Optimization for AI-Driven Sequential Decision-Making

Thesis by
Lin An

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy in Algorithms, Combinatorics, and Optimization

CARNEGIE MELLON UNIVERSITY
Tepper School of Business
Pittsburgh, Pennsylvania, USA

2026
Defended March 27

Lin An
ORCID: 0009-0005-7840-2194

ACKNOWLEDGEMENTS

I confess that I have rarely read the main body of other people's theses, but I have read quite a few acknowledgements. A person's academic contributions are recorded in the theorems, propositions, figures, and proofs that follow. But the acknowledgements are where I feel the presence of a real human being: their frustrations when a proof refuses to work, their hobbies and friendships outside of research, and their gratitude toward the people who helped them along the way.

When I felt frustrated or stuck during my PhD, reading these acknowledgements often gave me courage. They reminded me that I was not alone in this journey. And whenever I experienced moments that I knew I would cherish, I sometimes imagined the day I would write this acknowledgement myself. Still, now that the moment has arrived, it feels a little surreal.

To whoever is reading this: I hope these words can pass on some of the same encouragement, affection, and strength that others have given to me. And of course, I also hope that the hundreds of pages that follow might be helpful to you in some way.

Looking back, I can honestly say that the five years of my PhD have been the happiest five years of my life so far (though I certainly hope the rest of my life will be even more fulfilling). At some point I realized that feeling this way about one's PhD is itself a privilege.

First and foremost, I want to thank my advisors, Andrew Li and Benjamin Moseley. They are the two people who have helped me the most throughout my PhD, and I am deeply grateful for everything they have done for me.

Andrew is the person who shaped my research taste. Through working with him, I learned that research is not only about proving theorems. More importantly, it is about finding problems that truly matter: problems that are impactful for our field and, ideally, for the real world. Andrew has an extraordinary instinct for identifying such problems and modeling them in the right way. I have often felt that he is one of the very best researchers at discovering the right problems and framing them clearly. Watching him work has been one of the most valuable parts of my PhD training. My own research style has been profoundly influenced by Andrew's taste, and I expect it will continue to shape how I approach research throughout my career.

Outside of research itself, Andrew also taught me many of the softer skills needed to succeed in academia. These ranged from how to give a compelling presentation

to surprisingly practical advice about fly-outs, even down to what kind of shoes to wear. I still remember his recommendation: dark brown, pointed, real leather, with sole protection. At first I did not expect this kind of advice to matter very much, but it turned out to be unexpectedly valuable at many moments in my academic career. Andrew was always generous with his time and energy, and our meetings were frequent, lively, and extremely productive. Outside of research, he was also someone I could talk to openly about life and career. I am grateful for the many conversations we shared and for the support he gave me during both easy and difficult times.

Ben was the first professor in the group whom I interacted with, and if it were not for his encouragement, I might never have come to Tepper. From the very beginning, he welcomed me into the group and made me feel that this was a place where I could grow as a researcher. Throughout our collaborations, Ben's enthusiasm for mentoring students has had a strong influence on me. He genuinely cares about what his students want to pursue and always tries to find ways to support them.

One thing that always stood out to me about Ben is how approachable and encouraging he is when discussing research. Our meetings were often relaxed and conversational, yet somehow they were always productive. Even when I arrived with vague ideas or half-formed thoughts, Ben would patiently listen and then point out directions that were both insightful and practical to try. Many times, these suggestions turned out to be exactly what I needed to move the project forward. His ability to quickly see the core of a problem and suggest promising directions has been invaluable to me. I am very grateful for his generosity with his time, his thoughtful guidance, and the encouragement he has given me throughout my PhD.

I would also like to thank the rest of my committee members: Ravi, Joy, and Evelyn. Ravi played an important role early in my PhD by bringing together the group for my first summer project and by introducing me to Glance, which later became a fruitful collaboration. I am also grateful for his advice about both research and academic life. Joy introduced me to ideas and perspectives from marketing that were new to me and has been a wonderful collaborator. Evelyn provided thoughtful feedback during my proposal and defense, and I very much look forward to crossing paths again in our careers.

I am grateful to the faculty in the operations research community at Tepper, Gérard, John, Willem, Javier, Fatma, Karan, and Emily, for creating such a supportive environment. Their kindness and encouragement made this a wonderful place to

grow as a researcher.

I would also like to thank two mentors who influenced me before I arrived at CMU. Solomon Friedberg, my undergraduate advisor, inspired me to pursue research in the first place. The time and effort he devoted to mentoring an undergraduate student were extraordinary and had a lasting impact on me. Yuri Faenza, my research mentor during my master's studies, introduced me to the field of operations research and patiently guided me through my first research project, even though at that time I had very little experience in optimization.

Next, I want to thank my fellow PhD students at Tepper. Anthony, thank you for being such a great friend. You have listened patiently to my complaints about research and life and have always been there with encouragement. As you often joked, I could always "lean on" you, and that turned out to be very true. Thank you also for organizing so many gatherings that brought people together; without you, our PhD community would probably have hung out much less often.

I am grateful to Aidin and Weizhong for sharing the early struggles of classes and qualifying exams. I would also like to thank Kyra, Su, Thomas, Violet, Yuyan, Özgün, Daniel de Roux, Mik, Heather, Neha, Neda, and Savannah for being role models when I first joined the program. The time we spent together, especially the many game nights in the lounge, created memories that I will always treasure. To Nilsu, Siyue, Vrishabh, Seba, Maca, Helia, Stephen, Daniel Yamin, Emily, Aurora, Milkis, Mogu, Mian, and Juan, thank you for the friendship and the moments we shared together, and I wish you all the best in your future endeavors.

I would also like to thank my friends outside of Tepper. Chengyue He, I am incredibly lucky to have you as a friend. Since we met and worked together at Columbia, we have gone through the PhD application process, the PhD years, and the faculty job market alongside each other. Your support throughout all of this has meant a great deal to me. Our shared enthusiasm for poker has also been an important part of our friendship, from discussing poker strategy to sharing poker stories to going on poker trips together. Those trips brought me some of the most joyful and memorable moments of these years.

Ziyu Huang and Jieqi Di, thank you for your friendship since our undergraduate days. I feel lucky that we ended up working in the same field, which gave us many opportunities to reconnect through conferences and academic life. I am also grateful that in New York and Atlanta I always knew I had a place to stay because of you.

Hao Yang and Junjie Xiao, thank you for being my friends since high school and for being present at important milestones in my life. Shuai Guo, you have been my closest friend for many years. The laughter we shared, the challenges we faced together, and the important moments we experienced along the way mean more to me than I can fully express.

Finally, I want to thank my family.

To my grandparents, thank you for the love and care you gave me during my childhood. Before this year we had not seen each other for five years, and I know that distance was not easy. Seeing you again this year meant a great deal to me, and I hope we will have many more chances to spend time together.

To Vicky Han, thank you for your unwavering support, your care, your understanding, and your love throughout this journey. Since we have been together, you have been my closest companion, and I have always felt that we face life as a team. The academic job market was a difficult and uncertain period, and whenever I felt lost or discouraged, you were always there for me. Every obstacle I faced, you treated as if it were your own. The moments we shared, both joyful and difficult, are among the most meaningful in my life. Without your support, I could not have reached this point.

To my parents, Xuewei Dong and Chengbin An: words cannot fully express my gratitude for your unconditional love. Even though I know you would have preferred me to stay closer to home, you supported my decision to pursue opportunities abroad because you believed it would lead to a better future for me. You have always been my strongest foundation, giving me the freedom and confidence to pursue what I truly wanted. Everything I have achieved rests on the support you have given me throughout my life.

