O}(1/T))$. Their technique uses a much more complicated LP-bounding technique/proof that does not directly generalize to the case of $B_T > 0$ with varying blocking durations. On the other hand, our approximation ratio result holds for the general case. For example, if B_T grows slower than $r(\pi^+)$ with T, our algorithm guarantees an approximation ratio of $(1 + 2\frac{\tilde{D}}{\underset{\sim}{D}})^{-1}$.

4.4 The Adversarial Blocking Bandit Problem

Given the investigation of the (offline and online) MAXREWARD problems in the previous section, we now turn to the main focus of this chapter, the online MAXREWARD problem with bandit feedback, a.k.a the adversarial blocking bandit problem. As discussed, in Chapter 2, regret analyses for the classical stochastic MAB setting typically benchmark against policies which repeatedly pull the same arm. Of course, in blocking settings, such policies may be inadmissible. Instead, any policy benchmark should explicitly account for the effects of blocking on arm availability. Motivated by the analysis of Basu et al. (2019) for the stochastic blocking bandit setting, we evaluate the performance of a policy against a dynamic oracle algorithm that returns an approximate solution to the offline MAXREWARD problem. In other words, we aim to minimise α-regret, where α corresponds to the approximation ratio of the dynamic oracle. More precisely, let $\pi^\star$ denote an optimal deterministic policy for the offline MAXREWARD problem. For a given, instance of the MAXREWARD problem, the α-regret of a policy $\pi \in \mathcal{P}$ against $\pi^\star$ is defined as follows:

$$R_\pi^\alpha = \alpha r(\pi^\star) - \mathbb{E}^\pi[r(\pi)]$$

where the expectation is over all possible randomisation coming from the policy π. In what follows, we will propose two algorithms for the adversarial blocking bandit setting with finite-time regret guarantees with respect to α-regret, addressing Requirement 2c. First, we assume that the decision maker is aware of the path variation budget B_T prior to pulling any arms, and develop the repeating greedy algorithm (`RGA`), which partitions the time horizon into blocks and runs `Greedy-BAA` as a subroutine. As a result, `RGA` inherits the approximation properties of `Greedy-BAA`. After this, we relax the assumption that the decision maker is aware of the path variation budget. We develop the `META-RGA` algorithm for this setting. `META-RGA` is a meta-bandit algorithm, which uses the Exp3 algorithm (Auer et al., 2002) to learn an appropriate path variation budget so that an appropriately initialised version of `RGA` can be used to achieve sublinear regret.

4.4.1 Known Path Variation Budget

Next, we describe our bandit algorithm for the adversarial blocking bandit problem, when the total path variation budget is known, called the repeated greedy algorithm (`RGA`). This algorithm can be described as follows:

1. We split the time horizon T into batches $\mathcal{T}_1, \ldots, \mathcal{T}_m$ each of size Δ_T (except possibly the last batch):

$$\mathcal{T}_j = \{t : (j-1)\Delta_T \leq t \leq \min\{j\Delta_T, T\}\} \quad \text{for all } j = 1, \ldots, m$$

Algorithm 3: Repeating Greedy Algorithm (RGA)

Input: Δ_T.

while $1 \leq j \leq \left\lceil \frac{T}{\Delta_T} \right\rceil$ **do**
  Set $\tau = 1$
  while $\tau \leq \Delta_T$ **do**
    if $(1 \leq \tau \leq K)$ **then**
      Pull arm $k = \tau \mod K + 1$
      Receive reward and blocking duration (X_τ^k, D_τ^k)
      Set $\hat{X}_t^k = X_\tau^k$ for all $t \in [1, \Delta_T]$.
    if $(K + 1 \leq \tau \leq \tilde{D} + K)$ **then**
      Pull no arms
    if $(\tilde{D} + K + 1 \leq \tau \leq \Delta_T - \tilde{D})$ **then**
      Pick arms according to
      GREEDY-BAA$(\Delta_T - 2\tilde{D} - K, K, \hat{X}^1, \ldots, \hat{X}^K, D^1, \ldots, D^K)$
    if $(\Delta_T - \tilde{D} + 1 \leq \tau \leq \Delta_T)$ **then**
      Pull no arms
    $\tau \leftarrow \tau + 1$
  $j \leftarrow j + 1$

where $m = \left\lceil \frac{T}{\Delta_T} \right\rceil$ is the number of batches.

2. Within each batch we spend the first K rounds pulling each arm. Without loss of generality, we shall assume that arm k is pulled on round k. After this we spend the next $\tilde{D}$ rounds pulling no arms. This ensures that all arms will be available when we next pull an arm.

3. Then, up until the final $\tilde{D}$ rounds we play Greedy-BAA using the rewards observed in the first K rounds as the fixed rewards for each arm.

4. In the final $\tilde{D}$ rounds of each batch, we again pull no arms. This ensures that all of the arms are available at the beginning of the next batch.

The following theorem characterises the performance of RGA. In particular, our analysis is inspired by that of Besbes et al. (2014) for nonstationary stochastic bandits, and relies upon the approximation guarantees of Greedy-BAA.

Theorem 4.3. *Suppose that the variation budget B_T and maximal blocking duration $\tilde{D} \geq 1$ are such that $\tilde{D}B_T \in o(T)$. The α-regret of RGA, where $\alpha = \frac{\tilde{D}}{\tilde{D}+\tilde{D}}$, is at most*

$$\mathcal{O}\left(\sqrt{T(2\tilde{D} + K)B_T}\right)$$

when the parameter when Δ_T is set as follows:

$$\Delta_T = \left\lceil \sqrt{\frac{(T+1)(2\tilde{D}+K)}{2B_T}} \right\rceil.$$

Proof. Let π denote the policy generated by `RGA`, and let $\pi^\star$ denote an optimal deterministic policy for the offline MAXREWARD problem. For ease of presentation, we will use $X_t^\star$ (X_t^π) to denote the reward received by $\pi^\star$ (π) on time step t. To begin our analysis, we first investigate the regret incurred by `RGA`, with respect to $\pi^\star$, within each batch.

Since rewards are bounded between zero and one, `RGA` can accumulate at most K regret in the first K time steps of each batch. Similarly, for the next $\tilde{D}$ time steps, `RGA` can accumulate at most $\tilde{D}$ regret. The same can be said for the last $\tilde{D}$ time steps within any batch. As a result, we can upper bound the α-regret incurred by `RGA` within a batch $\mathcal{T}_j$ as follows:

$$\sum_{t\in\mathcal{T}_j}(\alpha X_t^\star - X_t^\pi) \le (2\tilde{D}+K) + \sum_{t\in\mathcal{T}_j'}(\alpha X_t^\star - X_t^\pi) \tag{4.1}$$

where $\mathcal{T}_j'$ is simply $\mathcal{T}_j$ with the first $K+\tilde{D}$ and last $\tilde{D}$ time steps removed. We let $\hat{X}^k$ denote the reward received from arm k when pulled on the kth time step of batch $\mathcal{T}_j$. Likewise, we use $\hat{X}_t^\star$ ($\hat{X}_t^\pi$) to denote the reward received when the arm $\pi_t^\star$ (π_t) pulled by $\pi^\star$ (π) on time step t is pulled on the $\pi_t^\star$-th (π_t-th) time step of batch $\mathcal{T}_j$. Additionally, let B_j denote the path variance within batch $\mathcal{T}_j$:

$$B_j = \sum_{t\in\mathcal{T}_j}\sum_{k\in[K]}\left|X_{t+1}^k - X_t^k\right|$$

Then we have,

$$\begin{aligned}
\sum_{t\in\mathcal{T}_j'}(\alpha X_t^\star - X_t^\pi) &= \sum_{t\in\mathcal{T}_j'}(\alpha\hat{X}_t^\star - \hat{X}_t^\pi) + \sum_{t\in\mathcal{T}_j'}(\alpha X_t^\star - \alpha\hat{X}_t^\star) + \sum_{t\in\mathcal{T}_j'}(\hat{X}_t^\pi - X_t^\pi) \\
&\le \sum_{t\in\mathcal{T}_j'}(\alpha\hat{X}_t^\star - \hat{X}_t^\pi) + \sum_{t\in\mathcal{T}_j'}|\alpha X_t^\star - \alpha\hat{X}_t^\star| + \sum_{t\in\mathcal{T}_j'}|\hat{X}_t^\pi - X_t^\pi| \\
&\le \sum_{t\in\mathcal{T}_j'}(\alpha\hat{X}_t^\star - \hat{X}_t^\pi) + \sum_{t\in\mathcal{T}_j'}|X_t^\star - \hat{X}_t^\star| + \sum_{t\in\mathcal{T}_j'}|\hat{X}_t^\pi - X_t^\pi| \\
&\le \sum_{t\in\mathcal{T}_j'}|X_t^\star - \hat{X}_t^\star| + \sum_{t\in\mathcal{T}_j'}|X_t^\pi - \hat{X}_t^\pi| \\
&\le \sum_{t\in\mathcal{T}_j'} 2B_j \\
&\le 2\Delta_T B_j.
\end{aligned} \tag{4.2}$$

The first inequality follows from taking absolute values. The second inequality follows from the fact that $\alpha \leq 1$. The third inequality follows from the fact that `RGA` simply runs `Greedy-BAA` within batch $\mathcal{T}_j'$. Therefore, by applying Theorem 4.2 with variation budget 0, we see that the first summation on the righthand side of the second inequality must be negative. Meanwhile, the third inequality is a consequence of the following observation:

$$|X_t^k - \hat{X}^k| \leq \sum_{t,t+1 \in \mathcal{T}_j} |X_{t+1}^k - X_t^k| \leq B_j \quad \text{for all } t \in \mathcal{T}_j \text{ and } k \in [K]. \tag{4.3}$$

To see why this holds, recall that $\hat{X}^k$ is the reward received from the first pull of arm k in batch $\mathcal{T}_j$. Therefore, the difference between $\hat{X}^k$ and X_t^k is bounded by the sum of reward changes associated with arm k over the batch $\mathcal{T}_j$, which is in turn bounded by B_j. Finally, the last inequality follows from that fact that the length of a single batch is at most Δ_T.

Substituting (4.2) into (4.1) yields the following inequality:

$$\sum_{t \in \mathcal{T}_j}^{T} (X_t^\star - X_t^\pi) \leq 2\Delta_T B_j + (2\tilde{D} + K).$$

Summing over all batches we have the following bound on the α-regret:

$$\begin{aligned}
\mathcal{R}_\pi^\alpha(B_T, \tilde{D}, T) &\leq \sum_{j=1}^{m} 2\Delta_T B_j + \left\lceil \frac{T}{\Delta_T} \right\rceil (2\tilde{D} + K) \\
&\leq 2B_T\Delta_T + \left\lceil \frac{T}{\Delta_T} \right\rceil (2\tilde{D} + K) \\
&\leq 2B_T\Delta_T + \frac{T+1}{\Delta_T}(2\tilde{D} + K).
\end{aligned}$$

Since $B_T \leq TK$ by definition and both $\tilde{D}, K \geq 1$, we have

$$\sqrt{\frac{(T+1)(2\tilde{D}+K)}{2B_T}} \geq 1,$$

and thus,

$$\left\lceil \sqrt{\frac{(T+1)(2\tilde{D}+K)}{2B_T}} \right\rceil \leq \sqrt{\frac{(T+1)(2\tilde{D}+K)}{2B_T}} + 1 \leq 2\sqrt{\frac{(T+1)(2\tilde{D}+K)}{2B_T}}.$$

By setting

$$\Delta_T = \left\lceil \sqrt{\frac{(T+1)(2\tilde{D}+K)}{2B_T}} \right\rceil \leq 2\sqrt{\frac{(T+1)(2\tilde{D}+K)}{2B_T}},$$

we obtain the desired result. □

Note that this bound is sublinear in T if $\tilde{D}B_T = o(T)$. This occurs, for example, when $\tilde{D}$ is bounded above by a constant and $B_T \in o(T)$. When $\underset{\sim}{D} = 1$, note that $\alpha = \frac{1}{1+\tilde{D}}$. As a result, one may be tempted to believe that the worst-case performance of RGA is better than that of the Greedy-BAA algorithm. However, this is not the case.

From Theorem 4.2 we have that:

$$r(\pi^\star) \leq \left(1 + \frac{D^{k^\star}_{\max}}{D^{k^\star}_{\min}}\right) r(\pi^+) + \frac{D^{k^\star}_{\max}}{D^{k^\star}_{\min}} B_T \leq \left(1 + \frac{\tilde{D}}{\underset{\sim}{D}}\right) r(\pi^+) + \frac{\tilde{D}}{\underset{\sim}{D}} B_T$$

where $\pi^\star$ is an optimal deterministic policy for the offline MAXREWARD problem, and π^+ is the Greedy-BAA algorithm. This can be rewritten as:

$$\frac{\underset{\sim}{D}}{\underset{\sim}{D} + \tilde{D}} r(\pi^\star) - \frac{\tilde{D}}{\underset{\sim}{D} + \tilde{D}} B_T \leq r(\pi^+). \tag{4.4}$$

For RGA, we know from Theorem 4.3 that

$$\frac{\underset{\sim}{D}}{\underset{\sim}{D} + \tilde{D}} r(\pi^*) - \mathcal{O}\left(\sqrt{T(2\tilde{D} + K)B_T}\right) \leq r(\texttt{RGA}). \tag{4.5}$$

If $B_T = o(T)$, we have

$$\sqrt{T(2\tilde{D} + K)B_T} > B_T.$$

Thus, the lefthand side of Inequality (4.4) is larger than the lefthand side of Inequality (4.5), which implies that the approximation ratio of Greedy-BAA is still a better performance guarantee than the α-regret bound for RGA. Of course, this is what one would intuitively expect.

We now verify the theoretical performance guarantees of RGA empirically via the following simple experiment. In what follows, we construct a sequence of adversarial blocking bandit problems for time horizons of different length. In particular, we consider problem instances with $K = 10$ arms and a max delay of $\tilde{D} = 5$, where one arm has a positive reward of 1, and all remaining arms have a reward of zero. The arm with positive reward is dictated by an adversary, who periodically switches the best arm uniformly at random. We assume that the adversary will swap rewards as frequently as possible according to their variation budget. In our experiments, we set the variation budget to $T^{\frac{1}{3}}$. The delay associated with each arm on a given time step is sampled uniformly at random. For 20 logarithmically spaced values of T from 10^3 to 10^7 we compute the average reward of both Greedy-BAA and RGA on five such problem instances. The results of our experiments are shown in Figure 4.1.

Note that RGA rapidly approximates the performance of Greedy-BAA, which remains relatively constant as the time horizon lengthens. This is what one would intuitively expect, as the variation budget of the adversary is sublinear in T. Additionally,

observe that the performance of RGA is worse than Greedy-BAA for any fixed time horizon since RGA must perform uniform exploration at the start of each internal block.

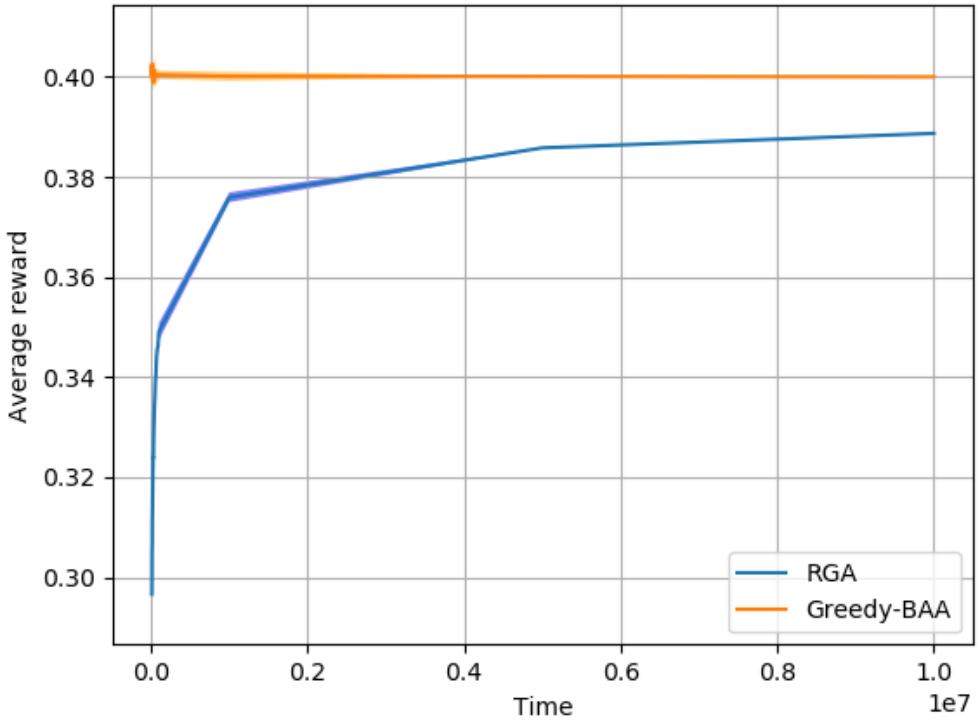

FIGURE 4.1: A performance evaluation of the RGA algorithm. We consider 20 logarithmically spaced time horizons from 10^3 to 10^7. For each horizon, we average the performance of both RGA and Greedy-BAA on five problem instances wherein the adversary periodically changes the index of the only arm with positive reward according to a variation budget of $T^{\frac{1}{3}}$.

4.4.2 Unknown Path Variation Budget

Note that RGA requires knowledge of B_T in order to properly set Δ_T. To resolve this issue, we propose META-RGA, a meta-bandit algorithm, where each arm corresponds to an instance of the RGA algorithm whose Δ_T parameter is tuned for a different total path variation budget. In META-RGA, the time horizon is broken into meta-blocks of length H. At the start of each meta-block an arm, that is an instance of RGA with its parameter tuned to a corresponding total path variation budget, is selected according to the well-known Exp3 algorithm (Auer et al., 2002). The chosen RGA algorithm is then played for the next H time steps. At the end of a meta-block, the Exp3 algorithm observes a reward corresponding to the total reward accumulated by the chosen RGA algorithm. The intuitive idea behind this approach is that, eventually, the Exp3 algorithm will learn the optimal path variation budget to use in each block.

Algorithm 4: Meta Repeating Greedy Algorithm (`META-RGA`)

Input: $T, K, \gamma \in (0,1]$, batch length H.

Initialize: $|\mathcal{J}| = \lceil \log_2(KT) \rceil + 1$, $\mathcal{J}_B = \{2^0, 2^1, \ldots, 2^{\lceil \log_2(KT) \rceil}\}$, $w_i(1) = 1$ for $i = 1, \ldots, |\mathcal{J}|$.

for $\tau = 1, \ldots, \lceil \frac{T}{H} \rceil$ **do**

Set

$$p_i(\tau) = (1-\gamma)\frac{w_i(\tau)}{\sum_{j=1}^{|\mathcal{J}|} w_j(\tau)} + \frac{\gamma}{|\mathcal{J}|} \quad i = 1, \ldots, |\mathcal{J}|$$

Draw i_τ randomly according to the probabilities $p_1(\tau), \ldots, p_{|\mathcal{J}|}(\tau)$

Run `RGA` in batch τ with budget $\mathcal{J}_B[i_\tau] = 2^{i_\tau - 1}$ and optimally tuned restarts

Receive reward $x_{i_t}(\tau) \in [0, H]$ at the end of the batch

for $j = 1, \ldots, |\mathcal{J}|$ **do**

$$\hat{x}_j(\tau) = \begin{cases} \frac{x_j(\tau)}{p_j(\tau)} & \text{if } j = i_\tau \\ 0 & \text{otherwise} \end{cases}$$

$$w_j(\tau+1) = w_j(\tau)\exp(\gamma \hat{x}_j(\tau)/(H|\mathcal{J}|))$$

In what follows, we shall denote the set of arms available to the Exp3 algorithm by $\mathcal{J}$, and denote the corresponding set of variation budgets by $\mathcal{J}_B$. The `META-RGA` algorithm uses $\lceil \log_2(KT) \rceil + 1$ meta-arms with budgets $\mathcal{J}_C = \{2^0, 2^1, \ldots, 2^{\lceil \log_2(KT) \rceil}\}$. That is, the budget values considered by the Exp3 algorithm are powers of 2 up to the smallest power which is larger than KT. In addition, let B_i denote the total path variance within batch i, and set $\tilde{B} = \max_i B_i$. The following theorem characterises the performance of `META-RGA`.

Theorem 4.4. *Suppose that the variation budget B_T and maximal blocking duration $\tilde{D} \geq 1$ are such that $\tilde{D}B_T \in o(T)$. The α-regret of `RGA-META`, where $\alpha = \frac{1}{1+\tilde{D}}$, is at most*

$$\mathcal{O}\left(\tilde{B}^{1/2} T^{3/4} (2\tilde{D} + K)^{1/4} \ln(KT)^{1/4} \ln(\ln(KT))^{1/4}\right)$$

when the parameters of `RGA-META` are set as follows:

$$H = \sqrt{\frac{T(2\tilde{D}+K)}{\ln(KT)\ln(\ln(KT))}}, \quad \gamma = \min\left\{1, \sqrt{\frac{\ln(KT)\ln(\ln(KT))}{(e-1)T}}\right\}.$$

Proof. Let π denote `META-RGA`. The α-regret of `META-RGA` can be expressed as follows:

$$\sum_{i=1}^{\lceil \frac{T}{H} \rceil} \sum_{t=(i-1)H+1}^{\max(T, iH)} (\alpha X_t^\star - X_t^\pi)$$

Let B_i denote the total path variance within batch i. Of all the RGA instances (i.e., meta-arms) available, there must be an instance who is a associated with a candidate budget $\tilde{B}$ such that:

$$\max_i B_i \leq \tilde{B} \leq 2 \max_i B_i \tag{4.6}$$

Le $\tilde{\pi}$ denote the policy of this RGA instance.