For the past five years, whenever I received kindness, support, or encouragement, from close friends or even from people I met only briefly, I sometimes imagined the day I would write this acknowledgement. Now that it is finally written, it feels like the closing of an important chapter of my life. And I could not have reached this moment without all of you.

ABSTRACT

Modern operations increasingly rely on artificial intelligence (AI) models, ranging from demand forecasts and learned value functions to transformer architectures, to guide sequential decision-making under uncertainty. While these models provide rich representations of demand, preferences, and resource value, incorporating them into optimization raises challenges. Predictive accuracy may be unknown or time-varying, model outputs may be misspecified, and expressive architectures can induce computationally intractable decision problems. This thesis develops optimization for AI-driven sequential decision-making, focusing on algorithms that translate predictive structure into decisions equipped with robustness, computational efficiency, and provable performance guarantees.

The first part studies real-time personalization under transformer architectures. Transformers capture economically meaningful interaction effects such as variety seeking, substitution, and complementarity, but optimizing recommendations under general attention models leads to NP-hard combinatorial problems. We identify a structured class of simple transformers whose induced objectives admit exploitable low-rank structure, and we develop near-optimal recommendation algorithms with sublinear-time complexity under mild assumptions. These results demonstrate that expressive AI models can be reconciled with tractable and theoretically grounded optimization.

The second part considers online resource allocation under uncertain arrivals, where AI systems provide predictions in the form of shadow prices for resources. We design algorithms that leverage predicted dual variables without requiring knowledge of their accuracy. The resulting policies achieve a principled balance between robustness and consistency: they maintain worst-case guarantees while converging to near-optimal performance as prediction quality improves.

The third part examines inventory control under nonstationary demand. We propose adaptive policies for the nonstationary newsvendor problem that achieve order-optimal regret without assuming a bound on demand drift. Incorporating generic AI-generated demand predictions, we show how to combine realized observations with predictive signals to obtain guarantees that remain robust to arbitrary prediction error while improving when predictions are informative.

Across personalization, resource allocation, and inventory control, this thesis provides

a unifying framework for integrating AI models into sequential decision-making. By identifying structural properties that enable tractable optimization and by designing algorithms that remain robust to imperfect model guidance, the results establish theoretical foundations for AI-driven operations and revenue management.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Motivation: AI models and sequential decisions

A broad set of operations problems can be viewed as *sequential decision-making under uncertainty*: a decision-maker repeatedly chooses actions (e.g., order quantities, accept/reject and allocation decisions, or recommended items) in the presence of evolving demand, arrivals, and user behavior. In recent years, these decisions have been increasingly guided by *AI models*, ranging from demand forecasts and learned value signals to modern neural architectures such as transformers. This creates an opportunity and a challenge.

The opportunity is that AI models can capture rich patterns from large-scale data and thereby provide powerful guidance. The challenge is that using AI guidance *within* decision-making is rarely straightforward. First, predictive accuracy is often unknown a priori and can vary over time, so blindly following a model can be harmful (e.g., when forecasts drift, arrivals become nonstationary, or user behavior shifts). Second, even when a model is accurate, its expressive power can turn the downstream optimization problem into a hard combinatorial object. A concrete example arises in personalization: moving from embedding-based objectives (which permit extremely fast retrieval at scale) to transformer-based objectives breaks the synergy between learning and optimization, and the induced recommendation problem can become NP-hard and difficult even to approximate efficiently. This thesis develops optimization foundations for *AI-driven sequential decision-making*: how to transform AI guidance into decisions that are (i) computationally efficient and (ii) supported by end-to-end performance guarantees.

A unifying viewpoint. Across the three chapters, AI enters decision-making in three increasingly expressive forms: (i) as a learned objective for set/sequence recommendation (transformers); (ii) as predicted dual variables (shadow prices) that summarize future opportunity costs in online allocation; and (iii) as generic predictions of demand statistics in inventory control. In each case, the central methodological question is the same:

How can operations fully leverage the predictive and representational

power of modern AI models, while ensuring that the resulting decisions remain robust, tractable, and equipped with performance guarantees under sequential uncertainty?

1.2 Thesis themes: predictive structure, adaptivity, and algorithmic guarantees

This thesis develops a unified perspective on AI-driven sequential decision-making through three recurring principles.

(1) Exploitable predictive structure. Modern AI models provide rich predictive and representational signals, but these signals are useful for decision-making only when their induced structure can be identified and exploited algorithmically. Across the three chapters, AI guidance enters in different forms: as a transformer-based objective for recommendation, as predicted shadow prices in online resource allocation, and as demand forecasts in nonstationary inventory control.

In Chapter 2, the transformer model induces a combinatorial objective over recommended sets. While general transformer objectives render the optimization problem NP-hard and difficult even to approximate , the chapter identifies a structured subclass, *simple transformers*, whose low-rank attention structure enables near-optimal optimization in sublinear time. Thus, expressive neural models can be reconciled with tractable decision rules by isolating and leveraging their latent algebraic structure.

In Chapter 3, structure appears in the dual formulation of the allocation problem. Predictions are interpreted as estimates of optimal Lagrangian multipliers (shadow prices). This dual viewpoint provides a principled interface between AI-generated predictions and sequential allocation decisions.

In Chapter 4, structure arises in the temporal evolution of demand. Nonstationarity is parameterized via a variation budget, capturing the intrinsic difficulty of learning in changing environments. This structural characterization enables tight regret bounds that separate the effects of demand drift and stochastic noise.

(2) Adaptivity to unknown model quality. A central challenge in AI-driven decision-making is that prediction accuracy is typically unknown and may vary over time. Blindly following model outputs can therefore lead to poor performance. Chapters 3 and 4 develop algorithms that adapt to prediction quality without requiring prior knowledge of its accuracy.

In online resource allocation, predictions are given as shadow prices, and accuracy is measured as distance to the true optimal dual variables. The proposed algorithm achieves a tight consistency–robustness tradeoff: it preserves worst-case guarantees when predictions are inaccurate, while converging to near-optimal performance as prediction error diminishes. In the nonstationary newsvendor, generic demand forecasts are incorporated into ordering decisions without assuming any bound on their error. The resulting policy matches the best of learning-only and prediction-only benchmarks up to logarithmic factors, achieving order-optimal regret under both nonstationarity and prediction error.

(3) End-to-end algorithmic guarantees. A recurring tension in operations is that behaviorally realistic models can induce computationally intractable decision problems. This thesis emphasizes algorithms that translate AI guidance into implementable policies with formal guarantees.

For personalization, Chapter 2 develops near-optimal algorithms with sublinear decision complexity, reconciling transformer expressiveness with real-time requirements. For online allocation and inventory control, Chapters 3 and 4 provide tight competitive and regret guarantees under unknown arrival models and unknown nonstationarity, respectively. Across all settings, the results establish end-to-end guarantees that integrate statistical learning, algorithmic design, and sequential decision-making.

Taken together, these principles demonstrate how operations can harness the full power of modern AI: by identifying exploitable predictive structure, designing algorithms that adapt to model quality, and maintaining tractability and provable performance under uncertainty.

1.3 Chapter overview and contributions

Chapter 2: Personalization with simple transformers

Chapter 2 studies the problem of selecting a set (or sequence) of up to k items for a user when preferences are modeled by a transformer. Embedding models enable fast optimization via approximate nearest neighbor methods, but they fail to capture interaction effects between items in the recommended set. Transformers can represent such interactions, yet they induce a downstream optimization problem that becomes NP-hard in general, undermining real-time decision-making. The chapter therefore asks two questions: (i) is there a nontrivial subclass of transformers that

admits fast (ideally sublinear in n) optimization with meaningful guarantees, and (ii) does restricting to such a subclass sacrifice modeling power?

The chapter resolves this tension through the notion of *simple transformers*, architectures consisting of a single self-attention layer that strike a deliberate balance between modeling expressiveness and optimization tractability. On the modeling side, we show that simple transformers can represent key behavioral patterns studied in economics, marketing, and operations management, including sequential variety-seeking behavior and pairwise complementarity and substitution effects. These interaction patterns cannot be captured by pure embedding-based models, highlighting the necessity of moving beyond independent-item representations.

On the algorithmic side, we design a two-phase retrieval-and-ranking framework that approximately solves the induced recommendation problem. Under mild rank assumptions on the attention structure, the algorithm achieves near-optimal recommendations with expected amortized runtime sublinear in the catalog size n. Sublinear complexity is essential in large-scale personalization systems, where decisions must be computed instantly upon user interaction.

We demonstrate the effectiveness of our approach through an empirical study on datasets from Spotify and Trivago. Our experiment results show that (1) simple transformers can model/predict user preferences substantially more accurately than non-transformer models and nearly as accurately as more complex transformers, and (2) our algorithm completes simple-transformer-based recommendation tasks quickly and effectively.

Chapter 2 is based on [6], joint work with Andrew A. Li, Vaisnavi Nemala, and Gabriel Visotsky.

Chapter 3: Online resource allocation with predicted shadow prices

Chapter 3 considers the Online Resource Allocation problem, a general model for sequential decisions with limited resources. At each time period, a request arrives with an action set; choosing an action yields reward and consumes resources, and the objective is to maximize total reward subject to a global resource budget. This model captures applications across revenue management, online retail, and online advertising.

The contribution of Chapter 3 is to incorporate predictions in a way that is robust to unknown prediction accuracy and unknown arrival model. Predictions are introduced in the *dual space* as estimated resource shadow prices, and prediction accuracy

is defined as a distance between predicted and true shadow prices. The chapter develops a tight lower bound describing the best achievable use of predictions, informally, an optimal "follow when accurate, ignore when inaccurate" behavior without knowing accuracy in advance, and proposes an algorithm that matches this bound. The algorithm is then empirically validated using large-scale real data from a retail setting.

Chapter 3 is based on [4], joint work with Andrew A. Li, Benjamin Moseley, and Gabriel Visotsky. In addition to establishing tight theoretical guarantees, the chapter includes a large-scale empirical evaluation using retail data from H&M, highlighting the practical impact of prediction-augmented allocation policies.

Chapter 4: Nonstationary newsvendor with (and without) predictions

Chapter 4 studies inventory control under nonstationary demand. The problem consists of a sequence of ordering decisions, where the underlying demand distribution can evolve over time. Nonstationarity is captured through a variation-based measure that quantifies the magnitude of demand drift, and performance is evaluated via regret relative to a dynamic benchmark that knows the demand distribution in each period.

The chapter first provides a complete characterization of the nonstationary newsvendor *without* predictions. It establishes matching lower and upper bounds and develops a policy that achieves order-optimal regret without prior knowledge of the degree of nonstationarity.

The chapter then incorporates *generic predictions* of mean demand observed before each ordering decision. No structural assumptions are imposed on how predictions are generated, and prediction quality is parameterized through cumulative prediction error. The central challenge is to leverage predictions optimally without knowing their accuracy in advance. To address this, the chapter proposes a prediction-robust policy that achieves near-optimal regret across regimes: it remains robust when predictions are inaccurate, and improves automatically when predictions are sufficiently informative, matching the better of learning-only and prediction-only strategies up to logarithmic factors.

Chapter 4 is based on [5], joint work with Andrew A. Li, Benjamin Moseley, and R. Ravi. Extensive numerical experiments, including evaluations on real-world demand data from Wikipedia and Rossmann drug store, validate the theoretical guarantees and demonstrate the practical benefits of combining adaptivity to nonstationarity

with robustness to prediction error.

1.4 Connections across chapters

This section highlights how the three chapters form a coherent progression.

From expressive AI objectives to tractable optimization. Chapter 2 begins with the most expressive AI component: transformer-based objectives for recommendation. It formalizes the computational barrier for general transformers and shows that by identifying the right structural subclass (simple transformers), one can obtain both modeling richness (capturing interaction effects) and tractable near-optimal optimization in sublinear time.

From tractable structure to robust use of AI guidance. Chapters 3 and 4 focus on a different role of AI: as predictive guidance for online decisions. A key lesson is that predictions are valuable precisely when algorithms can *adapt* to their quality without knowing it. Chapter 3 implements this through predicted dual variables (shadow prices) and proves tight guarantees under unknown arrival models and prediction accuracy. Chapter 4 implements an analogous principle in inventory control under nonstationary demand, combining learning from realized outcomes with prediction signals.

A unified perspective. Taken together, the chapters develop a unified perspective on AI-driven sequential decision-making. Across the different settings studied in this thesis, a common methodological pattern emerges:

1. Identify structural properties of AI models (or their outputs) that preserve practical modeling power;
2. Design algorithms that exploit these structures to achieve computational tractability;
3. Ensure robustness by adapting to unknown and potentially time-varying model quality;
4. Provide end-to-end guarantees (approximation and sublinear-time decisions in personalization; tight regret or competitive guarantees in online allocation and inventory).

1.5 Organization of the thesis

The remainder of the thesis is organized as follows. Chapter 2 studies real-time personalization under transformer objectives and develops near-optimal sublinear-time algorithms for simple transformers. Chapter 3 studies online resource allocation with predicted shadow prices under unknown arrival models and derives tight robustness–consistency guarantees. Chapter 4 studies the nonstationary newsvendor with and without predictions, providing order-optimal regret guarantees and prediction-error robustness. Chapter 5 concludes with directions for future research on optimization foundations for AI-driven sequential decision-making.

Chapter 2

NEAR-OPTIMAL PERSONALIZATION WITH SIMPLE TRANSFORMERS

Real-time personalization has advanced significantly in recent years, with platforms utilizing machine learning models to predict user preferences based on rich behavioral data on each individual user. Traditional approaches usually rely on embedding-based machine learning models to capture user preferences, and then reduce the final optimization task to nearest-neighbors, which can be performed extremely fast. However, these models struggle to capture complex user behaviors, which are essential for making accurate recommendations. Transformer-based models, on the other hand, are known for their practical ability to model sequential behaviors, and hence have been intensively used in personalization recently to overcome these limitations. However, optimizing recommendations under transformer-based models is challenging due to their complicated architectures. In this paper, we address this challenge by considering a specific class of transformers, showing its ability to represent complex user preferences, and developing efficient algorithms for real-time personalization.

We focus on a particular set of transformers, called simple transformers, which contain a single self-attention layer. We show that simple transformers are capable of capturing complex user preferences. We then develop an algorithm that enables fast optimization of recommendation tasks based on simple transformers. Our algorithm achieves near-optimal performance in sub-linear time. Finally, we demonstrate the effectiveness of our approach through an empirical study on datasets from Spotify and Trivago. Our experiment results show that (1) simple transformers can model/predict user preferences substantially more accurately than non-transformer models and nearly as accurately as more complex transformers, and (2) our algorithm completes simple-transformer-based recommendation tasks quickly and effectively.

2.1 Introduction

Personalization today is already immensely sophisticated. Media platforms, online retailers, and subscription services (just to name a few) capture rich data on their users in the form of their behavior and interactions with individual items/products. There are then two key ingredients: (1) this data is used *offline* to build machine-learning

(ML) based models of users' preferences, and then (2) these models are used in *real-time* to make personalized recommendations.