Using $\tilde{\pi}$ we can decompose the regret of META-RGA as follows:

$$\begin{aligned} &\left[\sum_{i=1}^{\lceil \frac{T}{H} \rceil} \sum_{t=(i-1)H+1}^{\max(T,iH)} (\alpha X_t^\star - X^{\tilde{\pi}})\right] \\ &+ \left[\sum_{i=1}^{\lceil \frac{T}{H} \rceil} \left(\sum_{t=(i-1)H+1}^{\max(T,iH)} X_t^{\tilde{\pi}}\right) - \left(\sum_{t=(i-1)H+1}^{\max(T,iH)} X^{\pi}\right)\right] \end{aligned} \tag{4.7}$$

Note that the RGA instance with policy $\tilde{\pi}$ might not be the best fixed meta-arm in hindsight, whose policy is denoted by π^+. Thus, we have:

$$\begin{aligned} &\left[\sum_{i=1}^{\lceil \frac{T}{H} \rceil} \left(\sum_{t=(i-1)H+1}^{\max(T,iH)} X_t^{\tilde{\pi}}\right) - \left(\sum_{t=(i-1)H+1}^{\max(T,iH)} X^{\pi}\right)\right] \leq \\ &\left[\sum_{i=1}^{\lceil \frac{T}{H} \rceil} \left(\sum_{t=(i-1)H+1}^{\max(T,iH)} X_t^{\pi^+}\right) - \left(\sum_{t=(i-1)H+1}^{\max(T,iH)} X^{\pi}\right)\right] \end{aligned}$$

The righthand side of this inequality is simply the difference between the rewards observed and accumulated by the Exp3 meta-algorithm and the best available RGA meta-arm in hindsight. Thus we can bound the second term with standard Exp3 regret bounds (Auer et al., 2002). Note that there are $\log_2(KT)$ arms available to the Exp3 algorithm, T/H is number of batches, and the maximum reward a meta-arm can receive within a batch is H (i.e., the length of each batch). Thus the second term can be bounded above by $\mathcal{O}\left(H\sqrt{T/H \ln(KT) \ln(\ln(KT))}\right) = \mathcal{O}\left(\sqrt{HT \ln(KT) \ln(\ln(KT))}\right)$.

Now we turn to bound the first term of (4.7). Each inner sum of the first term corresponds to the α-regret of policy $\tilde{\pi}$ over a block of length H. Our idea is to use Theorem 4.3 to bound the regret of $\tilde{\pi}$ in each batch i. In order to do so, we must check whether running RGA with budget $\tilde{B}$ in a batch (with time horizon H) will result in a valid value for Δ_H. That is, we must check $\Delta_H \geq 1$. From Equation (4.6) we know that $\tilde{B} \leq 2 \max_i B_i \leq 2HK$ (the second inequality comes from the definition of the total path variance budget, which is at most HK for time horizon H). Therefore, from Theorem 4.3 we know that $\Delta_H \geq \sqrt{\frac{(H+1)(2\tilde{D}+K)}{2\tilde{B}}} > \sqrt{\frac{H(2\tilde{D}+K)}{4HK}} \geq 1$ if $\tilde{D} \geq \frac{3K}{2}$. Now, since $\tilde{D}$ is an upper bound of the maximal blocking duration, we can set it to be at least $\frac{3K}{2}$ to make $\Delta_H \geq 1$. Therefore, we can apply Theorem 4.3 to each of the batches. In particular, the α-regret of $\tilde{\pi}$ over a batch i of length H using optimally tuned restarts

can be bounded as follows:

$$\begin{aligned}\sum_{t=(i-1)H+1}^{\max(T,iH)} (\alpha X_t^\star - X^{\tilde{\pi}}) &\leq \sqrt{2\tilde{B}(H+1)(2\tilde{D}+K)} \\ &\leq 2\sqrt{\tilde{B}H(2\tilde{D}+K)}.\end{aligned}$$

Summing over all blocks we have:

$$\begin{aligned}\sum_{i=1}^{\lceil \frac{T}{H} \rceil} \sum_{t=(i-1)H+1}^{\max(T,iH)} (\alpha X_t^\star - X^{\tilde{\pi}}) &\leq \left(\frac{T}{H}+1\right) 2\sqrt{\tilde{B}H(2\tilde{D}+K)} \\ &\leq 4\frac{T}{\sqrt{H}}\sqrt{\tilde{B}(2\tilde{D}+K)}.\end{aligned} \tag{4.8}$$

Combining (4.8) with the regret bound of the Exp3 meta-bandit algorithm, we get that the α-regret of `META-RGA` is at most

$$\mathcal{O}\left(\frac{T}{\sqrt{H}}\sqrt{\tilde{B}(2\tilde{D}+K)}\right) + \mathcal{O}\left(\sqrt{HT\ln(KT)\ln(\ln(KT))}\right). \tag{4.9}$$

By setting

$$H = \sqrt{\frac{T(2\tilde{D}+K)}{\ln(KT)\ln(\ln(KT))}}$$

we get the desired regret bound. □

Note that $\tilde{B} \leq HK$, since the maximum total path variance within a batch is at most HK. Thus, by setting as in the statement of Theorem 4.4, `META-RGA` can always achieve sublinear regret in T when $\tilde{D} \in \mathcal{O}(1)$. If this is not the case, then `META-RGA` requires $\tilde{B}^2\tilde{D} \in o(T)$ in order to achieve sublinear regret. Furthermore, when $\tilde{B}$ is small, our regret bound tends to $\mathcal{O}(T^{3/4})$. Whether a tighter upper bound (e.g., $\mathcal{O}(\sqrt{T})$) is possible when the variation budget is unknown is an open question.

As we did for `RGA`, we validate our theoretical results for `META-RGA` with accompanying empirical experiments. Like our previous experiments, we construct adversarial blocking bandit instances wherein there is a single arm amongst ten with positive reward, that is swapped periodically by an adversary. We set the variation budget of the adversary to both 1000 and $100 \log T$. Figure 4.2 showcases our findings. Note that convergence rate of `META-RGA` is far slower than what we observed `RGA`, even with a smaller variation budget. We attribute this to the internal Exp3 algorithm employed by `META-RGA`. As highlighted by Auer et al. (2019), the employment of meta-bandit algorithms for hyperparameter selection can lead to significant decreases in performance. We conjecture that an algorithm in the style proposed by Auer et al. (2019) may lead to both better practical and theoretical performance, but leave this as an open direction for future work.

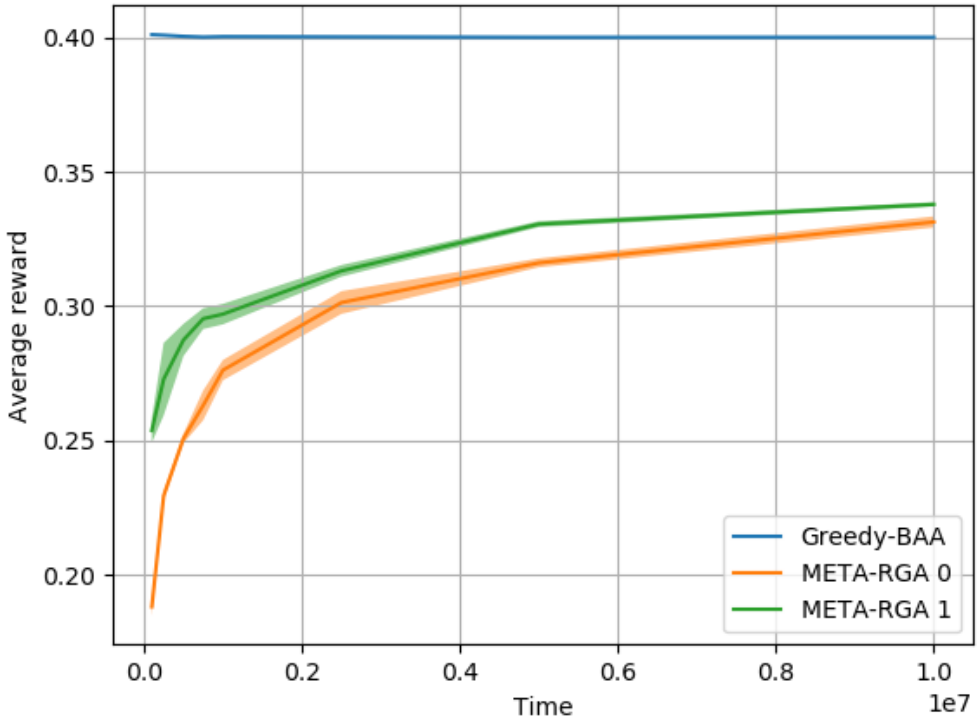

FIGURE 4.2: A performance evaluation of `META-RGA` on adversarial blocking bandit problem instances, where an adversary periodically changes the only arm with positive reward. We test on 20 time horizons logarithmically spaced between 10^3 and 10^7, taking the average of five different problem instances for each horizon. The series `META-RGA 0` corresponds to an adversary with a constant variation budget equal to 1000. Meanwhile, `META-RGA 1` corresponds to a time-varying variation budget of $100 \log T$.

4.5 Lower Bounds on Regret

In this section we provide justification as to why we require both B_T and $\tilde{D}$ to satisfy $\tilde{D}B_T \in o(T)$ in Theorems 4.3 and 4.4. In particular, we show that if either the variation budget or the maximum blocking duration is large, then the lower bound of the α-regret is $\Theta(T)$. We also discuss a potential lower bound for the α-regret of the adversarial blocking bandit problem in the case of $B_T \in o(KT)$ and $\tilde{D} \in \mathcal{O}(1)$.

Consider the case when $B_T \in \Theta(T)$. Theorem 4.3 provides a $\Theta(T)$ upper bound on α-regret when $\alpha = \frac{1}{1+\tilde{D}}$. Indeed, we show that no algorithm can do better.

Theorem 4.5. *For any $T > 0$ and $B_T \in \Theta(KT)$, there exists a sequence of rewards and blocking durations X and D such that $R_\pi^\alpha = \Theta(T)$.*

Proof. Consider the case of $B_T = KT$. This implies that the rewards can change in an arbitrary manner. Now consider the case where $D_t^k = 1$ for all $k \in [K]$ and $t \in \mathcal{T}$. Put differently, assume there is no blocking at all. In this case, we have $\alpha = 1/2$. The main idea of the proof is to randomly generate the sequences X^k and prove that, in expectation, the α-regret is large. In particular, for any arm pulling policy π we have:

$$\begin{aligned}
&\mathbb{E}\Big[\alpha \sum_t X_t^\star - \sum_t X_t^\pi\Big] = \mathbb{E}\Big[\alpha \sum_t \max_k X_t^k - \sum_t X_t^\pi\Big] \\
&\geq \mathbb{E}\Big[\alpha \sum_t \max_k X_t^k\Big] - \max_{k\in\mathcal{K}} \mathbb{E}\Big[\sum_t X_t^k\Big] + \max_k \mathbb{E}\Big[\sum_t X_t^k\Big] - \mathbb{E}\Big[\sum_t X_t^\pi\Big] \\
&\geq \alpha \sum_t \mathbb{E}\Big[\max_k X_t^k\Big] - \max_k \mathbb{E}\Big[\sum_t X_t^k\Big] + \tilde{R}_T^\pi
\end{aligned} \tag{4.10}$$

where $\tilde{R}_T^\pi$ is the pseudo-regret of π against the best fixed policy in hindsight. Now we use the standard stochastic setup to prove a lower bound of the pseudo-regret (see Bubeck and Cesa-Bianchi (2012) for technical details). That is, we draw the rewards for each arm from Bernoulli distributions with one arm set to have mean reward $\varepsilon + \sqrt{\frac{K}{T}}$, and the other arm set to have mean reward ε. By doing so, we can prove that $\tilde{R}_T^\pi \geq \frac{1}{8}\sqrt{KT}$. In addition, we have that:

$$\begin{aligned}
&\alpha \sum_t \mathbb{E}\Big[\max_{k\in\mathcal{K}} X_t^k\Big] - \max_{k\in\mathcal{K}} \mathbb{E}\Big[\sum_t X_t^k\Big] \\
&\qquad = \alpha T\Big(1 - (1-\varepsilon)^K(1 - \sqrt{K/T} - \varepsilon)\Big) - T(\sqrt{K/T} + \varepsilon) \\
&\qquad = T\Big(\alpha\Big(1 - (1-\varepsilon)^K(1 - \sqrt{K/T} - \varepsilon)\Big) - \Big(\sqrt{K/T} + \varepsilon\Big)\Big).
\end{aligned} \tag{4.11}$$

Substituting $\alpha = 1/2$ and $\beta = (1-\varepsilon)^K$ we further have:

$$\begin{aligned}
&T\Big(\alpha\Big(1 - (1-\varepsilon)^K(1 - \sqrt{K/T} - \varepsilon)\Big) - \Big(\sqrt{K/T} + \varepsilon\Big)\Big) \\
&\qquad = T\Big((1-\beta)/2 - (\sqrt{K/T} + \varepsilon)(1 - \beta/2)\Big) \\
&\qquad \geq T\Big((1-\beta)/2 - \varepsilon\Big) - \sqrt{KT}.
\end{aligned} \tag{4.12}$$

Putting everything together, we get:

$$\mathbb{E}\Big[\alpha \sum_t X_t^* - \sum_t X_t^\pi\Big] \geq T\Big((1-\beta)/2 - \varepsilon\Big) - \frac{7}{8}\sqrt{KT} \tag{4.13}$$

It is easy to show that for any $K \geq 2$, with a sufficiently small ε, there exists a constant $c > 0$ such that $(1-\beta)/2 - \varepsilon) > c$. This implies that $\mathbb{E}\Big[\alpha \sum_t X_t^* - \sum_t X_t^\pi\Big] \in \Theta(T)$, which concludes the proof. □

Next, we consider settings in which the maximal blocking duration is of order $\Theta(T)$. In this case, $\frac{1}{1+\tilde{D}} \in \Theta(1/T)$. Thus, to achieve sublinear α-regret when $\alpha = \frac{1}{1+\tilde{D}}$, a policy needs only to achieve $\Theta(1)$ cumulative reward, since the cumulative reward of

any policy is bounded above by T. As a result, problem instances in which $\tilde{D} \in \Theta(T)$ are trivial and uninteresting when $\alpha = \frac{1}{1+\tilde{D}}$. Instead we may ask what happens when α matches the approximation ratio guaranteed by `Greedy-BAA`. In this case, any policy incurs $\Theta(T)$ α-regret in the worst case.

Theorem 4.6. *For any $T > 0$ and $\tilde{D} \in \Theta(T)$, there exists a sequence of rewards and blocking durations X and D such that $R_\pi^\alpha = \Theta(T)$ for*

$$\alpha = \left(1 + \frac{D_{\max}^{k^\star}}{D_{\min}^{k^\star}}\right)^{-1} \left(1 - \frac{D_{\max}^{k^\star} B_T}{D_{\min}^{k^\star} r(\pi^\star)}\right)$$

Proof. Consider the following two problem instances, in which there are two arms. In problem instance 1, the rewards and blocking durations associated with each arm are defined as follows:

$$\begin{aligned} &X_1^1 = 1, X_1^2 = 0, D_1^1 = 1, D_1^2 = T \quad t = 1 \\ &X_t^1 = 0, X_t^2 = 1, D_t^1 = 1, D_t^2 = 1 \quad \text{for all } t \geq 2. \end{aligned}$$

In problem instance 2, the rewards and blocking durations associated with each arm are swapped. That is, the rewards and blocking durations associated with each arm are defined as follows:

$$\begin{aligned} &X_1^1 = 0, X_1^2 = 1, D_1^1 = T, D_1^2 = 1 \quad t = 1 \\ &X_t^1 = 1, X_t^2 = 0, D_t^1 = 1, D_t^2 = 1 \quad \text{for all } t \geq 2. \end{aligned}$$

In the case of problem instance 1, it is clear that `Greedy-BAA` is an optimal policy, pulling arm one on the first time step, and arm two thereafter. The total reward accumulated by this policy is T. In contrast, any policy which pulls arm two on the first time step accumulates no reward. Note that the reverse is true for problem instance 2. Any policy which pulls arm one on the first time step will achieve no reward, whilst any arm that pulls arm two on the first time step will achieve a cumulative reward of T. Additionally, note that the path variation budget for both problem instances can be as low as two. Now, consider an arbitrary policy π which pulls arm one on the first time step with probability $p \in [0, 1]$. For now, assume that $p \leq 1/2$. In this case, when π is applied to problem instance 1, its total expected reward will be $pT + (1-p)0 \leq T/2$, implying that the difference between the performance of π and that of `Greedy-BAA` is at least $T/2$. If $p > 1/2$ we may employ an analogous argument, replacing problem instance 2 with problem instance 1. □

Note that both the regret bounds we provide in Theorems 4.3 and 4.4 only hold when $\tilde{D}B_T \in o(T)$. Whilst, this assumption is well justified by Theorems 4.5 and 4.6, it is still an open question as to whether it is possible to achieve sublinear α-regret bounds in T when both B_T and $\tilde{D}$ are of order $o(T)$, but $\tilde{D}B_T \in \Omega(T)$.

Before moving on, we provide some brief discussion regarding when the α-regret bound provided by RGA is tight. Assume that $\tilde{D} = \mathcal{O}(1)$. In this case, the α-regret bound of RGA is reduces to $\mathcal{O}(\sqrt{KTB_T})$. In other words, the α-regret bound of RGA matches known lower bounds for the 1-regret for problem instances with no blocking (Auer et al., 2019). In particular, when $\tilde{D} = 1$ the Greedy-BAA algorithm becomes optimal, as observed by Basu et al. (2019), and α-regret becomes 1-regret. Therefore, any algorithm which achieves α-regret better than $\mathcal{O}(\sqrt{KTB_T})$ when the maximal blocking duration is constant with respect to time also achieves $\mathcal{O}(\sqrt{KTB_T})$ 1-regret in the standard adversarial bandit setting, without blocking. To the best of our knowledge, there are no exisitng theoretical results describing regret lower bounds for settings where the maximal blocking duration is not bounded above by a constant. Likewise, we are not aware of any lower bound results for settings where the total path variation budget is not known in advance.

4.6 Regret Analysis with Other Variation Budgets

So far, we have considered instances of the MAXREWARD problem in which reward sequences obeyed a total path variation budget constraint. This constraint on reward sequences makes sense in real world settings when the value of a given resource or action varies smoothly through time. However, in many situations, this is not the case. For example, in an emergency response scenario, the value of resources, such as emergency personnel, may vary erratically depending on whether their skills are needed at a given moment in time or not. With this in mind, we extend our regret analysis to other budgets which constrain how reward sequences may evolve through time. In particular, we show how our analysis can be adapted to the maximum variation budget, $B_T^{\max}$, and the number of changes budget, L_T, whose definitions are given below.

$$B_T^{\max} = \sum_{t,t+1 \in \mathcal{T}} \max_k |X_{t+1}^k - X_t^k|$$
$$L_T = \#\{t : 1 \leq t \leq T-1, \exists k : X_t^k \neq X_{t+1}^k\}.$$

Before analysing the regret of RGA, we first establish approximation guarantees for Greedy-BAA under each budget. Recall that, in Theorem 4.2, the approximation ratio for Greedy-BAA was calculated via the following inequality:

$$\begin{aligned} r(\pi^\star) &= \sum_{t=1}^{T} X_t^{\pi_t^\star} \\ &\leq \sum_{t=1}^{T} X_t^{\pi_t^+} + \frac{D_{\max}^{j^\star}}{D_{\min}^{j^\star}} \left(\sum_{t=1}^{T} X_t^{\pi_t^+} + \sum_{i \in [K]} \sum_{t=1}^{T-1} |X_t^i - X_{t+1}^i| \right). \end{aligned} \tag{4.14}$$

First, consider the maximum variation budget. Note that the term

$$\sum_{i\in[K]}\sum_{t=1}^{T-1}|X_t^i - X_{t+1}^i| \tag{4.15}$$

may be bounded above by $KB_T^{\max}$. In combination with Inequality (4.14), this immediately implies that

$$\begin{aligned} r(\pi^\star) &= \sum_{t=1}^{T} X_t^{\pi_t^\star} \\ &\leq \left(1 + \frac{D_{\max}^{j^\star}}{D_{\min}^{j^\star}}\right) r(\pi^+) + \frac{D_{\max}^{j^\star}}{D_{\min}^{j^\star}} KB_T^{\max}. \end{aligned}$$

As a result, we obtain the following performance guarantee for `Greedy-BAA` under the maximum variation budget.