Zooming out from personalization for just a moment, the most jarring improvements in ML models over the last few years have been in generative models for language, and specifically *transformer*-based models (e.g. the "T" in *ChatGPT*) that have proven to be extremely accurate in modeling *sequential* data. Perhaps unsurprisingly, these same models are well-equipped for personalization. To fix a concrete example, suppose an *Instacart* user is in the process of shopping online for groceries. This user's behavior consists of interactions with grocery items: browsing through items, viewing a subset of these in more detail, and adding a subset of these to their shopping cart. The task of learning this user's preferences essentially amounts to predicting their future interactions. The important observation here is that the user's behavior is naturally sequential, and so this prediction task is similar to completing a sentence, where the "words" are the items themselves. This connection to language suggests that the same transformer-based models may succeed in learning preferences.

This is already being done in practice, often with substantial empirical success (e.g. by *Alibaba* [50], *Amazon* [105], *Spotify* [135], and *Wayfair* [131]). However, as examples of such successes become increasingly common, there is little principled guidance on how transformers "should" be used for real-time personalization. This is the problem we seek to address.

Real-Time Personalization, Before Transformers: To make the nature of this problem more precise, it is worth reviewing how real-time personalization is performed *without* transformers. Referring to the two key ingredients mentioned at the outset: first, the ML-based models of user preferences are, by and large, pure *embedding*-based models. Using past data, each item i is mapped to some element $v_i \in \mathbb{R}^d$ in such a way that (a) "similar" items are "close" together, and perhaps more formally, (b) each user can be represented as some $u \in \mathbb{R}^d$ so that the inner products $v_i^\top u$ fully represent the preference/affinity of the user for each item i.

These pure embedding models are not necessarily the most *accurate* models that can be estimated from past data, but they enable *fast* execution of the second ingredient, which is to optimize a set (or sequence) of items for each user in real time:

$$\begin{aligned} \max \quad & f(S, u) \\ \text{s.t.} \quad & S \subset [n], |S| \leq k. \end{aligned} \tag{2.1}$$

The (extremely general) formulation above simply highlights that real-time personalization consists of solving cardinality-constrained *set-optimization* (or *sequence-optimization*, whose equivalence to set-optimization we will discuss later on) problems where (a) the objective function is user-dependent, (b) the number of items n is potentially quite large, often in the hundreds of millions, and (c) a solution must be found in real-time, often just milliseconds. This is of course hopeless in general, but feasible when the objective function $f(S, u)$ results from a pure embedding models. In particular, $f(S, u)$ typically takes one of two forms:

1. An additive function:
$$f(S, u) = \sum_{i \in S} g(v_i^\top u),$$
for some non-decreasing function $g(\cdot)$. In this case, the optimal set S^* can be characterized as follows: if the number of items i such that $g(v_i^\top u) > 0$ is no greater than k, then S^* consists of all such items. Otherwise, S^* is the set of k items with the largest values of $g(v_i^\top u)$.
2. A monotone submodular function (in the argument S, for any u), such that each item is fully encoded by $v_i^\top u$, so that a constant-factor approximation can be found using greedy-style algorithms along with (multiple queries to) a black box which computes the item with largest inner product to a given point in $\mathbb{R}^d$ [67].

In both of the above cases, the pure embedding model essentially reduces the final real-time optimization task to one of *nearest neighbor* algorithms, where the goal is to find the items whose embeddings are most similar (under inner product similarity) to the user embedding. This task can be performed extremely fast, both theoretically (using *approximate nearest neighbor* algorithms with runtime sub-linear in n) and practically (given the nonstop engineering and improvement of commercial vector databases).

An Attempt to Introduce Transformers: Returning to transformers now, the natural opportunity is to improve the accuracy of the pure embedding models in representing user preferences: the embedding models essentially fail to capture the effects that items may have on each other when present in the same recommended set. Such effects are well-known to exist in multiple fields of study, as we will discuss shortly, and often representable via transformers. Unfortunately, the just-described synergy between the "upstream" user preference model estimated from data, and the "downstream" optimization of user-dependent sets of items completely breaks down

here. As we will see in the next section, this is true in a formal sense: Problem (2.1) is NP-hard, and likely hard to even approximate in linear time, if the objective $f(\cdot, u)$ is transformer-based. As of now, any practical implementation using transformers either applies a pure nearest-neighbor or greedy-style algorithm (essentially ignoring this hardness), or a random search heuristic (such as *Beam Search*).

In the spirit of developing a principled approach to transformer-based real-time personalization, this raises two major questions:

1. **Fast Optimization:** The formal hardness results just alluded to imply that achieving fast (ideally sub-linear in n) optimization of Problem (2.1) with any meaningful optimality guarantee is impossible if the objective $f(\cdot, u)$ is to allow for *all* transformers. Naturally then, is there a non-trivial sub-class of transformers for which this is possible?
2. **Modeling Power:** Assuming a positive answer to the first question, i.e. assuming the existence of a subset of transformers which enable fast optimization, does restricting to this subset come at a substantial cost in terms of modeling user preferences? Put another way, can this smaller sub-class of transformers achieve the same predictive accuracy as the family of all transformers?

Our Contributions

In short, our contributions provide concrete, theoretically-backed answers to both of the above questions:

1. Modeling User Preferences with Simple Transformers. We focus our study on a sub-class of transformers that we refer to as *simple transformers*. These are transformers which contain a single *self-attention* layer (to be defined in the next section), whereas transformers as a whole may contain multiple attention layers. Addressing the pair of questions above in reverse, we first formally show that simple transformers are able to represent two known, popular parametric models of user preference (Proposition 3):

- *Sequential variety effects* in the context of marketing;
- *Pairwise complementarity and substitution effects* in the context of economics.

It should also be emphasized that none of these models are representable via pure embedding models.

2. Real-Time Personalization with Simple Transformers. Our main result (Theorem 1) is an algorithm (Algorithm 9) which approximately solves Problem (2.1) in sub-linear time, when the objective function is given by a simple transformer:

Theorem 1 (Informal)**.** *Let* OPT *denote the optimal objective value of Problem* (2.1) *when the objective function is given by a simple transformer. Under additional rank assumptions on the simple transformer, for any* $n, k \in \mathbb{N}$ *and* $\epsilon > 0$*, Algorithm 9 returns a solution with objective value at least* $(1-\epsilon)$OPT*, and runs in expected amortized time*

$$\tilde{O}\left(n^{1-c(\epsilon,k)}\, k^{\mu(\epsilon)}\right),$$

where $c(\epsilon, k) > 0$ *and* $\mu(\epsilon) > 0$ *depend only on* ϵ *and* k*. Here,* $\tilde{O}(\cdot)$ *hides factors of order* $n^{o(1)}$*.*

We will present and discuss the rank assumptions in Section 2.2. We also show that these assumptions are necessary for approximately solving Problem (2.1) in sub-linear time (Proposition 4 and Proposition 5), and that our algorithm's expected amortized runtime dependence on k and ϵ is optimal (Proposition 5). In particular, we give the following lower bounds:

Proposition 1 (Informal)**.** *Let* OPT *denote the optimal objective value of Problem* (2.1) *when the objective function is given by a simple transformer.*

(a) *Without the rank assumptions of Theorem 1, there is no exact algorithm for Problem* (2.1) *running in time* $n^{o(k)}$*, assuming the Exponential Time Hypothesis.* [1]

(b) *Even under the rank assumptions of Theorem 1, Problem* (2.1) *admits no* $(1-\epsilon)$*-approximation algorithm with runtime* $f(1/\epsilon)\, k^{o(1/\epsilon)}$ *for any function* f*, assuming the Exponential Time Hypothesis.*

3. Empirical Study. We empirically validated the theoretical results of the previous contributions on two large datasets from *Spotify* [47] (which includes $1,000,000$ playlists with $2,262,292$ unique songs) and the travel website *Trivago* [101] (which includes user sessions of searching for hotel bookings, with around $730,000$ unique

[1] The Exponential Time Hypothesis (ETH) asserts that the 3SAT problem cannot be solved in sub-exponential time. For a Boolean formula in conjunctive normal form with exactly three literals per clause, the 3SAT problem requires deciding if there exists a truth assignment to the variables that satisfies all clauses, and finding one if so.

users and around $340,000$ unique hotels recorded in around $900,000$ different sessions).