Corollary 4.7. *Let $k^\star = \arg\max_k \frac{D_{\max}^k}{D_{\min}^k}$ denote the arm with the highest max-min blocking duration ratio. Additionally, let π^+ denote the solution returned by `Greedy-BAA`, and $\pi^\star$ denote an optimal solution to the offline MAXREWARD problem. Then,*

$$\left(1 + \frac{D_{\max}^{k^\star}}{D_{\min}^{k^\star}}\right) r(\pi^+) + \frac{D_{\max}^{k^\star}}{D_{\min}^{k^\star}} KB_T^{max} \geq r(\pi^\star).$$

That is, `Greedy-BAA` has an approximation ratio of $\left(1 + \frac{D_{\max}^{k^\star}}{D_{\min}^{k^\star}}\right)^{-1}\left(1 - \frac{D_{\max}^{k^\star} KB_T^{max}}{D_{\min}^{k^\star} r(\pi^\star)}\right)$.

Now, consider the number of changes budget. Note that (4.15) is bounded above by KL_T. Thus, we have

$$r(\pi^\star) = \sum_{t=1}^{T} X_t^{\pi_t^\star} \quad \leq \left(1 + \frac{D_{\max}^{j^\star}}{D_{\min}^{j^\star}}\right) r(\pi^+) + \frac{D_{\max}^{j^\star}}{D_{\min}^{j^\star}} KL_T.$$

As a result, we obtain the following performance guarantee for `Greedy-BAA` under the number of changes budget.

Corollary 4.8. *Let $k^\star = \arg\max_k \frac{D_{\max}^k}{D_{\min}^k}$ denote the arm with the highest max-min blocking duration ratio. Additionally, let π^+ denote the solution returned by `Greedy-BAA`, and $\pi^\star$ denote an optimal solution for the offline MAXREWARD problem. Then,*

$$\left(1 + \frac{D_{\max}^{k^\star}}{D_{\min}^{k^\star}}\right) r(\pi^+) + \frac{D_{\max}^{k^\star}}{D_{\min}^{k^\star}} KL_T \geq r(\pi^\star).$$

That is, `Greedy-BAA` has an approximation ratio of $\left(1 + \frac{D_{\max}^{k^\star}}{D_{\min}^{k^\star}}\right)^{-1}\left(1 - \frac{D_{\max}^{k^\star} KBL_T}{D_{\min}^{k^\star} r(\pi^\star)}\right)$.

Note that the corollaries above show that the approximation guarantees of `Greedy-BAA` are preserved under both the maximum variation and number of changes

budgets. Next, we show that the performance guarantees of RGA generalise in a similar manner. It is easy to show that $B_T \leq KB_T^{\max} \leq KL_T$. Thus, just by replacing B_T with $KB_T^{\max}$ and KL_T we can obtain crude regret bounds for both the maximum variation and number of changes budgets. However, we can improve upon these bounds by a $\sqrt{K}$ factor via a more intricate approach. Our analysis for both the maximum variation and number of changes budgets are largely the same as our analysis for the total path variation budget. We need only modify the way we estimate the regret in Inequality (4.3), within the proof of Theorem 4.3.

First, we will consider the maximum variation budget. Note that for all arms k and all time steps $t \in \mathcal{T}_j'$ we have

$$\begin{aligned} |X_t^k - \hat{X}_t^k| &\leq \sum_{t,t+1\in\mathcal{T}_j}^{T} |X_{t+1}^k - \hat{X}_t^k| \\ &\leq \sum_{t,t+1\in\mathcal{T}_j}^{T} \max_l |X_{t+1}^l - \hat{X}_t^l| \\ &\leq B_j^{\max}. \end{aligned} \tag{4.16}$$

where $B_j^{\max}$ is the maximum variation budget of batch $\mathcal{T}_j$. Substituting this back into Inequality (4.2) we get:

$$\begin{aligned} \sum_{t\in\mathcal{T}_j'} (\alpha X_t^\star - X_t^\pi) &= \sum_{t\in\mathcal{T}_j'} (\alpha \hat{X}_t^\star - \hat{X}_t^\pi) + \sum_{t\in\mathcal{T}_j'} (\alpha X_t^\star - \alpha \hat{X}_t^\star) + \sum_{t\in\mathcal{T}_j'} (\hat{X}_t^\pi - X_t^\pi) \\ &\leq \sum_{t\in\mathcal{T}_j'} |X_t^\star - \hat{X}_t^\star| + \sum_{t\in\mathcal{T}_j'} |X_t^\pi - \hat{X}_t^\pi| \\ &\leq \sum_{t\in\mathcal{T}_j'} 2B_j^{\max} \\ &\leq 2\Delta_T B_j^{\max} \end{aligned} \tag{4.17}$$

By imitating the proof of Theorem 4.3, we arrive at the following corollary.

Corollary 4.9. *Suppose that the maximum variation budget B_T^{max} and maximal blocking duration $\tilde{D} \geq 1$ are such that $\tilde{D}B_T^{max} \in o(T)$. The α-regret of RGA, where $\alpha = \frac{D}{\tilde{D}+\tilde{D}}$, is at most*

$$\mathcal{O}\left(\sqrt{(2\tilde{D}+K)TB_T^{\max}}\right)$$

when the parameter Δ_T is tuned as follows

$$\Delta_T = \left\lceil \sqrt{\frac{(T+1)(2\tilde{D}+K)}{2B_T^{\max}}} \right\rceil.$$

Next, we consider the number of changes budget, L_T. Note that for all arms k and all time steps $t \in \mathcal{T}_j$ we have

$$\begin{aligned} |X_t^k - \hat{X}_t^k| &\leq \sum_{t,t+1\in\mathcal{T}_j}^{T} |X_{t+1}^k - \hat{X}_t^k| \leq \sum_{t,t+1\in\mathcal{T}_j}^{T} \mathcal{I}(X_{t+1}^k \neq \hat{X}_t^k) \\ &\leq \sum_{t,t+1\in\mathcal{T}_j}^{T} \mathcal{I}(\exists l \in \mathcal{K} : X_{t+1}^l \neq \hat{X}_t^l) \\ &\leq L_j \end{aligned} \tag{4.18}$$

where $\mathcal{I}(\cdot)$ is the indicator function and L_j is the total number of changes in batch j. Once again, imitating the proof of Theorem 4.3, we obtain the following corollary.

Corollary 4.10. *Suppose that the number of changes budget L_T and maximal blocking duration $\tilde{D} \geq 1$ are such that $\tilde{D}L_T \in o(T)$. The α-regret of* RGA*, where $\alpha = \frac{D}{\tilde{D}+\tilde{D}}$, is at most*

$$\mathcal{O}\left(\sqrt{(2\tilde{D}+K)TL_T}\right)$$

when the parameter Δ_T is tuned as follows

$$\Delta_T = \left\lceil \sqrt{\frac{(T+1)(2\tilde{D}+K)}{2L_T}} \right\rceil.$$

4.7 Conclusion and Future Work

Summarising, we first introduced the MAXREWARD problem. The MAXREWARD problem features both nonstationary rewards and blocking, and as such can model realistic reward sequences (Requirement 2a) and resource unavailability (Requirement 2b). In particular, we considered three versions of the MAXREWARD problem. We first showed that the offline MAXREWARD problem is strongly NP-Hard, eliminating the possibility of an algorithm which is both tractable and optimal. After this, we examined the online MAXREWARD setting, and developed the `Greedy-BAA` algorithm. In particular, we showed that `Greedy-BAA` has provable performance guarantees with respect to the optimal policy. Lastly, we examined the adversarial blocking bandits problem. We first developed the `RGA` algorithm, which assumes the decision maker is initially aware of the total path variation budget. Additionally, we proved a finite-time α-regret guarantee for `RGA` (addressing Requirement 2c). We then relaxed our assumption that the decision maker knows the total path variation budget in advance and proposed `META-RGA`, providing α-regret guarantees in the process. Then, we discussed several lower regret bounds for the adversarial blocking bandits setting. Lastly, we showed that our algorithms generalise to several different variation budgets that are popular within the literature.

To the best of our knowledge, the adversarial blocking bandit model is the first problem setting that considers nonstationary rewards and blocking simultaneously. One can think of many other ways to simultaneously integrate blocking and nonstationary rewards into the same bandit model. For example, in the adversarial blocking bandits setting, rewards are not stochastic. In many real world settings, decision makers only receive noisy feedback. As a result, stochastic bandit models which simultaneously integrate nonstationary rewards and blocking are of significant interest. In the case where blocking durations are fixed and deterministic and rewards are nonstationary and stochastic, we conjecture that a combination of the UCB algorithm for stochastic blocking bandits proposed by Basu et al. (2019) and sliding window UCB (Garivier and Moulines, 2011) may provide finite-time α-regret guarantees, but leave this as a direction for future work.

Adversarial blocking bandits also assume that the blocking durations associated with each arm can be arbitrary. Whilst incredibly general, this assumption may be too strong for a number of real world settings. For example, consider a setting in which taking an action corresponds to committing to a manufacturing cycle. The underlying manufacturing process may rely on scarce resources, such as gold, whose supply varies smoothly over time. It is reasonable to expect the amount of time it takes gather enough resources for new manufacturing cycle to evolve over time based on the supply of the underlying scarce resource. In such scenarios, it makes sense to constrain the blocking durations associated with each arm so that they obey a variation budget. For example, one may constrain blocking duration sequences according to a number of changes budget, just as Auer et al. (2019) does for reward sequences. Such models present avenues for future research.

Chapter 5

Sequential Blocked Matching

Next, we shift focus to Problem Domain 3. More specifically, we introduce the sequential blocked matching (SBM) problem, in which a central planner must repeatedly construct matchings between agents and services over a number of time steps. In contrast to existing models for repeated matching, the sequential blocked matching problem models resource unavailability directly through blocking. Once a service is assigned to an agent within a matching, the service is blocked for a fixed duration, depending on which agent it what assigned to. The sequential blocked matching problem is described formally in Section 5.1.

As a starting point for theoretical analysis, we first investigate an offline version of sequential blocked matching, in which each agent is fully aware of their own preferences (Section 5.2). Working under the assumption that agents hold cardinal preferences, but report information ordinally, we evaluate the performance of the decision maker via distortion (Procaccia and Rosenschein, 2006), a natural metric which accounts for the unavoidable loss in utility due to ordinal reporting. In particular, we provide lower bounds on the distortion of both deterministic and randomised matching policies. After this, we illustrate that trivial approaches, which repeatedly apply a truthful single shot one-sided matching algorithm on each time step, are not truthful in the sequential blocked matching setting. Therefore, we design an extension of RSD, called repeated RSD (`RRSD`). We show that `RRSD` has bounded incentive ratio, and therefore satisfies a relaxed notion of truthfulness (addressing Requirement 3b). Immediately after this, we show that `RRSD` is asymptotically optimal in terms of distortion (addressing Requirement 3a). Then, we show that `RRSD` can be derandomised to obtain a deterministic algorithm with good distortion guarantees.

In Section 5.3, we investigate a bandit version of the sequential blocked matching problem in which agents are initially unaware of their preferences over services, and must learn them over time. We assume that each agent receives stochastic feedback about a service only when matched to it. Within this setting, we develop a bandit

version of RRSD, called bandit RRSD (BRRSD), based on the ETC paradigm. To analyse BRRSD, we use two regret notions. We evaluate the efficiency of BRRSD via the α-dynamic regret, which benchmarks the decision maker against an α-approximation of the optimal matching sequence. To evaluate the truthfulness of BRRSD, we formalise the possible behaviours of each agent via misreporting policies. We then introduce the α-incentive compatible regret, which benchmarks the misreporting policy of each agent against an α-approximation of the best misreporting policy in hindsight. In Section 5.4, we show that BRRSD achieves sublinear dynamic α-regret, whilst also ensuring that each the α-incentive compatible regret of each agent is sublinear when they report truthfully (addressing Requirement 3c).

5.1 Preliminaries

In this section, we introduce the sequential blocked matching problem in detail. We start by describing how the preferences of each agent are modeled and how agents are matched to services in a single time step. After this, we introduce the first version of SBM that we study in this chapter, which features the blocking of services when they are assigned to agents.

In our model, we have a set of agents, $N = \{1, \dots, n\}$, who hold cardinal preferences over a set of services, $S = \{1, \dots, s\}$, where $s \gg n$ [1]. We use $\mu_{i,j} \in \mathbb{R}^+$ to describe the cardinal reward agent i receives for being assigned service j. Similarly, we denote by $\mu_i = (\mu)_{j=1}^{s}$ the vector of rewards associated with agent i. In what follows, we will also refer to μ_i as the utilities associated with agent i. Moreover, we restrict ourselves to utilities which lie in the probability simplex. That is, we assume $\mu_i \in \Delta^{s-1}$ for all $i \in N$. In other words, we make a unit-sum assumption about the utilities of each agent. Bounding constraints on utilities are common in the ordinal one-sided matching literature (Filos-Ratsikas et al., 2014), and are typically required in order to prove lower bounds for truthful algorithms such as RSD. Moreover, the unit-sum assumption is prevalent in social choice theory (Boutilier et al., 2015). Lastly, we denote by μ the n by s matrix of rewards, which we refer to as the reward profile of the agents.

We say that agent i (weakly) prefers service a to service b if agent i receives greater reward by being assigned service a over service b. That is, agent i prefers service a over service b if and only if $\mu_{i,a} \geq \mu_{i,b}$. We use the standard notation $a \succ_i b$ to say that agent i prefers service a to service b. That is, we write $\succ_i$ to denote the linear preference ordering induced by agent i's reward profile [2]. Additionally, we use the

[1] Note that this is without loss of generality, as we may always add dummy services corresponding to a null assignment.

[2] One reward profile may induce many linear orderings. However, the linear preference profile induced by a reward profile can be made unique via tie-breaking rules.

notation $\succ_i(j)$ to indicate the service in the jth position of the preference ordering $\succ_i$. More generally, we use the notation $\mu_{\text{ind}} \rhd \succ_{\text{ind}}$ to denote that $\succ_{\text{ind}}$ is a preference ordering induced by the reward profile μ_{ind}. We let $\mathcal{P}(S)$, or $\mathcal{P}$ for short, denote the set of all linear preference orderings over S. Furthermore, we let $\succ = (\succ)_{i=1}^{n} \in \mathcal{P}^n$ denote the preference profile of the agents. As is standard, we write $\succ_{-i}$ to denote $(\succ_1, \ldots, \succ_{i-1}, \succ_{i+1}, \ldots, \succ_n)$. Likewise, we may write $\succ$ as $(\succ_i, \succ_{-i})$ at times for ease of presentation.

A matching $m : N \to S \cup \{0\}$ is a mapping from agents to services. We let $m(i)$ denote the service allocated to agent i by the matching m. We use 0 to denote the null assignment. That is, agent i is assigned no service in a matching if $m(i) = 0$. We let $\emptyset$ denote the null matching, in which no agent is assigned a service. We say matching is feasible if no two agents are mapped to the same service. We let $\mathcal{M}$ denote the set of all feasible matchings.

In what follows, we consider discrete-time sequential decision problems, wherein a planner must select a sequence of (feasible) matchings over T time steps. We let m_t denote the matching chosen by the planner at time step t, and denote by $M = (m_t)_{t=1}^{T}$ a sequence of T matchings. We denote by $M(t, i) = m_t(i)$ the service matched to agent i at time t.

Furthermore, we assume that, when a service is assigned, it may be blocked for a time period depending on the agent it was assigned to. More specifically, when agent i is matched with service j, we assume that service j cannot be matched to any agent for the next $D_{i,j} - 1$ time steps. We refer to $D_{i,j}$ as the blocking duration associated with the agent-service pair i and j. Additionally, we let $\tilde{D}$ denote the maximal blocking duration possible, and let D denote the n by s matrix of blocking durations. From now on, we assume that all blocking durations are known a priori by both the planner and all agents.

We say that a matching sequence M is feasible with respect to the blocking duration matrix D if no service is matched to an agent on a time step where it has been blocked by a previous matching.

Definition 5.1. For a given blocking duration matrix D, the *set of feasible matching sequences* of length T, $\mathcal{M}_T^D \subseteq \mathcal{M}_T$, is the set of all matching sequences $M \in \mathcal{M}_T$ such that for all $t \in \{1, \ldots, T\}$, $i \in N$, and $j \in S$, if $M(t, i) = j$ then $M(t', i') \neq j$ for all $i' \in N$ and for all t' such that $t < t' \leq t + D_{i,j} - 1$.

In other words, we say that a matching sequence is feasible if there is no matching in the sequence which assigns an agent a service which has been blocked by a previous matching. Note that blocking of services is a common phenomenon in real-world scenarios. For example, consider a setting in which each service corresponds to a freelance contractor, and each agent corresponds to an employer. The matching of

services and agents then corresponds to employers contracting freelancers. For the duration of the contract, which may differ from employer to employer, the matched freelancer is unavailable before returning to the pool of available services once their contract ends.

We define the welfare, $\mathrm{W}_i(M, \mu_i)$, agent i receives from a matching sequence M as the sum of rewards it receives from each matching in the sequence. That is,

$$\mathrm{W}_i(M, \mu_i) = \sum_{t=1}^{T} \mu_{i,M(t,i)}.$$

Similarly, we define the social welfare, $\mathrm{SW}(M, \mu)$, of a matching sequence M as the summation of agent welfare. More specifically,

$$\mathrm{SW}(M, \mu) = \sum_{i=1}^{n} W_i(M, \mu_i).$$

Next, we describe the first sequential matching setting we consider in this chapter, which we call the offline SBM setting. In this setting, the planner must produce a feasible matching sequence of length T. Prior to the selection of a matching sequence, each agent submits a linear preference ordering to the planner. We denote by $\succ_i$ the preference ordering, or report, submitted by agent i. Analogously, we denote by $\succ$ the preference profile submitted cumulatively by the agents, and call it the report profile. A matching policy $\pi(M \mid \succ, D)$ assigns a probability of returning a matching sequence M given a submitted report profile $\succ$ and blocking duration matrix D. When it is clear from context, we will abuse notation and use $\pi(\succ, D)$ to refer to the (random) matching sequence prescribed by a policy π given a report profile $\succ$ and blocking duration matrix D.

We say that a matching policy is admissible if, for all possible report profiles and blocking duration matrices, the matching sequence returned by the policy is always feasible. The goal of the planner is to adopt an admissible matching policy which achieves high social welfare in expectation relative to the best feasible matching sequence in hindsight, $M^\star(\mu, D)$,

$$M^\star(\mu, D) = \arg \max_{M \in \mathcal{M}_T^D} \mathrm{SW}(M, \mu).$$

We assume that each agent, with full knowledge of the matching policy employed the planner, submits a linear preference ordering with the intention of maximising their own welfare, and therefore may try to manipulate the planner by submitting a preference ordering which is not induced by their underlying cardinal preferences. We say that an agent is truthful if they submit a preference ordering induced by their underlying cardinal preferences. That is, an agent is truthful if $\mu_i \rhd \succ_i$. We denote by $\succ_i^\star$ the report which maximises agent i's welfare in expectation under the assumption

that all other agents are truthful. We say that a policy is truthful if, for all possible μ and D, it is optimal for each agent to be truthful if all other agents are truthful. In other words, a policy is truthful if, for all μ and D, we have that $\mu_i \rhd \succ_i^\star$ for all $i \in N$.

To evaluate the efficiency of a given policy we use distortion, a standard notion of approximation for settings with ordinal preferences.

Definition 5.2. The distortion of a matching policy π is the worst-case ratio between the expected social welfare of the matching sequence $\pi(\succ, D)$, returned by the policy π when all agents report truthfully, and the social welfare of the optimal matching sequence, $M^\star(\mu, D)$:

$$\sup_{\mu,D} \frac{\mathrm{SW}(M^\star(\mu, D), \mu)}{\mathbb{E}\left[\mathrm{SW}(\pi(\succ, D), \mu)\right]}.$$

In other words, the distortion of a policy π is simply the approximation ratio of the policy π with respect to the policy that always returns the optimal matching sequence. In addition, note that the distortion is only a useful measure of a matching policy's efficiency if said policy encourages truthful reporting. For truthful policies, distortion is completely characterising of a policy's expected performance. As a result, we not only seek policies which have low distortion, but also policies which incentivise agents to submit truthful reports. To this end, we introduce the notion of incentive ratio, which measures the relative improvement in welfare an agent can achieve by lying about their preferences.

Definition 5.3. The incentive ratio $\zeta(\pi) \in \mathbb{R}_+$ of a matching policy π is given by:

$$\zeta(\pi) = \max_{D, \succ_{-i}, \mu_i \rhd \succ_i} \frac{\mathbb{E}[\mathrm{W}_i(\pi((\succ_i, \succ_{-i}), D), \mu_i)]}{\mathbb{E}[\mathrm{W}_i(\pi((\succ_i^*, \succ_{-i}), D), \mu_i)]}.$$

If a policy has an incentive ratio of 1, then it is truthful. There are many reasons that we may expect a policy with bounded incentive ratio to do well. A bounded incentive ratio implies truth telling is a good approximation to each agent's optimal report. If computing the optimal report is computationally intractable for the agent, being truthful is therefore an attractive alternative, especially if the approximation ratio implied by the incentive ratio is tight. In summary, we seek matching policies with good guarantees when it comes to both incentive ratio and distortion. This topic is treated in detail in the forthcoming sections.