In support of the first contribution, our first set of experiments demonstrates that, given data on past user behavior, simple transformers can effectively model and predict user preferences. They achieve (a) substantially higher accuracy than non-attention-based models (such as pure embedding models), and (b) performance that is nearly comparable to more complex transformer architectures. Specifically, simple transformers achieved, on average, 14.1% higher accuracy than non-attention models (e.g., logistic regression, random forest, support vector machine), and only 2.5% lower accuracy than general transformers with multiple self-attention layers.

In support of the second contribution, our second set of experiments demonstrates that our algorithm performs simple-transformer-based personalized recommendation tasks both efficiently and effectively. We solved instances of Problem (2.1) using the simple transformers trained in the first set of experiments and compared our algorithm to two widely used benchmark methods: k-Nearest Neighbor (for retrieval) and Beam Search (for ranking). Under a fixed candidate solution budget (to partially standardize runtime), our algorithm achieved objective values that were, on average, 20.86% higher than those obtained using k-Nearest Neighbor and 20.56% higher than those obtained using Beam Search. Therefore our algorithm achieved strong empirical performance in both the retrieval and ranking phases.

Literature Review

Transformers in Recommender Systems. Since their introduction by [165], transformer architectures have become central to recommender systems due to their ability to model long-range dependencies in user–item interaction sequences. They have been widely adopted in practice by companies such as Alibaba [50], Amazon [105], Spotify [135], and Wayfair [131]. Prior work has primarily focused on developing specialized transformer architectures, including sequential self-attention models [90, 131, 174], single-layer attention models [31, 44, 50, 171], and deeper or more complex variants [49, 69, 105, 112, 159, 177, 182].

In contrast, our work focuses on the optimization problem that arises once the architecture is fixed. We study single-attention-layer transformers, which are both prominent and empirically successful in practice, and design algorithms for fast, near-optimal recommendation under these models.

Representational Power of Transformers. Transformer architectures are built on self-attention, which provides strong representational power in theory and practice. On the theoretical side, Yun et al. [179] and Wei et al. [173] established universal approximation results, analogous to classical results for feedforward networks [80]. Pérez et al. [144] and Wei et al. [173] further showed that transformers are (approximately) Turing-complete. More recently, Sanford et al. [152] presented both positive and negative results, identifying a sparse averaging task where transformers scale logarithmically with input size, unlike recurrent or feedforward networks which scale polynomially, as well as a triple detection task requiring linear attention scaling. Related limitations for induction heads were shown by Bietti et al. [38], Elhage et al. [66], Sanford et al. [151].

In the context of recommender systems and choice modeling, Ko and Li [102] showed that classical models such as Halo-MNL can be represented by a single attention layer, and Wang et al. [170] proposed transformer architectures for learning a broad class of choice models. We complement this line of work by showing that a single attention layer can also capture user preference models with sequential variety effects and pairwise complementarity or substitution.

Approximate Nearest Neighbor. Approximate nearest neighbor (ANN) methods are central to real-time personalization, enabling efficient retrieval of relevant items from embeddings when exact nearest neighbor search is computationally prohibitive at scale, especially in high dimensions. Common approaches include hashing-based methods such as Locality-Sensitive Hashing (LSH) [7, 8, 86], graph-based traversal of proximity graphs [127], and tree-based methods such as KD-trees and Ball Trees, which are effective in low dimensions but degrade as dimensionality increases [136]. In our work, ANN enables fast item retrieval, allowing real-time execution.

Binary Quadratic Optimization. Under transformer architectures, recommendation can be formulated as a binary quadratic optimization (quadratic knapsack) problem, which is NP-hard, subsumes maximum clique and densest k-subgraph, and is NP-hard to approximate within any finite factor [147]. As a result, prior work focuses on tractable cases, including FPTAS results for series-parallel and bounded-treewidth graphs [147], and a PTAS for planar graphs [161].

In our setting, we exploit the low non-negative rank of the softmax matrix to obtain a PTAS. Related low-rank results include FPTAS algorithms for quasi-concave objectives [73], low-rank objectives over polytopes [134], and PTAS results for

binary nonlinear programs [140]. Our approach instead relies on low *non-negative* rank [53], and is also related to multi-objective knapsack problems [28, 29, 66].

2.2 Model

Real-Time Personalization without Transformers

Pure Embedding Models. Before introducing the transformer-based models that will be the focus of this paper, we begin with a brief overview of *pure embedding models*, which are widely used in modern personalization systems. These models represent users and items as vectors in a shared low-dimensional space, enabling efficient modeling of interactions via simple operations such as the inner product.

Formally, let $[n] \equiv 1, \ldots, n$ denote a set of items, and let $V \in \mathbb{R}^{n \times d}$ be a matrix whose i-th row $v_i^\top \in \mathbb{R}^d$ is the *value vector* of item i. These value vectors are designed so that items with similar embeddings are likely to be perceived similarly by users. Each user is likewise represented by a *user vector* $u \in \mathbb{R}^d$. Both the item and user vectors are typically learned from historical interaction data through matrix factorization, collaborative filtering, or more complex models trained on click or engagement feedback.

The utility (or reward) of recommending item i to user u is modeled as $f_i(v_i^\top u)$, where $f_i : \mathbb{R} \to \mathbb{R}$ is a non-decreasing reward function specific to item i. In many applications, the same function f is used for all items. However, in some cases it is important to allow heterogeneous reward functions that reflect item- or category-specific behavior. For example, different product categories often exhibit different click-through-rate (CTR) saturation patterns: a small increase in relevance (e.g., $v_i^\top u$) might sharply increase CTR for breaking news articles, whereas product ads might exhibit more gradual, linear gains, and fashion items may plateau early due to browsing behavior. Modeling such differences requires different shapes of the function f_i, even if all are monotonic. To maintain full generality, we therefore allow each item to have its own reward function f_i. Moreover, in many applications the image of f_i is often $[0, 1]$, in which case $f_i(v_i^\top u)$ can be interpreted as the probability of a positive user action, such as a click or purchase. In these cases, f_i is sometimes chosen to resemble functions like the logistic function.

One of the key advantages of pure embedding models is the efficiency of real-time personalization. Given a user vector u and a budget k, the goal is to select a subset

$S \subset [n]$ of at most k items that maximizes the total reward:

$$\text{(Pure Embedding)} \qquad \max \quad \sum_{i \in S} f_i(v_i^\top u)$$
$$\text{s.t.} \quad S \subset [n],\ |S| \leq k.$$

Since the objective function is additive and the items are treated independently, this problem can be solved exactly via *nearest neighbor*. In our setting, this corresponds to identifying the k items whose value vectors are most aligned with the user vector u under inner product similarity. That is, for a query $u \in \mathbb{R}^d$, the goal is to find the top-k items maximizing $v_i^\top u$.

Nearest neighbor arises in many applications across recommendation, vision, and language, where one often needs to retrieve items similar to a given input based on some feature representation. A naive solution evaluates all n inner products $v_i^\top u$, which takes $O(nd)$ time. In our case, we compute $f_i(v_i^\top u)$ for every $i \in [n]$, and then select the top k items with the largest values. The optimal solution S^* can be described as follows: if there are at most k items with positive reward (i.e., $f_i(v_i^\top u) > 0$), then S^* includes all of them. Otherwise, S^* consists of the k items with the largest values of $f_i(v_i^\top u)$. This greedy procedure yields an exact solution in linear time with respect to the total number of items.