5.2 The Offline SBM Setting

In this section, we present our analysis of the offline SBM setting. We first provide a lower bound on the distortion achievable by both randomised and deterministic

policies. Then, we discuss why trivial extensions of truthful single-shot matching algorithms do not result in truthful policies. Instead, we focus on designing policies which use truthful single-shot matching mechanisms as a basis, and have bounded incentive ratio. More precisely, we present the RRSD algorithm. We show that the incentive ratio of RRSD is bounded below by $1 - 1/e$, and provide upper bounds on the distortion achieved by RRSD, which match our previously established lower bounds on the best distortion achievable by any randomised algorithm.

5.2.1 Lower Bounds on the Distortion of Matching Policies

First, we prove that the distortion of any deterministic policy is $\Omega(s)$. That is, the distortion of any deterministic policy scales linearly with the number of services in the best case. In the proof, we first carefully construct a set of ordinal preferences. Then, given any matching sequence M, we show that there exists a reward profile which induces the aforementioned ordinal preferences and on which M incurs distortion of order $\Omega(s)$.

Theorem 5.4. *The distortion of any deterministic policy is* $\Omega(s)$.

Proof. We now consider an instance of SBM with n agents and n services, where each agent has the same preferences. That is, $\succ_a = \succ_b$ for all $(a, b) \in N$. Furthermore, assume that, without loss of generality, service j is in the jth position of this preference ordering. That is, assume that $\succ_a(j) = j$ for all $j \in S$. Lastly, assume that the blocking duration on each of the n services is $\tilde{D}$ for all agents. In other words, put more formally, assume that $D_{ij} = \tilde{D}$ for all $i \in N$ and all $j \in S$.

We proceed with the proof in the following manner. Given the matching sequence M returned by a deterministic policy using the above preference profile and blocking duration matrix, we will show that there exists a set of reward profiles which induce the preference profile, and on which the matching sequence M suffers a distortion of $\Omega(n) = \Omega(s)$. We construct this reward profile via an inductive argument.

Firstly, observe that there must exist some agent, $i_1 \in N$, who is assigned service 1 at most $T/\tilde{D}n$ times in the matching sequence M by the pigeonhole principle. We set the reward profile of agent i_1 to $(1, 0, \ldots, 0)$. Disregarding agent i_1, observe that there must exist a different agent, $i_2 \in N$, who is assigned service 1 or 2 at most $T/\tilde{D}(n-1)$ times, once again by the pigeonhole principle. We set the utilities of agent i_2 to $(1/2, 1/2, 0, \ldots, 0)$. Disregarding both agents i_1 i_2, we can find a new agent, $i_3 \in N$, who has been assigned services 1, 2 or 3 at most $T/\tilde{D}(n-2)$ times. We set the utilities of agent i_3 to $(1/3, 1/3, 1/3, 0 \ldots, 0)$. We proceed in this pattern for a total of n steps, until all agents are assigned utilities. Note that the resulting reward profile

constructed induces the desired preference profile (assuming that a numeric tie-breaking rule is used).

Given the assigned utilities, it is obvious that an optimal matching sequence assigns service j to agent i_j whenever the service is available. The social welfare of this optimal matching sequence is therefore of order $\Theta(\log(n)T/\tilde{D})$. In contrast, the matching sequence M has social welfare of order $\Theta(\log(n)T/\tilde{D}n)$. As we can always construct such a reward profile, no matter the matching sequence M returned by a policy, this implies that the distortion of any policy is of order $\Omega(n) = \Omega(s)$. □

Next, we prove that the distortion incurred by any randomised policy is $\Omega(\sqrt{s})$. To prove this, we first show that it is sufficient to consider only anonymous policies, which ensure that an agent is treated the same if its relative position, or indexing, amongst other agents changes. Then, we construct a reward profile which yields the desired distortion for all anonymous policies.

Theorem 5.5. *The distortion of the best randomised policy is* $\Omega(\sqrt{s})$.

Proof. Similarly to Theorem 5.4, we consider an instance of SBM with n agents and n services. Additionally, assume that the blocking durations for all services is the same for all agents. That is, $D_{ij} = d$ for some $d \leq \tilde{D}$.

Before moving to the content of the proof, we first show that it is sufficient to consider only anonymous policies. Given a preference profile $\succ$, we let $A_{ij}(\succ) \in \{0, 1, \ldots, T\}$ denote the random variable that indicates the number of times agent i was allocated service j. We call a randomised matching algorithm anonymous if $\mathbb{E}[A_{ij}(\succ_1, \ldots, \succ_n)] = \mathbb{E}[A_{\sigma(i)j}(\succ_{\sigma(1)}, \ldots, \succ_{\sigma(n)})]$ for all permutations σ. In other words, a policy is anonymous if an agent's relative position with regards to other agents does not affect its assignment of services in expectation.

Now, suppose we are given a matching policy which has distortion at most ρ i.e. $\sum_{ij} \mu_{ij}\mathbb{E}[A_{ij}(\succ)] \geq \rho\text{OPT}(\mu)$. We can consider a new matching policy that selects a permutation σ uniformly at random and then applies the same policy on the input $\succ_\sigma = (\succ_{\sigma(1)}, \ldots, \succ_{\sigma(n)})$. Then, the expected social welfare of the new policy is

$$\mathbb{E}_\sigma\left[\sum_{ij} \mu_{\sigma(i)j} A_{\sigma(i)j}(\succ_\sigma)\right] \geq \mathbb{E}_\sigma\left[\rho\text{OPT}(\mu_\sigma)\right] = \rho\text{OPT}(\mu)$$

The first inequality follows because the original policy gives ρ distortion even when applied to the profile μ_σ, and the second equality follows because the optimal welfare ($\text{OPT}(\mu) = \sum_{ij} \mu_{ij} A^\star_{ij}$) is invariant to permutation. Therefore, the new anonymous policy has distortion at most ρ. This implies that, for any matching policy, there is an anonymous matching policy with identical performance with respect to distortion. As

a result, from now on, we restrict our consideration to anonymous matching policies without loss of generality.

Next we will show that any anonymous matching policy incurs distortion of order $\Omega(\sqrt{s})$ via construction of a special set of reward profiles. The reward profile we construct is very similar to the one constructed in the proof of Lemma 8 of Filos-Ratsikas et al. (2014). In what follows we will assume that n is a square number, for ease of presentation. For each $i \in [\sqrt{n}]$, define

$$\mu_{i,j} = \begin{cases} 1 - \sum_{j \neq i} \mu_{i,j} & \text{if } j = i \\ \frac{n-j}{10n^3 d} & \text{o.w.} \end{cases}$$

And for each $\ell \in [\sqrt{n} - 1]$, define

$$\mu_{i+\ell\sqrt{n},j} = \begin{cases} 1 - \sum_{j \neq i} \mu_{i,j} & \text{if } j = i \\ \frac{1}{\sqrt{n}} - \frac{j}{10n^2} & \text{if } j \neq i \ \ \& \ \ j \leq \sqrt{n} \\ \frac{n-j}{10n^3 d} & \text{o.w.} \end{cases}$$

In other words, we partition the n agents into $\sqrt{n}$ groups, where all agents in group i have the same preference order. Let $G_i = \{i\} \cup \left\{i + \ell\sqrt{n} : \ell = 1, \ldots, \sqrt{n} - 1\right\}$ denote such a group. Observe that all the agents in group G_i have preference order $i \succ 1 \succ \ldots \succ i-1 \succ i+1 \succ \ldots \succ n$. Therefore, for any service j, all the agents in group G_i have the same expected number of assignments, due to anonymity of the matching policy. Let us call this number of assignments T_{ij}. Since any service j can be allocated at most T/d times we have

$$\sum_{i=1}^{\sqrt{n}} \sum_{p \in G_i} T_{ij} \leq \frac{T}{d} \ \Rightarrow \sum_{i=1}^{\sqrt{n}} T_{ij} \leq \frac{T}{d\sqrt{n}}. \tag{5.1}$$

For any agent $i \in [\sqrt{n}]$, the maximum expected welfare over T rounds is bounded above as follows:

$$T_{ii} + \sum_{j \neq i} T_{ij} \frac{n-j}{10n^3 d} \leq T_{ii} + \mathcal{O}\left(\frac{T}{nd}\right).$$

Now consider an agent $i + \ell\sqrt{n}$ for $\ell \in [\sqrt{n} - 1]$. Such an agent's welfare over the T rounds is at most

$$T_{ii}\mathcal{O}\left(\frac{1}{\sqrt{n}}\right) + \sum_{j \neq i, j \leq \sqrt{n}} T_{ij} \frac{1}{\sqrt{n}} + \sum_{j > \sqrt{n}} T_{ij} \frac{n-j}{10n^3 d} \leq \mathcal{O}\left(\frac{1}{\sqrt{n}}\right) \sum_{j=1}^{\sqrt{n}} T_{ij} + \mathcal{O}\left(\frac{T}{nd}\right).$$

Therefore, the social welfare is bounded by

$$\begin{aligned}
&\sum_{i=1}^{\sqrt{n}} T_{ii} + \mathcal{O}\left(\frac{1}{\sqrt{n}}\right) \sum_{i=1}^{\sqrt{n}} \sum_{\ell=1}^{\sqrt{n}-1} \sum_{j=1}^{\sqrt{n}} T_{ij} + \mathcal{O}\left(\frac{T}{d}\right) \\
&\leq \sum_{i=1}^{\sqrt{n}} T_{ii} + \sum_{i=1}^{\sqrt{n}} \sum_{j=1}^{\sqrt{n}} T_{ij} + \mathcal{O}\left(\frac{T}{d}\right) \\
&\leq 2 \sum_{j=1}^{\sqrt{n}} \sum_{i=1}^{\sqrt{n}} T_{ij} + \mathcal{O}\left(\frac{T}{d}\right) \\
&\leq 2 \sum_{j=1}^{\sqrt{n}} \frac{T}{d\sqrt{n}} + \mathcal{O}\left(\frac{T}{d}\right) = \mathcal{O}\left(\frac{T}{d}\right)
\end{aligned}$$

where the last line follows from (5.1). On the other hand, any deterministic and non-anonymous allocation rule that always assigns service i to agent i every d rounds achieves a social welfare of at least $\sqrt{n}\frac{T}{d}(1 - \frac{1}{10nd})$. This establishes a bound of $\Omega(\sqrt{n}) = \Omega(\sqrt{s})$ on distortion. □

5.2.2 Constructing Truthful Algorithms for the Offline SBM Setting

As previously mentioned, we assume that agents submit reports with the intention of maximising their own welfare. As a result, the distortion incurred by a policy may not reflect its performance in practice, as agents may be incentivised to misreport their preferences to increase their welfare. In standard single-shot one-sided matching problems, this issue is sidestepped via the employment of truthful policies, like RSD. In a similar way, we would like to develop truthful algorithms for the offline SBM setting.

One may be tempted to apply such truthful single-shot policies to our setting directly. That is, to apply an algorithm such as RSD repeatedly on every time step to devise a matching sequence. This intuition is correct when there is no blocking, as the matching problems for each time step are then independent of each other. However, with blocking, the matchings from previous time steps will have a substantial effect on the set of matchings which preserve the feasibility of the matching sequence in future rounds. As a result, immediately obvious approaches, such as matching according to RSD repeatedly, do not result in truthful policies.

For example, consider a problem instance involving three agents, $\{1, 2, 3\}$ and three services, $\{a, b, c\}$, in which the ordinal preferences of each agent are defined as follows

$$\begin{aligned}
1 &: a \succ b \succ c \\
2 &: b \succ c \succ a \\
3 &: b \succ c \succ a
\end{aligned}$$

and the blocking duration matrix is given by

$$D = \begin{pmatrix} 2 & 1 & 1 \\ 2 & 2 & 2 \\ 1 & 1 & 1 \end{pmatrix}.$$

Now, consider an algorithm in which a permutation, σ, of the agents is sampled uniformly at random, and RSD is run repeatedly on each time step in accordance with σ. In particular, consider the case where $\sigma = (1, 2, 3)$. Table 5.1 shows the assignment of services when each agent reports their true preferences to the algorithm. Note that agent 1 receives service a, followed by service c, repeatedly for the entire time horizon. Meanwhile, Table 5.2 shows the assignment of services when agent 1 lies and reports the preference ordering $b \succ a \succ c$. Note that in this case, agent 1 receives service b followed by service a, repeatedly for the entire time horizon. Since agent 1 prefers service b over service c, agent 1 prefers the assignment of services it receives from misreporting when $\sigma = (1, 2, 3)$. Moreover, it is easy to verify that misreporting $b \succ a \succ c$, does not worsen agent 1's welfare when any other permutation is sampled. Therefore, we conclude that agent 1 can benefit by misreporting the preference ordering $b \succ a \succ c$. Hence, we conclude that the algorithm proposed is not truthful.

	1	2	3	4
1 : $a \succ b \succ c$	a	c	a	c
2 : $b \succ c \succ a$	b		b	
3 : $b \succ c \succ a$	c		c	

	1	2	3	4
a	1	✗	1	✗
b	2	✗	2	✗
c	3	1	3	1

TABLE 5.1: The assignment of services under permutation σ for the first four time steps when each agent reports their true preferences. The left table describes the sequence of services assigned to each agent, whilst the right table describes the sequence of agents assigned to each service. We use ✗ to denote that a given service is blocked on the corresponding time step.

	1	2	3	4
1 : $a \succ b \succ c$	b	a	b	a
2 : $b \succ c \succ a$	c	c	c	c
3 : $b \succ c \succ a$	a			

	1	2	3	4
a	3	1	✗	1
b	1	✗	1	✗
c	2	2	2	2

TABLE 5.2: The assignment of services under permutation σ for the first four time steps when agents 2 and 3 report their preferences truthfully, and agent 1 misreports $b \succ a \succ c$. The left table describes the sequence of services assigned to each agent, whilst the right table describes the sequence of agents assigned to each service. We use ✗ to denote that a given service is blocked on the corresponding time step.

Observe that this example illustrates SBM's close connection to scheduling problems. If agent 1 takes service a on the first round, then service b will be blocked and unavailable in the next. As a result, it is better for agent 1 to secure service b as quickly as possible before it becomes blocked by other agents, and pursue service a later. Intuitively, assuming that all other agents are truthful, one can view an agent's report as selecting a certain (potentially randomised) schedule of service assignments. For an

algorithm to be truthful, truthful reporting must return the best (expected) schedule of assignments for the agent out of those offered by the algorithm, given the preferences of other agents. For an algorithm to be optimal in terms of distortion, the agent needs a wide range of schedules to choose from. However, as the number of offered schedules increases, the more difficult it becomes to identify the optimal schedule, and thus ensure truthfulness, as the problem begins to represent a full-blown scheduling problem. Hence, there is an implicit trade-off between the distortion of any algorithm and its guarantees with respect to truthfulness.

One simple way of generating truthful policies is to run a truthful single-shot one-sided matching policy once every $\tilde{D}$ time steps and simply return the empty matching in the remaining time steps. Such an approach decouples each time step from the next, resulting in truthfulness, but comes at the cost of only matching in at most $\lceil T/\tilde{D} \rceil$ rounds. Instead, we construct an algorithm for the offline SBM setting from truthful single-shot matching algorithms in a different manner. More specifically, we propose the repeated random serial dictatorship (RRSD) algorithm, which uses RSD as a basis. Whilst RRSD is not truthful, it does have bounded incentive ratio.

5.2.3 A Greedy Algorithm for the Offline SBM Setting

The RRSD algorithm slowly builds up a matching sequence M over time by iterating through agents and services. In other words, RRSD begins with the empty matching sequence, where $M(t, i) = 0$ for all t and $i \in N$. To begin, RRSD samples a permutation of agents, σ, uniformly at random. Next, RRSD iterates through the agents in the order given by the permutation sampled. For each agent i, RRSD iterates through services in the order specified by the preference ordering $\succ_i$ reported by agent i. For a given service j, RRSD repeatedly assigns service j to agent i at the earliest time step which does not cause the matching sequence to become infeasible. When no such time step exists, RRSD moves onto the next service in agent i's preference ordering. Once RRSD has iterated through the entire preference ordering of agent i, RRSD moves onto the next agent in the permutation σ and repeats this process until the end of the permutation is reached. The pseudocode for RRSD is given in Algorithm 5.

We now briefly describe the intuition behind RRSD. In essence, RRSD attempts to mimic the RSD algorithm for single-shot matching problems by allowing each agent to sequentially choose a feasible assignment of services over the entire time horizon (whilst respecting the assignments chosen by previous agents) via its reported ordering. In the case of RSD, given an agent's preference ordering, the same assignment is always optimal no matter the underlying utilities of the agent. That is, it is optimal for the agent to be assigned its most preferred available service, no matter its cardinal preferences. As a result, RSD is trivially truthful in the single-shot matching setting. In contrast, in the offline SBM setting, the optimal assignment of

Algorithm 5: RRSD

Input : $T, N, S, D, \tilde{\succ}$
Output: M

$M = (\emptyset)_{t=1}^{T}$
Sample σ uniformly at random
for $i = 1, \ldots, n$ **do**
  agent $= \sigma(i)$
  for $j = 1, \ldots, s$ **do**
    service $= \tilde{\succ}_{\text{agent}}(j)$
    while *available*$(M, agent, service)$ **do**
      $t = \text{earliest}(M, \text{agent}, \text{service})$
      $M(t, \text{agent}) = \text{service}$
    end
  end
end
return M

services can be different for two different sets of utilities which induce the same preference ordering. Hence, there is no trivial assignment, based on the preference ordering submitted by the agent, which guarantees that agents are truthful.

Instead, given an agent's preference ordering, we attempt to find an assignment which performs reasonably well, regardless of the agent's underlying utilities. RRSD uses a greedy algorithm to compute the assignment given to an agent. As long as this greedy algorithm is a good approximation of the optimal assignment, no matter the agent's underlying utilities, then RRSD will have a bounded incentive ratio. The next theorem formalises this argument.

Theorem 5.6. *The incentive ratio of RRSD is asymptotically bounded below by* $1 - 1/e$.

It is an open question as to whether the provided bound for the incentive ratio of RRSD is tight. More generally, it is unknown whether there exists any algorithm with a better incentive ratio than $1 - 1/e$. One can show that many scheduling problems, such as generic job interval scheduling and (dense) pinwheel scheduling, can be reduced to the optimal manipulation problem each agent faces in RRSD. Whilst it is known that generic job interval scheduling problems are MAXSNP-hard (Chuzhoy et al., 2006), it is still not known whether there exists a scheduling algorithm with approximation ratio better than $1 - 1/e$.

Proof. Without loss of generality, assume that agent k is selected at random in the kth position of the permutation σ sampled by RRSD. Assume, for the moment, that agents 1 to $k - 1$ are not allocated any services. Additionally, suppose that agent k is free to choose its own allocation of services independent of the RRSD algorithm. Under these assumptions, agent k is posed with an offline blocking bandits problem as described in

Basu et al. (2019). The solution proposed by RRSD corresponds to a greedy approach in which the best service available is allocated at each time step. Thus, proving that such a greedy algorithm has an approximation ratio of $1 - \frac{1}{e}$ implies the result in this restricted case. This fact was proven in Basu et al. (2019). We will show that this result holds more generally, regardless of the allocations chosen by RRSD in previous time steps.

Again, assume agent k is free to choose its own allocation, independent of RRSD. That is, agent k is tasked with solving the following integer linear programming problem (ILP):

$$\begin{aligned}
\max_{x_{t,j}} \quad & \sum_{t=1}^{T} \sum_{j=1}^{s} \mu_{k,j} x_{t,j} & \\
\text{s.t.} \quad & x_{t,j} \in \{0,1\} & \forall j \in S \\
& y_{t,j} + x_{t,j} \leq 1 & \forall t \in [T], \forall j \in S \\
& \sum_{j=1}^{s} x_{t,j} = 1 & \forall t \in [T] \\
& \sum_{t \in [D_{k,s}]} x_{t+t_0,j} \leq 1 & \forall t_0 \in T, \forall j \in S
\end{aligned}$$

The variables $x_{t,j}$ indicate whether agent k is assigned service j at time step t. Meanwhile, the constants $y_{t,j}$ indicate whether agent k cannot be assigned service j on time step t due to blocking duration constraints imposed by allocations of service j to agents 1 through $k-1$. The second set of constraints ensure that the assignments chosen by agent k do not breach the blocking duration constraints imposed by pre-existing assignments of services to agents 1 to $k-1$. The third set of constraints ensure that agent k may only be matched to one service at each time step. Lastly, the fourth set of constraints ensure that agent k chooses a sequence of assignments which obeys its own blocking duration constraints.

We will proceed to develop an upper bound on this ILP through a series of relaxations. We will then compare this upper bound to an assignment which is outperformed by RRSD to prove an asymptotic bound on the incentive ratio, as desired.

We derive an upper bound for this ILP through a series of relaxations. First of all, we relax the integer constraints, so that at each time step agent k can assign itself a fractional mixture of services. Additionally, we replace the constants $y_{t,j}$ with variables $z_{t,j}$ constrained to lie in $[0,1]$. The idea in introducing these variables is to remove the blocking constraints imposed by the previous agents and replace them with a constraint that stipulates that the total reduction in the time horizon available for agent k to assign itself each service j must remain the same. That is, agent k is free to fractionally redistribute the blocked parts of the time horizon imposed by the

previous $k-1$ agents. This results in the following linear program (LP):

$$\begin{aligned}
\max_{x_{t,j},\, z_{t,j}} \quad & \sum_{t=1}^{T}\sum_{j=1}^{s} \mu_{k,j} x_{t,j} & \\
\text{s.t.} \quad & x_{t,j} \in [0,1] & \forall j \in S \\
& z_{t,j} + x_{t,j} \leq 1 & \forall t \in [T], \forall j \in S \\
& \sum_{j=1}^{s} x_{t,j} = 1 & \forall t \in [T] \\
& \sum_{t=1}^{T} z_{t,j} = \sum_{t=1}^{T} y_{t,j} & \forall j \in [k-1] \\
& \sum_{t \in [D_{k,s}]} x_{t+t_0,j} \leq 1 & \forall t_0 \in T, \forall j \in S
\end{aligned}$$

It should be immediately obvious that this problem can be reformulated further, and, in fact, the individual fractional assignments per time step can be replaced with fractional assignments of agents to services for the entire time horizon. Similarly, the newly introduced auxiliary variables $z_{t,j}$ can be removed completely. In other words, it is clearly optimal for agent k to spread the blocked parts of the time horizon evenly across all time slots, and then greedily match services to itself in each time step whilst obeying its own blocking duration constraints. Therefore, it only matters how often each service is matched to agent k, as the fractional amount matched for every time step will be the same. This leads us to the following, equivalent, LP reformulation:

$$\begin{aligned}
\max_{a_j} \quad & \sum_{j=1}^{s} a_j \mu_{k,j} & \forall j \in S \\
\text{s.t.} \quad & a_j \in [0, T/D_{k,j}] & \forall j \in S \\
& a_j + \sum_{t=1}^{T} y_{t,j} \leq T & \forall j \in S \\
& \sum_{j=1}^{s} a_j = T &
\end{aligned}$$

Additionally, we let $C_j = \{t \in [T] \,:\, y_{t,j} = 1\}$ be the set of time steps in which agent k cannot (in practice) be matched with service j because of blocking duration constraints imposed by previous agents. Next, we show that this LP can be further formulated as a fractional bounded knapsack problem.

Consider each service j as an item with weight $D_{k,j}$ and value $\mu_{k,j}$. From this perspective, a_j is the (fractional) number of times we pack item j into a knapsack (whose capacity is T). Note that the maximum value a_j can take in the previous LP is determined by the pattern of C_j, and is also capped by $T/D_{k,j}$. Therefore, in our bounded knapsack formulation, we can replace the constraints of a_j to be $a_j \leq T/D_{k,j} - b_j$ where b_j is the number of blocks caused by C_j. Note that in general

$b_j \neq |C_j|$, as it heavily depends on the pattern of the blocks. Since $a_j \geq 0$, we have that $\frac{T}{D_{k,j}} \geq b_j$. It is well known that this fractional bounded knapsack admits the optimal solution $\forall j \in S, a_j^\star = \min\{T/D_{k,j} - b_j, (T - b_j - \sum_{l=1}^{j-1} a_l^\star)^+\}$. Note that the solution $a_j^\star$ implicitly specifies an upper bound for the original ILP.

Now consider the greedy sequence of matches for agent k generated by the RRSD algorithm. Let a_j^g denote the number of times service j is matched to agent k by RRSD. Similarly let A_j denote the set of time slots in which agent k is allocated services 1 to $j-1$. The time slot where the periodic matching of service j to agent k collides with previous matches is denoted by $col_j = \{t \in A_j \cup C_j \,:\, D_{k,j} \mid t\}$. The number of times service j is assigned to agent k is at least $\lceil (T - |col_j|)/D_{k,j} \rceil$. This holds because for service j we can remove the time slots with collisions and perform periodic placement perfectly with the remaining. $T - |col_j|$ time slots. Note that $|col_j| \leq \sum_{l=1}^{j-1} a_l^g + \sum_{t=1}^{T} y_{t,j} - |A_j \cap C_j|$.

We now define for each $j \in S$, $a'_j = T_j/D_{k,j} - b_j$, and $T_j = \left(T - \sum_{l=1}^{j-1} a'_l + |A_j \cap C_j|\right)^+$. We claim that $\sum_{l=1}^{j} a_l^g \geq \sum_{l=1}^{j} a'_l$. In turn, this immediately implies that $\sum_{j=1}^{s} a_j^g \mu_{k,j} \geq \sum_{j=1}^{s} a'_j \mu_{k,j}$. In other words, we will attempt to prove that the solution a'_j specifies a lower bound on the performance of RRSD.

We prove the claim using induction on j. We know that $a_1^g \geq \lceil (T - b_1) D_{1,k} \rceil$, so the base case is satisfied. By the inductive hypothesis, assume that $\sum_{l=1}^{j} a_l^g \geq \sum_{l=1}^{j} a'_l$ for all $j < j'$. We have:

$$
\begin{aligned}
a_{j'}^g &\geq \lceil (T - |col_{j'}|)/D_{k,j'} \rceil \\
&\geq \frac{1}{D_{k,j'}} \left(T - \sum_{l=1}^{j'-1} a_l^g - b_{j'} + |A_{j'} \cap C_{j'}| \right) \\
&= \frac{1}{D_{k,j'}} \left(T - \sum_{l=1}^{j'-1} a'_l - \sum_{l=1}^{j'-1} \left(a_l^g - a'_l\right) - b_{j'} + |A_{j'} \cap C_{j'}| \right) \\
&= a'_{j'} - \frac{1}{D_{k,j'}} \sum_{l=1}^{j'-1} \left(a_l^g - a'_l\right)
\end{aligned}
$$

Thus we have that

$$
\sum_{l=1}^{j'} (a_l^g - a'_l) \geq (1 - 1/D_{k,j'}) \sum_{l=1}^{j'} (a_l^g - a'_l)
$$

which means $\sum_{l=1}^{j'} a_l^g \geq \sum_{l=1}^{j'} a'_l$, and the inductive hypothesis holds. In what follows, we will refer to a' as the lower bound solution. Similarly, we will refer to $a^\star$ as the upper bound solution.

Note that for any j, if $\frac{T}{D_{k,j}} = b_j$ then both the upper bound and lower bound solutions will not contain service j (as $a'_j \leq a_j^\star = 0$). Therefore, without loss of generality, we assume that $\frac{T}{D_{k,j}} > b_j$. We set $D'_{k,j}$ such that $\frac{1}{D'_{k,j}} = \frac{1}{D_{k,j}} - \frac{b_j}{T}$. With induction in j we can

show that $a'_j = \frac{T}{D'_{k,j}} \prod_{l=1}^{j-1}(1 - \frac{1}{D'_{k,l}})$. In addition, we can also show that $a^\star_j \leq \frac{T}{D'_{k,j}} + 1$. The remainder of the proof consists of showing that the allocation a'_j performs asymptotically well compared to the optimal allocation a^* via the closed forms above, which in turn implies the desired asymptotic bound on the incentive ratio of RRSD. The proof of this fact is contained in the proof for the blocking bandits setting provided by Basu et al. (2019), and as a result is omitted. We point the enthusiastic reader to the 'Greedy Lower Bound vs LP Upper Bound' subsection of the proof of Theorem 3.3 in Basu et al. (2019). □

We now provide an upper bound on the distortion achieved by RRSD, which matches our previously established lower bound for randomised policies described in Theorem 5.5.

Theorem 5.7. *The distortion of RRSD is at most* $\mathcal{O}(\sqrt{s})$.

Proof. Our proof proceeds by upper bounding the distortion of RRSD by the distortion of RSD on a new reward profile. The distortion of RRSD is given as

$$\rho = \sup_{\mu, D} \frac{\mathrm{SW}(M^*(\mu, D), \mu)}{\mathbb{E}[\mathrm{SW}(\texttt{RRSD}(\succ, D), \mu)]}.$$

We first upper bound $\mathrm{SW}(M^\star(\mu, D), \mu)$ by the expected social welfare on a new instance with no blocking. For this new instance, there are $\tilde{D}n$ total agents and s services. The $\tilde{D}n$ agents are partitioned into n groups, one for each agent in the original profile. We denote the group of agents corresponding to agent i in the original profile by G_i and the ℓ-th agent in group G_i by i_ℓ. The new reward profile $\tilde{\mu} \in \mathbb{R}^{\tilde{D}n \times s}$ is defined as follows

$$\tilde{\mu}_{i_\ell, j} = \frac{\mu_{i,j}}{D_{ij}} \quad \forall j \, \forall i_\ell \in G_i$$

In the new instance, there is no blocking i.e. all the blocking lengths are one.

Now let $\pi^\star = M^\star(\mu, D)$ be the optimal policy for the original instance μ with blocking. We now construct a new policy for the non-blocking setting with reward matrix $\tilde{\mu}$. The new policy $\tilde{\pi}$ works as follows. At time t, if $\pi^\star$ allocated service j to agent i, then we select one available agent from the group G_i (say i_ℓ) and repeatedly allocate service j to agent i_ℓ for the next $D_{i,j}$ rounds i.e. we set $\tilde{\pi}_{t'}(i_\ell) = j$ for $t' = t, t+1, \ldots, t + D_{i,j} - 1$. Notice that, since there are $\tilde{D}$ agents in group G_i, it is always possible to find such an available service i_ℓ whose allocation hasn't been determined at round t. This is because under the original policy $\pi^\star$, at any time at most $\tilde{D}$ services can be simultaneously blocked as result of being assigned to agent i. Algorithm 6 formally describes how to construct the new policy $\tilde{\pi}$ from the old policy $\pi^\star$.

Let us now compare the social welfare of policy $\pi^\star$ with reward profile μ, and social welfare of policy $\tilde{\pi}$ with reward profile $\tilde{\mu}$. Notice that whenever $\pi^\star$ allocates service j

Algorithm 6: Policy Conversion ($\pi^\star \to \tilde{\pi}$).

Input: T, N, S, D, Policy $\pi^\star$
Output: $\tilde{\pi}$

```
/* Matrix F keeps track of the available agents within each group G_i.
   */
```

$F(i, i_\ell) = 0 \; \forall i_\ell \in G_i \; \forall i \in N$
for $t \in \{1, \ldots, T\}$ **do**
  for $i \in N$ **do**
    if $\pi^\star_t(i) = j$ **then**
      `/* Find an available agent within group G_i */`
      Choose i_ℓ s.t. $F(i, i_\ell) = 0$
      $\tilde{\pi}_{t'}(i_\ell) = j \; \forall t' \in [t, \ldots, t + D_{ij} - 1]$
      $F(i, i_\ell) = 1$
    end
    `/* If any arm becomes available under π*, then we make corresponding agents available */`
    for $j \in S$ **do**
      if $\pi^\star_{t-D_{i,j}+1}(i) = j$ **then**
        `/* i_j ∈ G_i is the corresponding agent with repeated allocations from {t − D_{i,j} + 1, ..., t} under π̃ */`
        $F(i, i_j) = 0$
      end
    end
  end
end
return $\tilde{\pi}$

to agent i, a corresponding agent (say i_ℓ) is assigned service j exactly $D_{i,j}$ times under the new policy $\tilde{\pi}$. As the new rewards are normalized by the blocking lengths, this implies that the total reward gathered by i under $\pi^\star$ is the same as the total reward gathered by all the agents in G_i under the new policy $\tilde{\pi}$. Thus, summing over all the agents, we have $\mathrm{SW}(\pi^\star, \mu) = \mathrm{SW}(\tilde{\pi}, \tilde{\mu})$.

Now observe that, under the new instance $\tilde{\mu}$, there is no blocking, so the optimal allocation rule is obtained by applying a fixed matching (say $\nu^\star$)[3] repeatedly over T rounds. Let $\mathrm{SW}_0(\nu^\star, \tilde{\mu})$ be the one-round social welfare of the matching $\nu^\star$. Then, under the non-blocking reward instance $\tilde{\mu}$, the best possible social welfare is $T \cdot \mathrm{SW}_0(\nu^\star, \tilde{\mu})$. This gives us the following bound on the welfare of the original policy $\pi^\star$ for the blocking instance.

$$\mathrm{SW}(\pi^\star, \mu) \leq \mathrm{SW}(\tilde{\pi}, \tilde{\mu}) \leq T \cdot \mathrm{SW}_0(\nu^\star, \tilde{\mu}) \tag{5.2}$$

[3]Ideally $\nu^\star$ is an assignment from $\tilde{D}n$ agents to s services based on the optimal achievable social welfare and not a one-to-one matching. But we will use the term matching instead of assignment to be consistent with the rest of the paper.

We now prove a lower bound on the expected social welfare of RRSD under the original reward instance μ. RRSD is a randomized policy, but for a given order of agents, the sequence of assignments generated by RRSD becomes a deterministic policy. Any such deterministic policy π can be converted to an equivalent policy (say $g(\pi)$) through Algorithm 6. Moreover, the new policy $g(\pi)$ preserves the social welfare under the new reward instance $\tilde{\mu}$. The expected social welfare of RRSD is given as

$$\begin{aligned}\mathbb{E}[\mathrm{SW}(\texttt{RRSD}, \mu)] &= \mathbb{E}_{\pi\sim\texttt{RRSD}}[\mathrm{SW}(\pi, \mu)] \\ &= \mathbb{E}_{\pi\sim\texttt{RRSD}}[\mathrm{SW}(g(\pi), \tilde{\mu})]\end{aligned}$$

The last line follows from the welfare preservation property of Algorithm 6. Moreover, given a policy π, the new policy $g(\pi)$ actually assigns the same service repeatedly to the same agent. This is because whenever the original RRSD assigns a new service to agent i, Algorithm 6 selects a new agent in G_i, and assigns the new service to the new agent. Now given $g(\pi)$ consider a simpler policy $g'(\pi)$ which only makes the first repeated assignments to any member from G_i for all i. Since this new policy makes fewer assignments than $g(\pi)$ we have the following inequality.

$$\mathbb{E}_{\pi\sim\texttt{RRSD}}[\mathrm{SW}(g(\pi), \tilde{\mu})] \geq \mathbb{E}_{\pi\sim\texttt{RRSD}}[\mathrm{SW}(g'(\pi), \tilde{\mu})]$$

Since the sequence π was generated under RRSD for the original instance μ, an alternative way to generate the sequence $g'(\pi)$ is the following: first randomly choose an order of the set N, and then replace each agent i in this sequence with a randomly chosen agent from G_i. Let us call this new randomised policy GRSD (short for grouped RSD). Then we have,

$$\begin{aligned}\mathbb{E}_{\pi\sim\texttt{RRSD}}[\mathrm{SW}(g'(\pi), \tilde{\mu})] &= T \cdot \mathbb{E}[\mathrm{SW}_0(\texttt{GRSD}, \tilde{\mu})] \\ &= T \cdot \mathbb{E}[\mathrm{SW}_0(\texttt{RSD}, \tilde{\mu})]\end{aligned}$$

The last equality follows from the following two observations. Under GRSD, the probability that an agent from group i shows up at position j equals $1/n$. On the other hand, under RSD, the probability that an agent from group i shows up at position j equals $\tilde{D} \times (1/\tilde{D}n) = 1/n$. Second, conditioned on the event an agent from group i shows up at position j, the expected utility of the agent is the same, as all the agents in group G_i are duplicates of the original agent i and have the same reward profile. Therefore, we have established the following lower bound on the expected welfare of RRSD under the original instance μ.

$$\mathbb{E}[\mathrm{SW}(\texttt{RRSD}, \mu)] \geq T \cdot \mathbb{E}[\mathrm{SW}_0(\texttt{RSD}, \tilde{\mu})] \tag{5.3}$$

We can now bound the distortion of RRSD as follows.

$$\begin{aligned}\rho &= \sup_{\mu} \frac{\mathrm{SW}(\pi^\star, \mu)}{\mathbb{E}_{\pi\sim\texttt{RRSD}}[\mathrm{SW}(\pi, \mu)]} \\ &\leq \sup_{\tilde{\mu}} \frac{T \cdot \mathrm{SW}_0(\nu^\star, \tilde{\mu})}{T \cdot \mathbb{E}_{\nu\sim\texttt{RSD}}[\mathrm{SW}_0(\nu, \tilde{\mu})]} \\ &= \sup_{\tilde{\mu}} \frac{\mathrm{SW}_0(\nu^\star, \tilde{\mu})}{\mathbb{E}_{\nu\sim\texttt{RSD}}[\mathrm{SW}_0(\nu, \tilde{\mu})]}\end{aligned}$$

Where the inequality is due to the lower bound from (5.3) and the upper bound from (5.2). Since the last quantity is just the distortion of RSD in the single shot matching setting, we can apply Lemma 4 from Filos-Ratsikas et al. (2014) and get a bound of $\mathcal{O}(\sqrt{s})$ on the distortion.[4] □

5.2.4 Derandomised RRSD

In this section, we present a deterministic matching policy for the offline SBM setting. More precisely, we present derandomised RRSD (DRRSD), which, as the name suggests, is a derandomised version of RRSD. Instead of sampling a single permutation, like RRSD, DRRSD uses a set of $4n^2 \log(n)$ permutations. In addition, this set is constrained to ensure that the fraction of permutations in which an agent i appears in the jth position is at least $\frac{1}{2n}$. The following lemma stipulates that such a set of permutations always exists.

Lemma 5.8. *There exists a set of $4n^2 \log(n)$ permutations over n agents such that the fraction of times agent i appears at the jth position is at least $\frac{1}{2n}$.*

Proof. The proof is by the probabilistic method. Let us draw P permutations over the n agents uniformly at random. Let X_{ij} be the fraction of times agent i appears at jth position over the P permutations. Then $\mathbb{E}[X_{ij}] = 1/n$. Moreover, from the Chernoff-Hoeffding inequality,

$$P\left(X_{ij} \leq \frac{1}{2n}\right) \leq 2e^{-2P\frac{1}{4n^2}} = 2e^{-\frac{P}{2n^2}}.$$

Moreover, by a union bound over the n agents and n positions we get that

$$P\left(\exists i, j\ X_{ij} \leq \frac{1}{2n}\right) \leq 2e^{-\frac{P}{2n^2}}.$$

Therefore, if $P \geq 4n^2 \log(n)$, the probability of observing a set of permutations such that each $X_{ij} \geq 1/2n$ is positive. This implies that if $P = 4n^2 \log(n)$, we can find a required set of permutations. □

[4] Filos-Ratsikas et al. (2014) actually considered a setting where $n = s$, but their proof naturally generalizes for the setting with $n > s$

DRRSD splits the time horizon into evenly sized blocks. In each block, a different permutation is used. Within each block, agents are assigned to services by the same greedy method used by RRSD, with one caveat. If the blocking duration caused by the assignment of an agent-service pair would overrun into the next block, then this assignment is skipped. This to ensure that all services will be available at the beginning of each block. The pseudocode for DRRSD is presented in Algorithm 7.

Next, we prove that DRRSD incurs a distortion of order $\mathcal{O}(s)$, which matches the lower bound we established for the distortion of deterministic policies in Theorem 5.4.

Theorem 5.9. *There is an admissible deterministic policy with distortion at most $\mathcal{O}(s)$ for any $T \geq O(n^2 \log(n))$*

Proof. We will write i_j to denote agent i's jth favourite service. That is, $i_j = \succ_i(j)$. By Lemma 5.8, agent i gets her jth favourite service (or better) in at least $\frac{P}{2n}$ groups. Within any such group, there are T/P time slots, and agent i is assigned her jth favourite service at least $\left\lfloor T/(PD_{i,i_j}) \right\rfloor$ times. Therefore, the total welfare guaranteed by DRRSD is at least

$$\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=1}^{S} \mu_{i,i_j} \frac{P}{2n} \left\lfloor \frac{T}{PD_{i,i_j}} \right\rfloor \\
\geq &\sum_{i=1}^{n}\sum_{j=1}^{s} \mu_{i,i_j} \frac{P}{4n} \frac{T}{PD_{i,i_j}} \\
= &\frac{T}{4n} \sum_{i=1}^{n}\sum_{j=1}^{s} \frac{\mu_{i,i_j}}{D_{i,i_j}}
\end{aligned}$$

On the other hand, consider a matching algorithm that assigns service i_j to agent i exactly $A_{i,j}$ times. Whenever item j is matched to agent i, it is blocked for D_{i,i_j} rounds. This implies that $A_{i,j} \leq T/D_{i,i_j}$. Therefore, the maximum welfare achievable by such a matching algorithm is at most

$$\sum_{i=1}^{n}\sum_{j=1}^{s} \mu_{i,j} A_{i,j} \leq \sum_{i=1}^{n}\sum_{j=1}^{s} \mu_{i,j} \frac{T}{D_{i,i_j}}$$

This establishes that the distortion of DRRSD is at most $4n = \mathcal{O}(s)$. □

5.3 SBM with Bandit Feedback

Note that, in order for the guarantees above to hold in practice, we must assume that agents are fully aware of their ordinal preferences before matching begins. However, in many real-world scenarios, agents may be initially unaware of their preferences and

Algorithm 7: DRRSD (Derandomized RRSD)

Input: T, N, D, S, $\succ$, and a set of $P = 4n^2 \log(n)$ permutations $\{\sigma_1, \ldots, \sigma_P\}$

$M = (m_t)_{t=1}^T = (\emptyset)_{t=1}^T$
for $p = 1, \ldots, P$ **do**
  $\sigma = \sigma_p$
  for $i = 1, \ldots, n$ **do**
    // Select agent
    $\text{ag} = \sigma(i)$
    $\text{start} = (p-1)T/P$
    $\text{end} = pT/P$
    for $j = 1, \ldots, s$ **do**
      // Select service
      $\text{ser} = \tilde{\succ}_{\text{ag}}(j)$
      while *available*$(M, ag, ser, start, end)$ **do**
        $t = \text{earliest}(M, \text{ag}, \text{ser}, \text{start}, \text{end})$
        if *overrun*(ag, ser, t, end) **then**
          break
        end
        $M(t, \text{ag}) = \text{ser}$
      end
    end
  end
end
return M

learn them over time by matching with services. In addition, the reward an agent receives for being matched with a service may be inherently stochastic, depending on unobservable aspects of the underlying environment. With these concerns in mind, we present a new sequential blocked matching setting, which we call the online SBM setting with bandit feedback, or online SBM for short. Note that our use of the term "bandit" refers to the feedback received by agents, rather than the feedback received by the central planner.