Algorithms: Approximate Nearest Neighbor. The greedy implementation of nearest neighbor becomes computationally prohibitive as the number of points n grows. In most applications such as personalization tasks, the number of items n is on the scale of millions, and an algorithm with runtime linear in n cannot be performed in real-time. To address this problem, the notion of *approximate nearest neighbor* (ANN) has been widely adopted. Rather than finding the exact nearest point, ANN aims to return a point whose distance to the query is within a factor of the true minimum. Formally, for an additive approximation factor $\epsilon > 0$, the objective of ϵ-ANN is to find any point v_{i^*}, where $i^* \in [n]$, that approximately maximizes the inner product similarity to the query u:

$$v_{i^*}^\top u \geq \arg\max_{i \in [n]} v_i^\top u - \epsilon.$$

This relaxation enables much more efficient data structures and algorithms, often sub-linear in n, which can be implemented in real-time. [2] Various techniques have

[2] There are also other versions of ϵ-ANN that concerns multiplicative approximation factors instead of additive ones. These two notions are equivalent under some boundedness assumptions on the points.

been applied to ϵ-ANN algorithms. We give a detailed overview in Appendix A.1.

Because companies aim to recommend a set of at most k items to a user in their personalization task, their objective is not merely to identify a single item that is attractive to the user, but rather to efficiently retrieve a set of k items that are collectively among the most attractive to the user. Therefore, they consider the notion of *ϵ-Approximate k-Nearest Neighbor*, which returns a set of k items whose inner product similarities are within an additive error of ϵ compared to the top-k true nearest items. This is formally defined below.

Definition 1 (ϵ-Approximate k-Nearest Neighbor Algorithm)**.** An ϵ-Approximate k-Nearest Neighbor algorithm builds a data structure on any given set of points $\{v_1, \ldots, v_n\} \subset \mathbb{R}^d$, and takes any query $u \in \mathbb{R}^d$, any $1 \leq k \leq n$, and any $\epsilon > 0$ as inputs. Let $\pi : [n] \to [n]$ be a permutation of the indices such that $v_{\pi(1)}^\top u \geq \cdots \geq v_{\pi(n)}^\top u$. Let $(i_1^*, \ldots, i_k^*) = (\pi(1), \ldots, \pi(k))$. The oracle outputs k indices $i_1, \ldots, i_k \in [n]$ such that $v_{i_j}^\top u \geq v_{i_j^*}^\top u - \epsilon$ for each $j = 1, \ldots, k$ with expected amortized runtime $k\text{-ANN}(n, d, k, \epsilon)$.

Many ϵ-ANN algorithms can be naturally modified to ϵ-Approximate k-Nearest Neighbor algorithms, with a similar expected amortized runtime, that is, sub-linear in n. As an example, we give an ϵ-Approximate k-Nearest Neighbor algorithm in Appendix A.1.

Lemma 1. *There exists an ϵ-Approximate k-Nearest Neighbor algorithm, which we give in Appendix A.1, with expected amortized runtime*

$$k\text{-ANN}(n, d, k, \epsilon) = O\left(k(d \log(d) + \epsilon^{-3} \log^2(n))\right).$$

Finally, companies apply any given ϵ-Approximate k-Nearest Neighbor algorithm to approximately solve Problem (Pure Embedding). This can be done by first partitioning the items according to their reward functions f_i. For each partition, they apply an ϵ-Approximate k-Nearest Neighbor algorithm to identify k items that are collectively among the most attractive to the user within that partition. They then evaluate the rewards of all such candidate items across partitions and select the top-k items with the highest overall reward.

To analyze the approximation error incurred by this procedure, we introduce the following parametrization of the reward functions. For all $i \in [n]$ and $\epsilon > 0$, the function f_i satisfies:

$$f_i(x - \epsilon) \geq (1 - g(\epsilon)) f_i(x) - h(\epsilon) \quad \text{for all } x,$$

where $0 \leq g(\epsilon) \leq 1$ and $h(\epsilon) \geq 0$ are non-negative functions of ϵ. This parametrization captures the idea that a small additive error in the input x leads to a controlled multiplicative and additive error in the output $f_i(x)$.

This parametrization captures the behavior of many reward functions commonly used in practice. In particular, we make the following observation:

Observation 1. *If $f_i(\cdot)$ is non-decreasing and L-Lipschitz, then it satisfies the above condition with $g(\epsilon) = 0$ and $h(\epsilon) = L\epsilon$. In particular, many standard reward functions (also referred to as activation functions in machine learning), such as logistic, ReLU, leaky ReLU, PReLU, tanh, and softplus, satisfy this condition with $g(\epsilon) = 0$ and $h(\epsilon) = O(\epsilon)$.*

There are other activation functions that are not Lipschitz, yet still satisfies the condition with $g(\epsilon) = O(\epsilon)$ and $h(\epsilon) = O(\epsilon)$. Examples include Fractional Power ReLU, Log-ReLU, and Soft Root Function.

With this parametrization, we can now state our main result on applying ANN algorithms to real-time personalization with pure embedding models. We begin by introducing notations that will be used throughout the paper. For an optimization problem P with objective function f_{P}, let x^*_{P} denote an optimal solution and define $\mathrm{OPT}_{\mathsf{P}} = f_{\mathsf{P}}(x^*_{\mathsf{P}})$ as its optimal value. For an algorithm ALG applied to P, let $x^{\mathrm{ALG}}_{\mathsf{P}}$ be the solution returned by ALG, and set $\mathrm{ALG}_{\mathsf{P}} = f_{\mathsf{P}}(x^{\mathrm{ALG}}_{\mathsf{P}})$ to be the corresponding objective value.

Proposition 2. *Suppose we have an ϵ-Approximate k-Nearest Neighbor algorithm with expected amortized runtime k-ANN(n, d, k, ϵ), and suppose k-ANN(n, d, k, ϵ) is concave in n. Let τ be the number of distinct functions among $f_1, \ldots, f_n$. Given any $\epsilon > 0$, we give an algorithm* ALG *for solving Problem* (Pure Embedding) *that satisfies*

$$\mathrm{ALG}_{\text{Pure Embedding}} \geq (1 - g(\epsilon))\mathrm{OPT}_{\text{Pure Embedding}} - kh(\epsilon).$$

with expected amortized runtime

$$\tau \cdot k\text{-}ANN\left(\frac{n}{\tau}, d, k, \epsilon\right) + \log_2(\tau k).$$

The proof of Proposition 2 appears in Appendix A.1. Notice that many ϵ-Approximate k-Nearest Neighbor algorithms have k-ANN (n, d, k, ϵ) sub-linear and concave in n, and we have given an example in Lemma 1. Therefore Problem (Pure Embedding)

can be approximately solved in sub-linear time, which is practical to implemented in real-time.

Modeling Limitations. Proposition 2 shows that pure embedding models can be optimized efficiently. However, pure embedding models have a fundamental limitation: they treat each item independently and fail to capture how the value of an item may depend on the context in which it is presented. That is, the reward associated with item i is fixed once u and v_i are known, regardless of what other items are presented alongside it. In many personalization applications, this is an unrealistic assumption. For example, the value of a song recommendation may drop if a similar song is already in the playlist; or the likelihood a user clicks on a hotel result may depend on what other hotels appear in the same search result page and how they compare. These effects, often referred to as *set effects*, are not captured by pure embedding models. Concretely, we present three common parametric models used in personalization that cannot be represented by pure embedding models.