In the online SBM setting, matching occurs time step by time step. At the beginning of each time step, every agent must submit a report, $\tilde{\succ}_i^t$, to the planner. The planner is then tasked with returning a matching of agents to services which obeys the blocking constraints imposed by the matchings from previous time steps. At the end of each time step, agent i receives a reward, $r_{i,t} \in [0, 1]$, sampled from a distribution with mean $\mu_{i,j}$, where j is the service agent i was assigned in the matching returned by the planner. In other words, each agent receives bandit feedback according to the service they were assigned. In contrast, the preference reports of each agent are fully revealed to the central planner at the start of each time step. Additionally, we assume that each agent maintains an internal estimation, $\succ_i^t$, of its own preference ordering at every time step, based on the rewards received thus far.

We use $H_t^{\succ} = (\succ^1, \ldots, \succ^t)$ to denote the report history up to time step t. Furthermore, we use $H_t^m = (m_1, \ldots, m_t)$ to describe the matching history at the end of time step t. We say that a matching history is feasible if its matchings form a feasible matching sequence. Similarly, we use $H_t^r = (r_1, \ldots, r_t)$ to denote the reward history. That, is the tuple of reward vectors observed by the agents at every time step. An (online) matching policy $\pi = (\pi_1, \ldots, \pi_T)$ is a tuple of functions $\pi_t(m|\tilde{H}_t^{\succ}, H_t^m, D)$ which assigns a probability of returning the matching m given a report history $H_t^{\succ}$, a feasible matching history H_t^m and a blocking duration matrix D. Similar to the offline setting, we say that a matching policy is admissible if it always returns a feasible matching sequence.

Likewise, an (online) report policy for agent i, $\tilde{\psi}_i = (\tilde{\psi}_1, \ldots \tilde{\psi}_t)$, is a tuple of functions $\tilde{\psi}_t(\succ_i^t | H_t^r, H_t^m, D)$ which assign a probability to agent i reporting $\succ_i^t$ at time step t given a reward history H_t^r, a matching history H_t^m, and blocking duration matrix D. We denote by $\tilde{\psi} = (\tilde{\psi}_1, \ldots, \ldots \tilde{\psi}_n)$ the tuple of report policies used by the agents. As before we use the notation $\tilde{\psi}_{-i}$ to denote the report policies of all agents bar agent i and use ψ to denote the tuple of report policies where each agent reports its internal estimation $\succ_i^t$ at every time step. We say that an agent is truthful if it employs the report policy ψ_i.

The goal of each agent is to employ a report policy that maximises the sum of their own rewards across the time horizon. In contrast, goal of the planner is to employ a matching policy which maximises the sum of rewards across all agents and across all time steps. Note that these goals are simply the stochastic analogs of maximising welfare and social welfare respectively.

As we saw in Chapter 2, classical bandits problems typically adopt policy benchmarks which correspond to repeatedly taking the same action in hindsight. In the case of SBM, such policies correspond to repeatedly selecting the same matching in as many time steps as possible. Such a benchmark policy may have very poor performance relative to the optimal matching sequence in expectation, and as such, classical notions of regret are unsuitable performance measures in the online SBM setting. To resolve this issue, we propose the following regret definition:

Definition 5.10. The dynamic α-regret of a policy π is:

$$R_\pi^\alpha(D, \mu, T) = \alpha \mathrm{SW}(M^*, \mu) - \mathbb{E}_{\psi,\pi}\left[\sum_{i=1}^{n}\sum_{t=1}^{T} r_{i,t}\right]$$

In other words, we compare the expected performance of a matching policy against a dynamic oracle which returns an $1/\alpha$-optimal solution to the corresponding offline SBM problem, under the assumption that agents truthfully report their internal estimation of their preferences at each time step. Recall that, in the offline SBM setting,

the distortion incurred by any policy is at least $\Omega(s)$. As a result, we cannot expect to construct algorithms with vanishing $1/\alpha$-regret for $\alpha < \sqrt{s}$. In addition, one would not expect any matching policy to have low dynamic regret if the internal estimations computed by each agent are inaccurate. For example, if any agent's internal estimator consists of returning a random preference ordering, then we cannot hope to learn about said agent's preferences. As a result, we need to make reasonable assumptions regarding the internal estimator of each agent.

Similar to distortion for the offline SBM setting, dynamic α-regret is only a meaningful performance measure for policies which motivate agents to adopt truthful reporting policies. Inspired by the concept of incentive ratio for the offline SBM setting, we define a new notion of regret which, given a matching policy π captures the expected gain in cumulative reward an agent can achieve by misreporting.

Definition 5.11. For a given matching policy π, we define agent i's α–IC regret (or α-incentive compatible regret) as follows:

$$\mathrm{I}_{\pi}^{\alpha}(D, \mu, T) = \\ \alpha \max_{\tilde{\psi}} \mathbb{E}_{(\psi_{-i}, \tilde{\psi}_i), \pi} \left[\sum_{t=1}^{T} r_{i,t} \right] - \mathbb{E}_{\psi, \pi} \left[\sum_{t=1}^{T} r_{i,t} \right]$$

Note that for some matching policies, computing the optimal reporting policy may be computational intractable. If agents have vanishing α-IC regret for a such a matching policy, then truthtelling forms a good approximation of each agent's optimal reporting policy. If this approximation is better than what can be computed by the agent, then we can expect each agent to adopt their truthful reporting policy. Thus, we seek matching policies with good guarantees with respect to both dynamic α-regret and α-IC regret.

5.4 Algorithms for Online SBM

Next, we present a matching policy which achieves meaningful guarantees with respect to both dynamic α-regret and α-IC regret. More precisely, we present the bandit repeated random serial dictatorship (`BRRSD`) algorithm. Before we describe `BRRSD` formally, we first state our assumptions regarding the internal estimator used by each agent.

Let $\hat{\mu}_{i,j}$ denote the empirical mean of the reward samples agent i receives from being assigned service j. We say that an agent i is mean-based if service a is preferred to service b in $\succ_i^t$ if and only if $\hat{\mu}_{i,a} \geq \hat{\mu}_{i,b}$. That is, a mean-based agent prefers services with higher empirical mean reward. From here on, we assume that all agents are mean-based.

Additionally, we use $\Delta_{\min}$ to denote the smallest gap in mean rewards between two services for the same agent. That is, $\Delta_{\min} = \min_{i,a \neq b} |\mu_{i,a} - \mu_{i,b}|$. Note that $\Delta_{\min}$ is analogous to common complexity measures used in bandit exploration problems. Intuitively, if the mean rewards received from being assigned two services are similar, it will take more samples for a mean-based agent to decide which service they prefer.

We are now ready to describe BRRSD. BRRSD is split into two phases. In the first phase, BRRSD assigns each agent each service exactly $\lceil 2\log(2Tsn)/\Delta_{\min}^2 \rceil$ times. BRRSD performs these assignments in a greedy manner. At each time step, BRRSD iterates through the agent-service pairs that still need to be assigned in an arbitrary order. If an agent-service pair does not violate blocking constraints, then it is added to the current matching. Once this iteration is completed, or all agents have been assigned services, the matching is returned and BRRSD moves onto the next time step. Once all required assignments have been completed, BRRSD waits until all services are available, matching no agents to services in the meantime. Note that this takes a maximum of $\tilde{D}$ rounds. Then, BRRSD begins its second phase. At the beginning of the next time step, BRRSD observes the report profile $\succ_i^t$ and selects matchings according to RRSD using this report profile for the remainder of the time horizon. BRRSD is described formally in Algorithm 8.

BRRSD falls in the class of explore-then-commit (ETC) algorithms common in the bandit literature (Lattimore and Szepesvári, 2020). The first phase of BRRSD serves as an exploration phase in which agents learn their preference ordering. Meanwhile, the second phase of BRRSD serves as an exploitation phase in which agents have the opportunity to disclose their accumulated knowledge to the planner in the form of ordinal preferences. Observe that this decoupling of exploration and exploitation avoids complicated incentive issues that may arise for sequential algorithms, which make no such clear separation.

The exploration phase of BRRSD is simple relative to typical approaches in the bandit exploration literature. One may hope to apply a more complicated scheme for exploration, however, approaches with better performance guarantees typically depend directly on the reward samples observed, which the planner does not have access to. The next theorem describes the guarantees of BRRSD in terms of α-dynamic regret and α-IC regret.

Theorem 5.12. *Under the assumption that agents are mean-based, the following is true for all μ and D:*
(i) The dynamic $(1/\sqrt{s})$-regret of BRRSD is $O\left(\tilde{D}\sqrt{s}\log\left(Tsn\right)/\Delta_{min}^2\right)$.
(ii) The $(1-1/e)$-IC regret for all agents under BRRSD is $O\left(\tilde{D}s\log\left(Tsn\right)/\Delta_{min}^2\right)$.
(iii) The greedy algorithm used by BRRSD in the exploration phase uses at most twice as many time steps as the shortest feasible matching sequence which completes the required assignments.

Proof. Claim (i) can be proved as follows. By the end of the exploration phase, we know that each agent has received a reward from being assigned each service at least $\lceil 2\log(2Tsn)/\Delta_{\min}^2 \rceil$ times. In addition, note that this exploration phase takes at most $\tilde{D}s\lceil 2\log(2Tsn)/\Delta_{\min}^2 \rceil + \tilde{D}$ rounds. By the Chernoff-Hoeffding inequality, we have that for all agents i and services j:

$$P\left(|\mu_{i,j} - \hat{\mu}_{i,j}| \geq \frac{\Delta_{\min}}{2}\right) \leq \frac{1}{Tsn}.$$

Thus, by the union bound and the assumption that all agents are mean-based, with probability $1 - 1/T$, the internal estimation of every agent will be correct. From now, unless explicitly stated, we will assume that all agents have learned the correct preference ordering by the end of the exploration phase.

For the sake of simplicity, let T_1 denote the number of rounds for which the exploration phase runs, and let T_2 denote the number of rounds for which the exploitation phase runs. Similarly let OPT_1 denote the social welfare of the optimal matching sequence of length T_1, and OPT_2 denote the social welfare of the optimal matching sequence of length T_2. Furthermore, let $\text{SW}_1(\texttt{BRRSD})$ denote the social welfare generated by `BRRSD` in the exploration phase, and $\text{SW}_2(\texttt{BRRSD})$ denote the social welfare generated by `BRSSD` in the exploitation phase.

As each reward is bounded between $[0, 1]$, and the exploration phase proceeds for at most $\tilde{D}s\lceil 2\log(2Tsn)/\Delta_{\min}^2 \rceil + \tilde{D}$ time steps, it is easy to show that

$$\frac{1}{\sqrt{s}}\mathbb{E}\left[\text{OPT}_1\right] - \mathbb{E}\left[\text{SW}_1(\texttt{BRRSD})\right] \leq \frac{1}{\sqrt{s}}\left(\tilde{D}s\left\lceil 2\log(2Tsn)/\Delta_{\min}^2 \right\rceil + \tilde{D}\right).$$

In addition, by Theorem 5.7, and the assumption that agents are mean-based, we have the following lower bound:

$$\frac{1}{\sqrt{s}}\mathbb{E}\left[\text{OPT}_2\right] \leq \mathbb{E}\left[\text{SW}_2(\texttt{BRRSD})\right].$$

Let OPT denote the social welfare of the optimal matching sequence of length T. Combining the bounds above and noting that $\text{OPT} \leq \mathbb{E}[\text{OPT}_1] + \mathbb{E}[\text{OPT}_2]$ we have:

$$\frac{1}{\sqrt{s}}\mathbb{E}[\text{OPT}] - \mathbb{E}[\text{SW}(\texttt{BRRSD})] \leq \frac{1}{\sqrt{s}}\left(\tilde{D}s\left\lceil 2\log(2Tsn)/\Delta_{\min}^2 \right\rceil + \tilde{D}\right)$$

For the case when at least one agent does not learn the correct preference ordering by the end of the exploration phase, the $(\frac{1}{\sqrt{s}})$-dynamic regret of `BRRSD` is bounded above

by $\frac{1}{\sqrt{s}}nT$. Combining both cases together, we see that the dynamic $(\frac{1}{\sqrt{s}})$-regret of BRRSD is bounded above by $\frac{1}{\sqrt{s}}\left(\tilde{D}s\left\lceil 2\log(2Tsn)/\Delta_{\min}^2\right\rceil + \tilde{D}\right) + \frac{1}{\sqrt{s}}n$, implying the desired regret bound.

To prove claim (ii), note that the an agent can only affect its assignment of services in the exploitation phase, in which BRRSD deploys the RRSD algorithm. Thus, following a similar argument as above, replacing the use of Theorem 5.7 with Theorem 5.6, we find that the $(1-1/e)$-IC regret of BRRSD is bounded above by $(1-1/e)\left(\tilde{D}s\left\lceil 2\log(2Tsn)/\Delta_{\min}^2\right\rceil + \tilde{D}\right) + (1-1/e)n$.

Finally, to prove claim (iii), we consider the open shop scheduling problem. An instance of the open shop problem consists of a set of N machines and S jobs. Associated with each job j is a set of n independent tasks $j_1, \ldots, j_n$. The task j for job i must be processed on machine i for an uninterrupted $D_{i,j}$ time units. A schedule assigns every task j_i to a time interval $D_{i,j}$ so that no job is simultaneously processed on two different machines, and so that no machine simultaneously processes two different jobs. The makespan C_{max} of a schedule is the longest job completion time. The optimal makespan is denoted by $C^{\star}_{max}$.

It is easy to show that the exploration phase of BRRSD reduces to an open shop scheduling problem in which there is a job for each agent i, and a task for each assignment of service j to agent i. Similarly, observe that the assignment procedure used by BRRSD is simply an implementation of the greedy algorithm for open shop scheduling as described by Woeginger (2018). The claim follows from that fact that the greedy algorithm is a 2-approximation for open shop scheduling (see Woeginger (2018)). □

5.5 Conclusion

In this chapter, we introduced the sequential blocked matching problem. We first considered an offline version of SBM, in which agents are fully aware of their preferences in advance. We saw, via a simple example, that trivial algorithms, such as repeatedly applying RSD on every time step, are not truthful in the SBM setting. As a result, we proposed RRSD, an extension of the RSD, which is asymptotically optimal in terms of distortion. In addition, we showed that RRSD has bounded incentive ratio, and thus encourages agents to report truthfully. Moreover, we showed that RRSD could be derandomised to devise an algorithm that matches the distortion lower bound for deterministic policies. Then, we considered a bandit version of SBM, in which agents must learn their preferences over time by being assigned services. We developed BRRSD for this setting, and showed that BRRSD has sublinear dynamic α-regret for $\alpha = \sqrt{s}$, and ensures that each agent has sublinear α-IC regret for $\alpha = 1 - 1/e$.

Note that RRSD relied conceptually upon RSD. More specifically, RRSD is a serial dictatorship algorithm in which each agent chooses their entire assignment of services over the entire time horizon all at once. It is natural to ask whether other popular algorithms, such as the probabilistic serial mechanism can be generalised to the SBM setting in a similar manner. Extending single shot matching algorithms, such as the PS mechanism, to the SBM setting therefore presents an interesting direction for future work. We also only considered agents with cardinal preferences that report ordinally. Recall that, in the online SBM setting, agents receive cardinal feedback upon being assigned a service. However, as agents report ordinally, the central planner cannot access this information, precluding the use of UCB-style algorithms. Put differently, if agents could report cardinally, then a wider range of bandit algorithms would be available to the decision maker. In terms of performance metrics, only social welfare was considered in this chapter. In real world settings, the central planner may want to ensure that each agent is treated fairly. In this case, alternate solution concepts such as Nash social welfare and CEEI are more suitable, as discussed in Chapter 2. Designing algorithms for SBM with such fairness criteria in mind presents a significant technical challenge.

Algorithm 8: `BRRSD` (Bandit `RRSD`)

Input: T, N, D, S, $\Delta_{\min}$

$M = (m_t)_{t=1}^{T} = (\emptyset)_{t=1}^{T}$
repeats $= \lceil 2\log(2Tsn)/\Delta_{\min}^2 \rceil$
`// Build a list of exploration assignments`
jobList $=$ buildJobList$(n, s,$ repeats$)$
explore $=$ true
waiting $=$ false
count $= 0$
for $t = 1, \dots T$ **do**
  `// Exploration phase`
  if *explore* **then**
    `// Greedily add remaining agent-service pairs to current matching`
    for (i, j) *in jobList* **do**
      if *available(M, i, j, t)* **then**
        $M(t, i) = j$
        jobList.remove(i, j)
      end
    end
    `// Start waiting phase`
    if *jobList.isEmpty* **then**
      explore $=$ false
      waiting $=$ true
    end
  end
  `// Wait for all services to become available`
  if *waiting* **then**
    if *count* $< \tilde{D}$ **then**
      count++
    end
    `// Start exploitation phase`
    else
      $\tilde{\succ} = \tilde{\succ}^t$
      Sample σ
      waiting $=$ false
    end
  end
  `// Exploitation phase`
  else
    for $i = 1, \dots n$ **do**
      ag $= \sigma(i)$
      for $j = 1, \dots, s$ **do**
        ser $= \tilde{\succ}_{\text{ag}}(j)$
        if *available*(M, ag, ser, t) **then**
          $M(t,$ ag$) =$ ser
        end
      end
    end
  end
end
return M

Chapter 6

Conclusions

In real world settings, actions typically have a wide range of consequences that must be carefully considered by the decision maker. For example, once again consider the insurance problem that we have revisited many times throughout this thesis. The decisions made by the insurer effect not only their own financial stability, but the financial stability of their customers. If the insurer chooses to provide expensive insurance quotes in a bid to increase their own profits, customers may become dissatisfied and encouraged to take mitigating actions. For example, a larger proportion of the customer base may turn to fraud with the intention of lowering their insurance premiums. Put differently, narrow-minded decision makers, who focus solely on their own goals and incentives, may fall prey to the second-hand consequences of their actions.

In the case of the insurance problem, both the insurer and their customer base are engaged in a strategic interaction. More specifically, the incentives of the insurer and the customer base are misaligned. As a result, the insurer must carefully consider the strategic recourse available to their customers before committing before to a decision. The framework we propose in Chapter 3 models this phenomena explicitly, by adding a new set of labels which characterise a data provider's preferences, building upon Stackelberg prediction game framework (Brückner and Scheffer, 2011). Our analysis was focused on linear regression contexts in which all involved parties adopt a square loss function. Within this setting, we saw that Stackelberg empirical risk minimisation (STERM), a natural analog of empirical risk minimisation, satisfied a number of desirable properties. In particular, we devised a polynomial time algorithm (`SDP-BISECT`) for STERM. Additionally, we saw that hypothesis class induced by the strategic behaviour of agents had bounded Rademacher complexity, implying that STERM generalises well.

Second hand consequences need not only occur due to a misalignment of incentives. For example, consider sequential resource deployment problems. In many cases, the

value and availability of a given resource may vary through time. For example, a resource may correspond to some commodity, such as gold, whose supply fluctuates. Moreover, the availability or value of a given resource may be a consequence of when it was last deployed. For instance, consider the disaster response problem in which a decision maker is tasked with assigning emergency vehicles and personnel over time. Once deployed, an emergency vehicle becomes unavailable for redeployment in the near future. In other words, the availability and value of a given resource can explicitly depend on the actions of the decision maker.

The adversarial blocking bandits model, which we introduce in Chapter 4, aims to address such settings. In particular, the adversarial blocking bandits model incorporates both blocking and nonstationary reward sequences. We devised two algorithms, `RGA` and `META-RGA`, which achieve finite-time regret guarantees with respect to a greedy oracle that provably approximates the best the policy in hindsight.

Note that, in Problem Domain 1, the decision maker must account for the strategic behaviour of data providing agents. On the other hand, in Problem Domain 2 the decision maker must account for the blocking of resources. In Problem Domain 3, the decision maker (or central planner) must handle both issues simultaneously. In particular, we saw that the introduction of blocking causes standard truthful algorithms for one-sided matching, such as RSD, to fail in the repeated setting. To rectify this issue, we proposed the `RRSD` algorithm, which is optimal in terms of distortion and has bounded incentive ratio. In addition, we showed how `RRSD` can be naturally extended to a bandit setting in which agents must learn their preferences over time.

To conclude this thesis, we return to the original research requirements outlined in Chapter 1. We treat each problem domain in turn, discussing how each research requirement has been addressed, whilst also taking time to discuss potential directions for future work.

6.1 Linear Regression with Strategic Agents

Recall the framework proposed in Chapter 3 for addressing Problem Domain 1. Each input example is accompanied by two labels. The first label, $y \in \mathbb{R}$, denotes the labelling preferred by the learner, whilst the second label, z, denotes the labelling preferred by the corresponding agent. By varying z, the learner can consider a wide range of agents with varying incentives. Secondly, by varying the cost parameter γ, the learner can consider agents with varying capacities for manipulation. In other words, the framework we investigate is flexible with respect to agent incentives, and addresses Requirement 1a.

In the context of square losses and square costs functions, we proposed the `SDP-BISECT` algorithm, which finds global solutions to the corresponding STERM problem in polynomial time (see Theorem 3.6), significantly improving upon the nonconvex problem reformulation proposed by Brückner and Scheffer (2011) for generic SPGs. That is, we provide a relatively complete answer to Requirement 1b in the case of square losses. One downside of `SDP-BISECT` is that it involves a sequence of SDPs. Although, our empirical runtime experiments shows that the number of SDPs solved by `SDP-BISECT` remains essentially constant as the size of the training dataset grows. Extending directly from our analysis, Wang et al. (2021) improve upon `SDP-BISECT`, and show that only a single second-order cone program needs to be solved in the case of square losses and square costs. From the statistical perspective, Corollary 3.8 illustrates that the Rademacher complexity of the hypothesis class induced by the strategic actions of each agent is bounded in terms of γ. The lower γ, the lower the Rademacher complexity of the induced hypothesis class. As a result, the generalisation error of STERM is lower when agents have a greater capacity for manipulation. In this sense, Corollary 3.8 provides an answer to Requirement 1c in the case of square losses and square costs.

Unfortunately, our analysis focuses on agents that employ a square loss and a square cost function. One can easily think of problem settings in which alternate cost functions may be more suitable. For example, when agents are only permitted to make sparse modifications, a cost function based on the L_1-norm may be more suitable than a square one. As we saw at the end of Chapter 3, solving the corresponding SPG for this problem is equivalent to optimising over a hypothesis class of nonconvex piecewise linear functions. Therefore, we conjecture that finding global solutions to this version of STERM may be an intractable problem. There are also many real world settings where other loss functions are better models of agent behaviour than the square loss. For example, consider any setting in which output labels correspond to a test grading. Of course, each agent would like to achieve a perfect score, but often will be satisfied with a passing grade. In this case, the ϵ-insensitive loss is a far more reasonable model of each agent's incentives than the square loss. Summarising, we conclude that Requirements 1a through 1c have been fully addressed in the context of square losses and square costs, but open questions remain regarding other losses and cost functions.

Additionally, observe that Stackelberg prediction games have a sole focus on minimising prediction error, and pay no mind to other properties that are typically of interest in multiagent systems. For example, the learner may wish to select a hypothesis which is incentive compatible, and encourages agents to submit data without modification. Alternatively, the learner may wish to ensure that the selected hypothesis satisfies some notion of fairness, such as statistical parity or bounded group loss (Agarwal et al., 2019). One advantage of the Stackelberg prediction game

framework is that these properties can be easily integrated into STERM via the introduction of additional constraints. Solving constrained STERM problems, where constraints map to certain desirable properties, such as fairness and incentive compatibility, presents a natural direction for future work.

6.2 Sequential Deployment of Reusable Resources

To address Problem Domain 2, we proposed the adversarial blocking bandits setting. Unlike previous bandit settings, adversarial blocking bandits model nonstationary rewards and the blocking of resources simultaneously. In particular, we propose the `RGA` algorithm for his setting, which achieves sublinear regret (see Theorem 4.3) with respect to `Greedy-BAA`, a greedy oracle algorithm which provably approximates the optimal arm pulling policy (see Theorem 4.2). In addition, we provide a number of lower bounds on the α-regret of any policy, covering a wide variety of assumptions on the path variation of rewards and maximal blocking duration (see Theorems 4.5 and 4.6). In particular, when the maximal blocking duration is constant, we found that a matching lower bound for `RGA` exists. As a result, we provide a relatively complete characterisation of finite-time regret guarantees in the adversarial blocking bandits setting, addressing Requirement 2c.

In addition, we showed that `RGA` achieves sublinear α-regret under a wide range of variation budget constraints which are popular in the literature, including the path variation budget (Besbes et al., 2014), maximum variation budget, and number of changes budget (Auer et al., 2019) (see Corollaries 4.9 and 4.10). Hence, `RGA` can accommodate a wide range of realistic reward sequences, addressing Requirement 2a. For example, consider a disaster response setting, where the relevance of emergency personnel may change rapidly depending on the situation at hand. In this case, one could employ `RGA` with a number of changes budget constraint on reward sequences. Unfortunately, `RGA`, assumes that the decision maker is fully aware of the total variation budget of reward sequences in advance. To rectify this issue, we proposed `META-RGA`, a meta-bandit algorithm which requires no advance information regarding the total variation of rewards.

Recall that the adversarial blocking bandits setting allows the blocking duration associated with each arm to change arbitrarily. As a result, adversarial blocking bandits provide a more general model for resource unavailability compared to other blocking bandit models, such as stochastic blocking bandits (Basu et al., 2019), which typically assume the blocking duration associated with each arm is fixed. In this sense, the adversarial blocking bandit setting addresses Requirement 2a. However, one may argue that the adversarial blocking bandits setting is too pessimistic in this regard. In many settings, the blocking duration associated with each arm varies in a

nice manner, rather than changing erratically every time step. For example, consider an expert crowdsourcing problem where a job requester is tasked with assigning projects to workers. Most projects will take each worker roughly the same amount of time to complete, with there being there being the occasional project which requires a long-term commitment. In this example, the variation of blocking durations is bounded. Just as we do for rewards, we could restrict ourselves to problem instances where blocking durations obey a path variation budget constraint, in the hope of achieving better performance guarantees. In other words, the investigation and development of blocking bandit settings, in which both rewards and blocking durations must obey a variation budget, emerges as a potential direction for future work.

Moreover, note that the reward sequences considered in the adversarial blocking bandits problem are nonstochastic in nature. In many real world settings, decision makers receive noisy feedback regarding the efficacy of a given action. As a result, it is natural to consider blocking bandit problems in which rewards are both nonstationary and stochastic. As discussed at the end of Chapter 4, we believe that a combination of existing approaches (Basu et al., 2019; Garivier and Moulines, 2011) may be sufficient to address settings in which the blocking durations associated with each arm are fixed, and the rewards are nonstationary and stochastic, though we leave this as a direction for future work.

6.3 Repeated Matching of Reusable Resources

In Chapter 5, we proposed the sequential blocked matching problem, in which a central planner is tasked with constructing matchings repeatedly whilst adhering to blocking constraints. We saw that traditional one-sided matching algorithms, such as RSD, are no longer truthful in this setting, as agents may exploit the blocking of services to obtain a better allocation in the long-term. In other words, agents can benefit from being non-myopic, and sacrifice utility in the short term to arrange a pattern of blocking which is beneficial. To rectify this problem, we proposed repeated RSD (`RRSD`), a natural extension of RSD, which makes each agent to choose a greedy allocation of services over the entire time horizon in a random order. Whilst not truthful, `RRSD` does have bounded incentive ratio (see Theorem 5.6). That is, an agent can only improve their allocation through misreporting by a fixed multiplicative factor, no matter their preferences. In this sense, `RRSD` is approximately truthful, providing a partial resolution to Requirement 3b.

Additionally, we investigated the distortion incurred by any randomised or deterministic policy in the sequential blocked matching setting. In particular, we showed that any randomised (deterministic) policy must incur $\Omega(\sqrt{s})$ ($\Omega(s)$)

distortion (see Theorems 5.4 and 5.5). Following this we showed that RRSD matches the established lower bound on distortion for randomised policies (see Theorem 5.7), and that RRSD can be derandomised to produce an algorithm which is optimal amongst deterministic policies (see Theorem 5.9). In this sense, we completely address Requirement 3a in the case of cardinal preferences and ordinal reporting.

Lastly, we investigated a bandit version of the sequential blocked matching setting, where agents must learn their preferences over time. For this setting, we proposed a bandit version of RRSD (BRRSD) based on the ETC paradigm for multi-armed bandits. To evaluate the efficiency and truthfulness properties of BRRSD, we introduced two notions of regret. In particular, we showed that BRRSD incurs sublinear dynamic α-regret, implying that the reward accumulated by BRRSD approaches that accumulated by RRSD as the time horizon lengthens. Additionally, we showed that each agent achieves sublinear $(1 - 1/e)$-IC regret under BRRSD when they report truthfully (see Theorem 5.12). This implies that truthful reporting is a $(1 - 1/e)$-approximation of the best misreporting policy in the asymptotic sense. In other words, BRRSD allows each agent to learn their preferences, whilst also being approximately optimal and truthful, satisfying Requirement 3c.

It is an open question as to whether there exists a tractable matching policy which is optimal in terms of distortion and has better incentive ratio than RRSD . Establishing tight upper bounds on the best achievable incentive ratio presents a significant challenge, and would imply the existence of similar upper bounds for the generic job scheduling problem, as discussed in Chapter 5. Likewise, it is unclear whether there exist algorithms for the online sequential blocked matching setting which achieve better regret guarantees than BRRSD. Both of these open questions present immediate directions for future work.

Observe that our analysis is limited to settings in which agents have cardinal preferences and report information ordinally. Investigating agents with different preference and reporting structures is an obvious avenue for future work. When agents report cardinally, we conjecture that better algorithms, with stronger regret guarantees than BRRSD, exist. For instance, with access to cardinal information, the central planner can construct confidence bounds on the behalf of each agent and employ UCB-style algorithms. Moreover, our work only considers social welfare, and neglects other meaningful performance benchmarks that better trade-off fairness and efficiency, such as Nash social welfare and CEEI.

Whether investigating new performance metrics, or different reporting and preference structures, a natural approach to designing algorithms for repeated matching is to adapt existing algorithms from the corresponding single shot setting. The RSD algorithm had a fairly natural translation to the repeated setting. Put simply, instead of applying a serial dictatorship over a single time step, one applies a serial

dictatorship over the entire time horizon. Other algorithms for one-sided matching, such as the PS mechanism, may not have such intuitive extensions. As a result, developing repeated matching algorithms for different versions of the sequential blocked matching problem presents a significant technical challenge.

Appendix A

Additional Results (Chapter 3)

This appendix describes additional material related to Chapter 3. In particular, we describe the dual of the semidefinite program that is solved in each iteration of the `SDP-BISECT` algorithm. After this, in Section A.2, we provide an empirical evaluation of `SDP-BISECT` on the red wine dataset. Finally we compare the runtime of `SDP-BISECT` against an interior point method, which solves the nonconvex problem relaxation for Stackelberg prediction games proposed by Brückner and Scheffer (2011).

A.1 The Dual Problem

In this section, we describe the dual of the SDP that we solve at each iteration of `SDP-BISECT`. This dual can be used to obtain a linear predictor at every iteration. First, recall the SDP which is solved at each time step of `SDP-BISECT`:

$$\max_{\tau,\lambda} \quad \tau \quad \text{s.t.} \quad \begin{bmatrix} A + \lambda B & \mathbf{a} + \lambda \mathbf{b} \\ \mathbf{a}^T + \lambda \mathbf{b}^T & c - \tau \end{bmatrix} \succeq 0 \tag{A.1}$$

We can rewrite this SDP as follows:

$$\max_{\tau,\lambda} \quad \begin{bmatrix} 1 \\ 0 \end{bmatrix}^T \begin{bmatrix} \tau \\ \lambda \end{bmatrix} \quad \text{s.t.} \quad \lambda \begin{bmatrix} -B & -\mathbf{b} \\ -\mathbf{b}^T & 0 \end{bmatrix} + \tau \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \preceq \begin{bmatrix} A & \mathbf{a} \\ \mathbf{a}^T & c \end{bmatrix}$$

Taking the dual yields the following SDP:

$$\min_{W \in \mathbb{S}^{n+2}} \quad \begin{bmatrix} A & \mathbf{a} \\ \mathbf{a}^T & c \end{bmatrix} \cdot W \quad \text{s.t.} \quad \begin{bmatrix} B & \mathbf{b} \\ \mathbf{b}^T & 0 \end{bmatrix} \cdot W = 0, \quad \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \cdot W = 1, \quad W \succeq 0 \tag{A.2}$$

Taking an appropriate rank-1 decomposition of the optimal solution to Problem (A.2) yields a vector containing a linear predictor and its squared Euclidean norm. As a direct consequence of the results in Section 3.5, this linear predictor is optimal for the Dinkelbach program associated with the primal SDP (A.1).

A.2 Red Wine Dataset

In this section, we compare Algorithm 1 to ridge regression and the nonconvex relaxation of Brückner and Scheffer (2011) using the red wine dataset (Cortez et al., 2009). The red wine dataset contains 1599 instances each with 11 features. Each feature is a sensory or physiochemical measurement for wine. The response variable is a wine rating out of 10 points, where 10 is the best rating possible.

We place ourselves in the position of wine producers, who may wish to increase the rating of their wine by submitting fake input data. We assume that a wine producer is happy with the rating for their wine if it is greater than or equal to a threshold, $t \in [0, 10]$. Thus, if the true output label associated with a wine is greater than or equal to the threshold, then the target label of the agent is identical to the true output label. Otherwise, the target output label of the agent is set to t. Formally:

$$z_i = \max\{y_i, t\} \tag{A.3}$$

Similar to experiments conducted on the medical personals costs dataset, we perform 10-fold cross validation on the dataset and average the MSE of each approach over each fold for $\gamma \in [1 \times 10^{-5}, 0.2]$. As before, a ridge regression hyperparameter is selected for each γ by grid search on 8 logarithmically spaced points in the interval $[1 \times 10^{-5}, 1000]$. We define two different data providers, each with a different threshold: a modest data provider with $t_{\text{modest}} = 6$, and a severe data provider with $t_{\text{severe}} = 8$. The results of the experiments are shown in Figure A.1. As in the case of the medical personal costs dataset, Algorithm 1 outperforms other approaches for all values of γ. For values of $\gamma > 0.1$, we observe that, for modest data providers, our algorithm is at least 0.1 points more accurate than ridge regression on average. For severe data providers, we observe an even greater difference. Meanwhile, for severe data providers, Algorithm 1 is more than an entire point more accurate than ridge regression on average.

A.3 Run Time Comparison

Lastly, we compare the run time of `SDP-BISECT` to that of the interior point method for finding local solutions to the nonconvex relaxation of Brückner and Scheffer (2011).

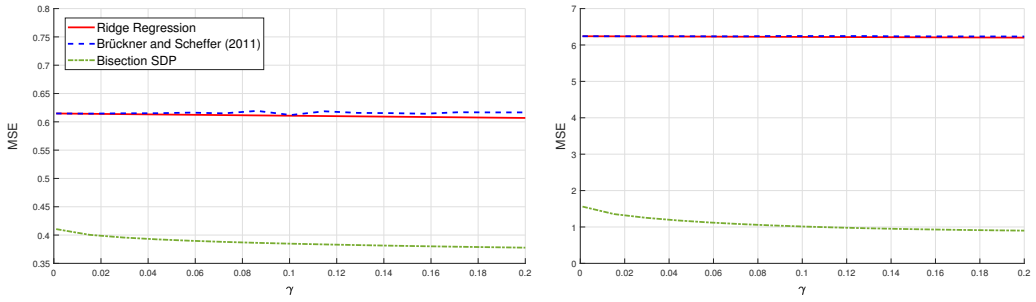

FIGURE A.1: A performance comparison between different algorithms run on the red wine dataset in which the target labels of each data provider are given by equation (A.3). The left plot corresponds to experiments run with t_{modest}, whilst the right plot corresponds experiments run with t_{severe}.

For these experiments, we fix $\gamma = 0.5$. Using the medical personal costs dataset, we run `SDP-BISECT` and the interior point method on training sets of varying sizes. For each training set size, we create 10 training sets by sampling uniformly from the entire dataset. This experiment was on run on an AMD Ryzen 1600 3.20GHz six-core processor on a single thread of execution. The error tolerances of the interior point method and `SDP-BISECT` are set to 1×10^{-2}.

Figure A.2 shows the mean run time of each algorithm for different problems sizes, with each error bar representing a 95% confidence interval according to the student t-distribution. Note that, in all cases, the interior point method quickly grows in run time as the scale of the problem increases, whilst `SDP-BISECT` has a similar running time for all problem scales tested. Whilst the interior point approach is faster on smaller problem instances, it quickly becomes apparent that `SDP-BISECT` is faster on larger instances. Also note that the run time of Algorithm `SDP-BISECT` is far more consistent, and has far less variance, especially as the problem scale grows.

It is worth noting that, for every problem scale, we use the same initial upper bound for q in our bisection search. More specifically, we take the entire medical personal costs dataset and upper bound q using the inner product of the training labels. As a result, the number of SDPs that need to be solved to achieve the same error tolerance for different problem scales is roughly equivalent. This partly explains why the performance of the bisection method stays relatively constant across different problem scales in our experiments.

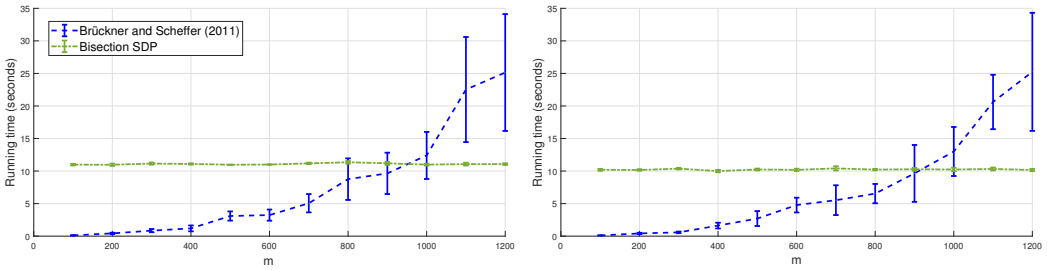

FIGURE A.2: A run time comparison between the interior point method approach and Algorithm 1 using the medical personal costs dataset. The plot on the left corresponds experiments run with $\mathcal{A}_{\text{modest}}$, whilst the right plot corresponds to experiments run with $\mathcal{A}_{\text{severe}}$.

Appendix B

Additional Pseudocode (Chapter 5)

In this appendix, we provide additional pseudocode regarding the subroutines used by RRSD, DRRSD, and BRRSD. The pseudocode for earliest(), which checks the earliest time step in which an agent can be assigned a service, is given in Algorithm 9. The pseudocode for available(), which checks whether a service can be assigned to an agent, is given in Algorithm 10. The pseudocode for overrun(), which checks whether an assignment of a service in one block of DRRSD will cause a blocking delay in the next block, is given in Algorithm 11. Lastly, the pseudocode for buildJobList(), which specifies the agent-service assignments for BRRSD to complete during the exploration phase, is given in Algorithm 12.

Algorithm 9: Different versions of the auxiliary process earliest

```
1  Function earliest(M, i, j, start, end):
2      for t = start, ..., end do
3          if available(M, i, j, t) then
4              return t
5          end
6      end
       // Failure state
7      return 0
8  End Function
9  Function earliest(M, i, j):
10     start = 1
11     end = T
12     return earliest(M, i, j, start, end)
13 End Function
```

Algorithm 10: Different versions of the auxiliary process available

```
1  Function available(M, i_1, j, t_0):
2      if M(t_0, i_1) ≠ 0 then
3          return false
4      end
5      for i = 1, ..., n do
6          for t = t_0 − D_{i,j} + 1, ..., t_0 + D_{i,j} − 1 do
7              if M(t, i) = j then
8                  return false
9              end
10         end
11     end
12     return true
13 End Function
14 Function available(M, i, j, start, end):
15     for t = start, ..., end do
16         if available(M, i, j, t) then
17             return true
18         end
19     end
20     return false
21 End Function
22 Function available(M, i, j):
23     start = 1
24     end = T
25     return available(M, i, j, start, end)
26 End Function
```

Algorithm 11: The auxiliary process overrun

```
1  Function overrun(M, i, j, t, end):
2      if t + D_{i,j} − 1 ≥ end then
3          return false
4      end
5      return true
6  End Function
```

Algorithm 12: The auxiliary process buildJobList

```
1  Function buildJobList(n, s, repeat):
2      list = []
3      for i = 1, ..., n do
4          for j = 1, ..., s do
5              for k = 1, ..., repeat do
6                  list.append((i, j))
7              end
8          end
9      end
10 End Function
```

References

Atilla Abdulkadiroglu and Tayfun Sonmez. Random serial dictatorship and the core from random endowments in house allocation problems. *Econometrica*, 66(3): 689–701, 1998. ISSN 00129682, 14680262.

Rediet Abebe, Richard Cole, Vasilis Gkatzelis, and Jason D. Hartline. A truthful cardinal mechanism for one-sided matching. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2096–2113, 2020.

Marek Adamczyk, Piotr Sankowski, and Qiang Zhang. Efficiency of truthful and symmetric mechanisms in one-sided matching. In *Algorithmic Game Theory*, pages 13–24, 2014. Springer Berlin Heidelberg. ISBN 978-3-662-44803-8.

Alekh Agarwal, Miroslav Dudik, and Zhiwei Steven Wu. Fair regression: Quantitative definitions and reduction-based algorithms. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 120–129. PMLR, 2019.