First, we consider a famous parametric model of *sequential variety effects* in sequences. The concept of variety/diversity has been examined extensively in the marketing literature (see e.g. [79, 129, 148]). This model proposes that the perceived utility of an item depends not only on its intrinsic quality but also on its novelty relative to previously seen items. In particular, repeated exposure to similar items leads to diminishing marginal utility, while introducing diverse or contrasting items can restore or amplify engagement. This intuition aligns closely with the notion of *discounted utility* over sequences, where the utility derived from an item is multiplicatively reduced based on its similarity to past items (see, e.g. [25, 26]). Such models capture behavioral tendencies like satiation, boredom, and the desire for exploration, and have been influential in both economic theory and practical recommendation systems. Below we give a mathematically formulation of sequential variety effects.

Model 1 (Sequential Variety Effects)**.** Let $[n] := \{1, \ldots, n\}$ denote the set of items. Each item i has a *base utility* $\hat{u}_i > 0$ and a *similarity embedding* $x_i \in \mathbb{S}^{d-1}$. Define *pairwise similarity score* between item i and item j by $s_{ij} := x_i^\top x_j \in [-1, 1]$. Fix a sequence length $k \geq 1$ and nonnegative *lag weights* $\lambda_1, \ldots, \lambda_{k-1} \geq 0$.

For a sequence $S = (i_1, \ldots, i_k)$ of length k, the *sequential–variety–adjusted utility*

of the item at position t is

$$g(S, i_t) = \hat{u}_{i_t} \exp\left(\beta \sum_{\ell=1}^{t-1} \lambda_\ell \, s_{i_t, i_{t-\ell}}\right), \qquad t = 1, \ldots, k,$$

where $\beta \in \mathbb{R}$ controls the strength and sign of the variety effect. [3]

Model 1 adjusts the context-free utility $\hat{u}_{i_t}$ multiplicatively according to the similarity between the current item and recently shown items, with older influences discounted by λ_ℓ. When $\beta < 0$ (preference for variety), similarity to recent items ($s_{ij} > 0$) decreases the current utility, while dissimilarity ($s_{ij} < 0$) increases it; when $\beta > 0$ (preference for continuity), the effect reverses and thematic similarity boosts utility. The weights λ_ℓ specify the memory profile: choosing $\lambda_\ell = \rho^\ell$ with $\rho \in (0, 1)$ yields exponential decay in influence with lag, whereas setting $\lambda_1 > 0$ and $\lambda_\ell = 0$ for $\ell > 1$ recovers a one-step effect based only on the immediate predecessor. The exponential factor ensures $g(S, i_t) > 0$ and provides a smooth, multiplicative adjustment driven by recent sequence context.

Model 1 also connects closely to the notion of *Maximal Marginal Relevance (MMR)* introduced by Carbonell and Goldstein [42], a classical approach in information retrieval that balances relevance and diversity. MMR selects an item by trading off its intrinsic relevance score against its similarity to previously selected items. Our model can be viewed as a smooth, parametric generalization of this idea. When $\beta < 0$ and the lag weights concentrate on the most recent item (e.g., $\lambda_1 > 0$, $\lambda_\ell = 0$ for $\ell > 1$), the adjustment term penalizes items that are highly similar to those already chosen, mirroring the diversity-inducing term in MMR. More generally, Model 1 replaces the linear trade-off in MMR with a multiplicative, exponentially weighted discount that naturally extends to sequential settings with memory and graded similarity effects. Thus, the model provides a behavioral micro-foundation for diversity-aware ranking rules widely used in marketing and recommendation systems.

Second, we consider a model of *pairwise complementarity and substitution effects*. In economics, these effects describe how the presence of one item influences the desirability of another: complementary items enhance each other's attractiveness, while substitutes reduce it. A common modeling approach represents items as vectors in a feature space, with pairwise interactions captured via inner products (see, e.g., [32, 109, 150]).

[3] By convention, the empty sum equals 0, so $g(S, i_1) = \hat{u}_{i_1}$.

Model 2 (Pairwise Complementarity and Substitution Effects)**.** Let $[n]$ denote the set of items. Each item i has a value vector $\hat{v}_i \in \mathbb{R}^d$. The pairwise complementarity and substitution effects is parametrized by a matrix $H \in \mathbb{R}^{n\times n}$ where $H_{ii} = 0$ for every $i \in [n]$. For a user $\hat{u} \in \mathbb{R}^d$ and a subset of items $S \subset [n]$, the *interaction-adjusted utility* of item $i \in S$ is defined as

$$g(S, i) = \hat{v}_i^\top \hat{u} + \sum_{j\in S} H_{ij}.$$

In Model 2, the value vector $\hat{v}_i$ captures the intrinsic preference of the user u for item i, independent of any other items shown. The matrix H encodes all pairwise interaction effects between items: a positive entry H_{ij} means that the presence of item j increases the utility of item i (complementarity), while a negative entry H_{ij} means that the presence of j reduces the utility of i (substitution). The diagonal entries are zero by definition, so an item does not directly influence its own utility. When $H_{ij} = 0$, item j has no effect on the utility of item i.

Model 2 is closely related to *choice models* that generate *conversion probabilities*. We discuss this connection in Appendix A.3.

Because in both models the reward associated with an item can depend on the presence or absence of other recommended items, we make the (informal) observation that these models cannot be represented by pure embedding models. In contrast, we will later show that both models can be naturally expressed within the framework we study in this paper.

Simple Transformers

In this section, we formally state the model which will be our primary object of study: *simple transformers*, or neural networks with a single *self-attention* layer. First, some preliminary definitions: the row-wise *softmax* operator, which we denote as $\text{softmax} : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$, is given by

$$\text{softmax}(A)_{i,j} = \frac{\exp(A_{i,j})}{\sum_{j'=1}^{d} \exp(A_{i,j'})}.$$

In particular, each row of $\text{softmax}(A)$ sums up to 1, and can be interpreted as a vector of weights. The notion of attention is fundamental to transformer-based models (e.g. [152, 165]). For input dimension n, output dimension d_v, embedding dimension d_{kq}, and matrices $Q, K \in \mathbb{R}^{n\times d_{kq}}$, and $V \in \mathbb{R}^{n\times d_v}$, a *self-attention layer* is a function $\text{SA}_{Q,K,V} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times d_v}$ given by

$$\text{SA}_{Q,K,V}(X) = \text{softmax}((XQ)(XK)^\top)XV.$$

Here, the matrices Q, K, and V are often called the *query*, *key*, and *value* matrix, respectively.

To better understand the self-attention layer, it is natural to view it as a function on subsets of items. Let $[n]$ denote a set of items. Then the self-attention layer is fully parameterized by an individual query, key, and value vector for each item (forming the rows of the respective matrices Q, K, and V). For any subset $S \subset [n]$, we define the *set-membership matrix* $X_S \in \{0,1\}^{n \times n}$ by

$$(X_S)_{ii} = \begin{cases} 1, & i \in S, \\ 0, & i \notin S. \end{cases}$$

The function $\mathrm{SA}_{Q,K,V}(\cdot)$ is then applied to X_S. The output is an $n \times d_v$ matrix in which each row $i \in [n]$ is interpreted as follows:

- If $i \in S$, then the i-th row is a weighted average of the value vectors $\{v_i : i \in S\}$, where the weight on each v_i is given by the softmax-normalized dot product between the query vector q_i and the key vector k_j for $j \in S$.
- If $i \notin S$, then the i-th row is the uniform average of the value vectors $\{v_i : i \in S\}$. This is because item i's query vector is zeroed by X_S, that is, $(X_S Q)_i$ is the zero vector, and thus assigns equal softmax weight to all items in S.