Shipra Agrawal, Nikhil R. Devanur, and Lihong Li. An efficient algorithm for contextual bandits with knapsacks, and an extension to concave objectives. In *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 4–18, 2016. PMLR.

Saeed Alaei, Pooya Jalaly Khalilabadi, and Eva Tardos. Computing equilibrium in matching markets. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, EC '17, page 245–261, 2017. Association for Computing Machinery. ISBN 9781450345279.

Alexia Atsidakou, Orestis Papadigenopoulos, Soumya Basu, Constantine Caramanis, and Sanjay Shakkottai. Combinatorial blocking bandits with stochastic delays. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 404–413. PMLR, 2021.

C. Audet, G. Savard, and W. Zghal. New branch-and-cut algorithm for bilevel linear programming. *Journal of Optimization Theory and Applications*, 134(2):353–370, 2007. ISSN 1573-2878.

Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

Peter Auer, Pratik Gajane, and Ronald Ortner. Adaptively tracking the best bandit arm with an unknown number of distribution changes. In *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pages 138–158. PMLR, 2019.

Pranjal Awasthi, Natalie Frank, and Mehryar Mohri. Adversarial learning guarantees for linear hypotheses and neural networks. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 431–441. PMLR, 2020a.

Pranjal Awasthi, Natalie Frank, and Mehryar Mohri. On the rademacher complexity of linear hypothesis sets. *arXiv preprint arXiv:2007.11045*, 2020b.

Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. *J. ACM*, 65(3), 2018. ISSN 0004-5411.

Ashwinkumar Badanidiyuru, John Langford, and Aleksandrs Slivkins. Resourceful contextual bandits. In *Proceedings of The 27th Conference on Learning Theory*, volume 35 of *Proceedings of Machine Learning Research*, pages 1109–1134, 2014. PMLR.

Peter L. Bartlett, Stéphane Boucheron, and Gábor Lugosi. Model selection and error estimation. *Machine Learning*, 48(1):85–113, 2002. ISSN 1573-0565.

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

Soumya Basu, Orestis Papadigenopoulos, Constantine Caramanis, and Sanjay Shakkottai. Contextual blocking bandits. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 271–279. PMLR, 2021a.

Soumya Basu, Karthik Abinav Sankararaman, and Abishek Sankararaman. Beyond $log^2(t)$ regret for decentralized bandits in matching markets. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 705–715. PMLR, 2021b.

Soumya Basu, Rajat Sen, Sujay Sanghavi, and Sanjay Shakkottai. Blocking bandits. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018. ISSN 0031-3203.

Nicholas Bishop, Hau Chan, Debmalya Mandal, and Long Tran-Thanh. Adversarial blocking bandits. In *Advances in Neural Information Processing Systems*, volume 33, pages 8139–8149. Curran Associates, Inc., 2020a.

Nicholas Bishop, Hau Chan, Debmalya Mandal, and Long Tran-Thanh. Sequential blocked matching. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(5): 4834–4842, 2022.

Nicholas Bishop, Long Tran-Thanh, and Enrico Gerding. Optimal learning from verified training data. In *Advances in Neural Information Processing Systems*, volume 33, pages 9520–9529. Curran Associates, Inc., 2020b.

Anna Bogomolnaia and Hervé Moulin. A new solution to the random assignment problem. *Journal of Economic Theory*, 100(2):295–328, 2001. ISSN 0022-0531.

Craig Boutilier, Ioannis Caragiannis, Simi Haber, Tyler Lu, Ariel D Procaccia, and Or Sheffet. Optimal social choice functions: A utilitarian view. *Artificial Intelligence*, 227:190–213, 2015.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Michael Brückner and Tobias Scheffer. Stackelberg games for adversarial prediction problems. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, page 547–555, 2011. Association for Computing Machinery. ISBN 9781450308137.

Sébastien Bubeck and Nicolò Cesa-Bianchi. *Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems*. Now Foundations and Trends, 2012.

Samuel Burer and Renato DC Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, (2):329–357, 2003.

Samuel Burer and Renato DC Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming*, 103(3):427–444, 2005.

Apostolos N. Burnetas and Michael N. Katehakis. Optimal adaptive policies for sequential allocation problems. *Advances in Applied Mathematics*, 17(2):122–142, 1996. ISSN 0196-8858.

Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D. Procaccia, Nisarg Shah, and Junxing Wang. The unreasonable fairness of maximum nash welfare. *ACM Trans. Econ. Comput.*, 7(3), 2019. ISSN 2167-8375.

Sarah H. Cen and Devavrat Shah. Regret, stability and fairness in matching markets with bandit learners. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 8938–8968. PMLR, 2022.

Deepayan Chakrabarti, Ravi Kumar, Filip Radlinski, and Eli Upfal. Mortal multi-armed bandits. In *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008.

Ronshee Chawla, Abishek Sankararaman, Ayalvadi Ganesh, and Sanjay Shakkottai. The gossiping insert-eliminate algorithm for multi-agent bandits. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3471–3481. PMLR, 2020.

Yeon-Koo Che and Fuhito Kojima. Asymptotic equivalence of probabilistic serial and random priority mechanisms. *Econometrica*, 78(5):1625–1672, 2010.

Ning Chen, Xiaotie Deng, Hongyang Zhang, and Jie Zhang. Incentive ratios of fisher markets. In *Automata, Languages, and Programming*, pages 464–475, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-31585-5.

Thomas Chen, Xi Chen, Binghui Peng, and Mihalis Yannakakis. Computational hardness of the hylland-zeckhauser scheme. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2253–2268, 2022.

Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework and applications. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 151–159, 2013. PMLR.

Yiling Chen, Yang Liu, and Chara Podimata. Learning strategy-aware linear classifiers. In *Advances in Neural Information Processing Systems*, volume 33, pages 15265–15276. Curran Associates, Inc., 2020.

Miri Choi. Medical Cost Personal Datasets — Kaggle, 2018.

George Christodoulou, Aris Filos-Ratsikas, Søren Kristoffer Stiil Frederiksen, Paul W. Goldberg, Jie Zhang, and Jinshan Zhang. Social welfare in one-sided matching mechanisms. In *Autonomous Agents and Multiagent Systems*, pages 30–50, 2016. Springer International Publishing. ISBN 978-3-319-46882-2.

Julia Chuzhoy, Rafail Ostrovsky, and Yuval Rabani. Approximation algorithms for the job interval selection problem and related scheduling problems. *Mathematics of Operations Research*, 31(4):730–738, 2006.

Edward H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971. ISSN 00485829, 15737101.

Richard Cole, Vasilis Gkatzelis, and Gagan Goel. Mechanism design for fair division: Allocating divisible items without payments. In *Proceedings of the Fourteenth ACM Conference on Electronic Commerce*, EC '13, page 251–268, 2013. Association for Computing Machinery. ISBN 9781450319621.

Richard Combes, Mohammad Sadegh Talebi Mazraeh Shahi, Alexandre Proutiere, and marc lelarge. Combinatorial bandits revisited. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547 – 553, 2009. Smart Business Networks: Concepts and Empirical Evidence.

Daniel Cullina, Arjun Nitin Bhagoji, and Prateek Mittal. Pac-learning in the presence of evasion adversaries. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 228–239, 2018. Curran Associates Inc.

Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, page 99–108, 2004. Association for Computing Machinery. ISBN 1581138881.

Sanmay Das and Emir Kamenica. Two-sided bandits and the dating market. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, page 947–952, 2005. Morgan Kaufmann Publishers Inc.

Carlos Henrique Medeiros de Sabóia, Manoel Campêlo, and Susana Scheimberg. A computational study of global algorithms for linear bilevel programming. *Numerical Algorithms*, 35(2):155–173, 2004. ISSN 1572-9265.

Stephan Dempe. *Foundations of bilevel programming*. Springer Science & Business Media, 2002.

II Dikin. Iterative solution of problems of linear and quadratic programming. In *Doklady Akademii Nauk*, volume 174, pages 747–748. Russian Academy of Sciences, 1967.

Wenkui Ding, Tao Qiny, Xu-Dong Zhang, and Tie-Yan Liu. Multi-armed bandit with budget constraint and variable costs. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, AAAI'13, page 232–238. AAAI Press, 2013.

Werner Dinkelbach. On nonlinear fractional programming. *Management Science*, 13(7): 492–498, 1967.

Jinshuo Dong, Aaron Roth, Zachary Schutzman, Bo Waggoner, and Zhiwei Steven Wu. Strategic classification from revealed preferences. In *Proceedings of the 2018*

ACM Conference on Economics and Computation, EC '18, page 55–70, 2018. Association for Computing Machinery. ISBN 9781450358293.

Dmitriy Drusvyatskiy and Lin Xiao. Stochastic optimization with decision-dependent distributions, 2020.

Laurent El Ghaoui and Hervé Lebret. Robust solutions to least-squares problems with uncertain data. *SIAM Journal on Matrix Analysis and Applications*, 18(4):1035–1064, 1997.

Aris Filos-Ratsikas, Søren Kristoffer Stiil Frederiksen, and Jie Zhang. Social welfare in one-sided matchings: Random priority and beyond. In *Algorithmic Game Theory*, pages 1–12, 2014. Springer Berlin Heidelberg. ISBN 978-3-662-44803-8.

D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

Aurelien Garivier, Tor Lattimore, and Emilie Kaufmann. On explore-then-commit strategies. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Aurélien Garivier, Pierre Ménard, and Gilles Stoltz. Explore first, exploit next: The true shape of regret in bandit problems. *Mathematics of Operations Research*, 44(2): 377–399, 2019.

Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. In *Algorithmic Learning Theory*, pages 174–188, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-24412-4.

Aurélien Garivier and Olivier Cappé. The kl-ucb algorithm for bounded stochastic bandits and beyond. In *Proceedings of the 24th Annual Conference on Learning Theory*, volume 19 of *Proceedings of Machine Learning Research*, pages 359–376, 2011. PMLR.

Amir Globerson and Sam Roweis. Nightmare at test time: Robust learning by feature deletion. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 353–360, 2006. Association for Computing Machinery. ISBN 1595933832.

Theodore Groves. Incentives in teams. *Econometrica*, 41(4):617–631, 1973. ISSN 00129682, 14680262.

Tatiana V. Gruzdeva and Alexander S. Strekalovsky. On solving the sum-of-ratios problem. *Applied Mathematics and Computation*, 318:260–269, 2018. ISSN 0096-3003. Recent Trends in Numerical Computations: Theory and Algorithms.

Tat'yana Vladimirovna Gruzdeva and Elena Gennadievna Petrova. Numerical solution of a linear bilevel problem. *Computational Mathematics and Mathematical Physics*, 50(10):1631–1641, 2010.

Moritz Hardt, Nimrod Megiddo, Christos Papadimitriou, and Mary Wootters. Strategic classification. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, ITCS '16, page 111–122, 2016. Association for Computing Machinery. ISBN 9781450340571.

Hadi Hosseini, Kate Larson, and Robin Cohen. Matching with dynamic ordinal preferences. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Hadi Hosseini, Kate Larson, and Robin Cohen. Investigating the characteristics of one-sided matching mechanisms under various preferences and risk attitudes. *Autonomous Agents and Multi-Agent Systems*, 32(4):534–567, 2018.

Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I.P. Rubinstein, and J. D. Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, AISec '11, page 43–58, 2011. Association for Computing Machinery. ISBN 9781450310031.

Aanund Hylland and Richard Zeckhauser. The efficient allocation of individuals to positions. *Journal of Political Economy*, 87(2):293–314, 1979. ISSN 00223808, 1537534X.

Nicole Immorlica, Karthik Abinav Sankararaman, Robert Schapire, and Aleksandrs Slivkins. Adversarial bandits with knapsacks. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 202–219, 2019.

Adel Javanmard, Mahdi Soltanolkotabi, and Hamed Hassani. Precise tradeoffs in adversarial training for linear regression. In *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 2034–2078. PMLR, 2020.

Ying Ji, Yijun Li, and Pengyu Lu. A global optimization algorithm for sum of quadratic ratios problem with coefficients. *Applied Mathematics and Computation*, 218 (19):9965–9973, 2012.

Hongwei Jiao, Zhankui Wang, and Yongqiang Chen. Global optimization algorithm for sum of generalized polynomial ratios problem. *Applied Mathematical Modelling*, 37(1-2):187–197, 2013.

Sham M Kakade, Karthik Sridharan, and Ambuj Tewari. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. In *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008.

Satyen Kale, Chansoo Lee, and David Pal. Hardness of online sleeping combinatorial optimization problems. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Bala Kalyanasundaram and Kirk Pruhs. Online weighted matching. *Journal of Algorithms*, 14(3):478–488, 1993.

Varun Kanade, H. Brendan McMahan, and Brent Bryan. Sleeping experts and bandits with stochastic action availability and adversarial rewards. In *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 272–279, 2009. PMLR.

Varun Kanade and Thomas Steinke. Learning hurdles for sleeping experts. *ACM Trans. Comput. Theory*, 6(3), 2014. ISSN 1942-3454.

Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, page 587–596, 2011. Association for Computing Machinery. ISBN 9781450306911.

N Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

Zohar S Karnin and Oren Anava. Multi-armed bandits: Competing with optimal sequences. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358, 1990.

Samir Khuller, Stephen G. Mitchell, and Vijay V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science*, 127 (2):255–267, 1994. ISSN 0304-3975.

Robert Kleinberg, Alexandru Niculescu-Mizil, and Yogeshwer Sharma. Regret bounds for sleeping experts and bandits. *Machine Learning*, 80(2):245–272, 2010. ISSN 1573-0565.

Levente Kocsis and Csaba Szepesvári. Discounted ucb. In *2nd PASCAL Challenges Workshop*, volume 2, pages 51–134, 2006.

Fuhito Kojima and Mihai Manea. Incentives in the probabilistic serial mechanism. *Journal of Economic Theory*, 145(1):106–123, 2010. ISSN 0022-0531.

V. Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Transactions on Information Theory*, 47(5):1902–1914, 2001.

Branislav Kveton, Zheng Wen, Azin Ashkan, and Csaba Szepesvari. Tight Regret Bounds for Stochastic Combinatorial Semi-Bandits. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 535–543, 2015. PMLR.

T.L Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985. ISSN 0196-8858.

Tor Lattimore. Refining the confidence level for optimistic bandit strategies. *Journal of Machine Learning Research*, 19(20):1–32, 2018.

Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

Tosca Lechner and Ruth Urner. Learning losses for strategic classification, 2022.

Lydia T. Liu, Horia Mania, and Michael Jordan. Competing bandits in matching markets. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1618–1628. PMLR, 2020.

Lydia T. Liu, Feng Ruan, Horia Mania, and Michael I. Jordan. Bandit learning in decentralized matching markets. *Journal of Machine Learning Research*, 22(211):1–34, 2021.

Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, page 641–647, 2005. Association for Computing Machinery. ISBN 159593135X.

Celestine Mendler-Dünner, Juan C. Perdomo, Tijana Zrnic, and Moritz Hardt. Stochastic optimization for performative prediction. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, 2020. Curran Associates Inc. ISBN 9781713829546.

John P Miller, Juan C Perdomo, and Tijana Zrnic. Outside the echo chamber: Optimizing the performative risk. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7710–7720. PMLR, 2021.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

Omar Montasser, Steve Hanneke, and Nathan Srebro. Vc classes are adversarially robustly learnable, but only improperly. In *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pages 2512–2530. PMLR, 2019.

Le Dung Muu and Nguyen Van Quy. A global optimization method for solving convex quadratic bilevel programming problems. *Journal of Global Optimization*, 26 (2):199–219, 2003. ISSN 1573-2916.

John F Nash Jr. The bargaining problem. *Econometrica: Journal of the econometric society*, pages 155–162, 1950.

Blaine Nelson, Benjamin I. P. Rubinstein, Ling Huang, Anthony D. Joseph, Steven J. Lee, Satish Rao, and J. D. Tygar. Query strategies for evading convex-inducing classifiers. *Journal of Machine Learning Research*, 13(44):1293–1332, 2012.

Gergely Neu and Michal Valko. Online combinatorial optimization with stochastic decision sets and adversarial losses. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

Ronald Ortner, Daniil Ryabko, Peter Auer, and Rémi Munos. Regret bounds for restless markov bandits. In *Algorithmic Learning Theory*, pages 214–228, 2012. Springer Berlin Heidelberg.

Orestis Papadigenopoulos and Constantine Caramanis. Recurrent submodular welfare and matroid blocking semi-bandits. In *Advances in Neural Information Processing Systems*, volume 34, pages 23334–23346. Curran Associates, Inc., 2021.

David C. Parkes and Ariel D. Procaccia. Dynamic social choice with evolving preferences. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, AAAI'13, page 767–773. AAAI Press, 2013.

Juan Perdomo, Tijana Zrnic, Celestine Mendler-Dünner, and Moritz Hardt. Performative prediction. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7599–7609. PMLR, 2020.

Ariel D. Procaccia and Jeffrey S. Rosenschein. The distortion of cardinal preferences in voting. In *Cooperative Information Agents X*, pages 317–331, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-38570-7.

Shao-Jian Qu, Ke-Cun Zhang, and Jia-Kun Zhao. An efficient algorithm for globally minimizing sum of quadratic ratios problem with nonconvex quadratic constraints. *Applied Mathematics and Computation*, 189(2):1624–1636, 2007. ISSN 0096-3003.

Anshuka Rangi, Massimo Franceschetti, and Long Tran-Thanh. Unifying the stochastic and the adversarial bandits with knapsack. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 3311–3317. International Joint Conferences on Artificial Intelligence Organization, 2019.

Aadirupa Saha, Pierre Gaillard, and Michal Valko. Improved sleeping bandits with stochastic action sets and adversarial rewards. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8357–8366. PMLR, 2020.

Abishek Sankararaman, Soumya Basu, and Karthik Abinav Sankararaman. Dominate or delete: Decentralized competing bandits in serial dictatorship. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 1252–1260. PMLR, 2021.

Siegfried Schaible and Jianming Shi. Fractional programming: The sum-of-ratios case. *Optimization Methods and Software*, 18(2):219–229, 2003.

Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 5019–5031, 2018. Curran Associates Inc.

Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In *Computational Learning Theory*, pages 416–426, 2001. Springer Berlin Heidelberg.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright. *Optimization for Machine Learning*. The MIT Press, 2011. ISBN 026201646X.

Alexander S. Strekalovsky and Andrei V. Orlov. *Global Search for Bilevel Optimization with Quadratic Data*, pages 313–334. Springer International Publishing, 2020. ISBN 978-3-030-52119-6.

Ravi Sundaram, Anil Vullikanti, Haifeng Xu, and Fan Yao. Pac-learning for strategic classification. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9978–9988. PMLR, 2021.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Csaba Szepesvári. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010.

K. C. Toh, M. J. Todd, and R. H. Tütüncü. Sdpt3 — a matlab software package for semidefinite programming, version 1.3. *Optimization Methods and Software*, 11(1-4): 545–581, 1999.

Liang Tong, Sixie Yu, Scott Alfeld, and Yevgeniy Vorobeychik. Adversarial regression with multiple learners. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, pages 4953–4961, 2018.

Long Tran-Thanh, Archie Chapman, Alex Rogers, and Nicholas Jennings. Knapsack based optimal policies for budget–limited multi–armed bandits. *Proceedings of the AAAI Conference on Artificial Intelligence*, 26(1):1134–1140, 2012.

L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984. ISSN 0001-0782.

V. N. Vapnik and A. Ya. Chervonenkis. *On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities*, pages 11–30. Springer International Publishing, 2015. ISBN 978-3-319-21852-6.

William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37, 1961. ISSN 00221082, 15406261.

Heinrich Von Stackelberg. *Market structure and equilibrium*. Springer Science & Business Media, 2010.

Yevgeniy Vorobeychik and Murat Kantarcioglu. Adversarial machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–169, 2018.

Jiali Wang, He Chen, Rujun Jiang, Xudong Li, and Zihao Li. Fast algorithms for stackelberg prediction game with least squares loss. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10708–10716. PMLR, 2021.

Jiali Wang, Wen Huang, Rujun Jiang, Xudong Li, and Alex L Wang. Solving stackelberg prediction game with least squares loss via spherically constrained least squares reformulation. *arXiv preprint arXiv:2206.02991*, 2022.

Zihe Wang, Zhide Wei, and Jie Zhang. Bounded incentives in manipulating the probabilistic serial rule. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34 (02):2276–2283, 2020.

Chen-Yu Wei, Yi-Te Hong, and Chi-Jen Lu. Tracking the best expert in non-stationary stochastic environments. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

P. Whittle. Restless bandits: activity allocation in a changing world. *Journal of Applied Probability*, 25(A):287–298, 1988.

Gerhard J Woeginger. The open shop scheduling problem. In *35th Symposium on Theoretical Aspects of Computer Science (STACS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

Yong Xia, Shu Wang, and Ruey-Lin Sheu. S-lemma with equality and its applications. *Mathematical Programming*, 156(1):513–547, 2016. ISSN 1436-4646.

Dong Yin, Ramchandran Kannan, and Peter Bartlett. Rademacher complexity for adversarially robust generalization. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7085–7094. PMLR, 2019.

Hanrui Zhang and Vincent Conitzer. Incentive-aware pac learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(6):5797–5804, 2021.

Lin Zhou. On a conjecture by gale about one-sided matching problems. *Journal of Economic Theory*, 52(1):123–135, 1990. ISSN 0022-0531.