Transformers are a broad family of functions (also known as *neural networks*) constructed by repeatedly applying self-attention layers along with simple transformations that operate independently on each item. These point-wise transformations typically consist of linear mappings followed by non-linear functions known as *activation functions*, which introduce flexibility and allow the model to capture complex behaviors. In our context, these activation functions can be naturally interpreted as the reward functions f_i, representing how the utility of each item responds to its input.

Simple transformers are a subclass of transformers with a single self-attention layer, followed by point-wise activation functions f_i. Formally, we define them as follows:

Definition 2 (Simple Transformer)**.** For matrices $Q, K \in \mathbb{R}^{n \times d_{kq}}$ and $V \in \mathbb{R}^{n \times d_v}$, a vector $u \in \mathbb{R}^{d_v}$, and non-decreasing *activation functions* $f_1, \ldots, f_n : \mathbb{R} \to \mathbb{R}$, a

simple transformer is a function $\mathcal{T}_{Q,K,V,f_1,\ldots,f_n,u} : \mathbb{R}^{n\times n} \to \mathbb{R}^n$ given by

$$\mathcal{T}_{Q,K,V,f_1,\ldots,f_n,u}(X) = \begin{bmatrix} f_1\big(\mathrm{SA}_{Q,K,V}(X)_1^\top u\big) \\ f_2\big(\mathrm{SA}_{Q,K,V}(X)_2^\top u\big) \\ \vdots \\ f_n\big(\mathrm{SA}_{Q,K,V}(X)_n^\top u\big) \end{bmatrix}.$$

As a side note, a simple transformer can also include multiple *attention heads*, in which case several self-attention layers are computed in parallel, each called an attention head, using different learned Q, K, and V matrices. The outputs of all attention heads are then concatenated and passed through point-wise transformations. Equivalently, a simple transformer with multiple attention heads can be written as one with a single head by arranging each Q, K, and V in block form. All of our results extend naturally to this setting. For clarity of notation, however, throughout this paper we focus on simple transformers with a single attention head, as defined above.

Modeling Power

Simple transformers are already widely used in personalization [31, 44, 50, 171], though more sophisticated multi-layer transformers are also common. This raises a key question: to what extent can simple transformers effectively model user preferences?

To address this, we build intuition for how simple transformers operate in personalization. A simple transformer can be viewed as modeling interactions between a user and a recommended set of items, while also capturing pairwise interactions among items, often called "set effects." Concretely, let $[n]$ denote a set of n items, and let $S \subset [n]$ be a subset recommended to a user $u \in \mathbb{R}^{d_v}$. The matrix $V \in \mathbb{R}^{n\times d_v}$ contains value vectors, where each row $v_i^\top$ represents item i in isolation. When $S = \{i\}$, the self-attention output is simply v_i, and the reward is $f_i(v_i^\top u)$.

When $|S| > 1$, the self-attention layer transforms each v_i into a convex combination of $\{v_j : j \in S\}$, with weights determined by the softmax of inner products between the query vector q_i and key vectors $\{k_j : j \in S\}$, i.e.,

$$\mathrm{softmax}\big((X_S Q)(X_S K)^\top\big)_i.$$

Intuitively, q_i captures how other items in S influence item i, while k_j captures how item j influences others. The resulting vector $\mathrm{SA}_{Q,K,V}(X_S)_i$ is projected onto u and

passed through f_i, yielding reward

$$f_i\left(\mathrm{SA}_{Q,K,V}(X_S)_i^\top u\right),$$

which corresponds to the i-th coordinate of $\mathcal{T}_{Q,K,V,f_1,\ldots,f_n,u}(X_S)$.

This formulation is permutation invariant, reflecting the unordered nature of S. Sequence models can be incorporated by augmenting Q, K, and V with positional encodings, which tag items with their positions and allow the model to distinguish otherwise identical items appearing in different positions. Thus, sequential structure is captured within the same framework as set-based interactions.

Having discussed how simple transformers operate in the personalization setting, we now examine their ability to model user preferences. While our later experiments will empirically demonstrate that restricting the architecture to a single attention layer results in only a modest reduction in modeling/predictive power, we begin by supporting this claim through an analysis of the two common parametric models used in personalization presented in the previous section. We have already seen that these two models cannot be represented by pure embedding models. On the contrary, below we show that both of them can be represented by simple transformers.

Proposition 3. *The sequential variety effects in Model 1 and the complementarity and substitution effects in Model 2 can both be represented by a simple transformer.*

The proof of Proposition 3 appears in Appendix A.2.

Finally, we provide a graph interpretation of self-attention layers in Appendix A.4.

Optimization

The primary purpose of this paper is to study the problem of personalizing a set (or sequence, equivalently) of items in real-time, where the underlying model of user preferences is given by a simple transformer. Following the setup and terminology in the previous subsections, let $[n]$ index a set of items, for which the query, key, and value matrices $Q, K \in \mathbb{R}^{n\times d_{kq}}$ and $V \in \mathbb{R}^{n\times d_v}$ are fixed in advance (these should be thought of as having been learned from prior data). Each user is represented by a vector $u \in \mathbb{R}^{d_v}$ in the same space as the value vectors. For a given set $S \subset [n]$, the simple transformer's output that corresponds to item $i \in S$ is given by $\mathcal{T}_{Q,K,V,f_1,\ldots,f_n,u}(X_S)_i = f_i(\mathrm{SA}_{Q,K,V}(X_S)_i^\top u)$. Intuitively, this can be thought of as the reward obtained from recommending item i.

As a user arrives, our goal is to choose a set $S \subset [n]$ of at most k items that maximizes the total reward. Formally, the optimization problem we study is:

Definition 3 (Simple-Transformer Based Recommendations)**.**

$$\text{(Main)} \qquad \max \quad \sum_{i \in S} f_i(\mathrm{SA}_{Q,K,V}(X_S)_i^\top u)$$
$$\text{s.t.} \quad S \subset [n],\ |S| \leq k.$$

Let OPT denote the optimal objective value of Problem (Main). Notice that if $S = \emptyset$, that is, if nothing is recommended to the user, then the objective value of of Problem (Main) equals to 0. Therefore OPT ≥ 0. To ensure that Problem (Main) is meaningful, from now on we assume that OPT > 0, since otherwise the best decision would be recommending nothing.

Hardness

As a starting point toward solving Problem (Main), observe that it can be solved exactly in $O(n^k k^2)$ time via brute-force evaluation of all feasible solutions. As mentioned earlier, we are motivated by settings in which the number of items n is extremely large (possibly hundreds of millions), and Problem (Main) must be solved in real time (possibly milliseconds). Thus, our goal will be to find an algorithm, potentially approximate rather than exact, whose runtime is *sub-linear* in n, i.e., $O(n^\gamma)$ for some $\gamma < 1$. Moreover, while the budget on the number of items to recommend, k, is typically moderate in practice (often around ten), exponential dependence on k would still be impractical to be implemented in real time. Therefore, our algorithm's runtime should also be polynomial in k.

Before proceeding, it is useful to present some initial hardness results to temper our expectations. We provide two sets of results. The first addresses the requirement of sub-linear runtime in n, with hardness parametrized by d_{kq}, the dimension of the key and query vectors. The second addresses the requirement of polynomial runtime in k, with hardness parametrized by the *non-negative rank* of the matrix $W := \mathrm{softmax}(QK^\top)$, denoted as $\mathrm{rank}_+(W)$.

Hardness Parametrized by d_{kq}. We first present a proposition that reduces Problem (Main) to graph problems involving cliques, and then discuss its implications.

Proposition 4.

(a) *If* $d_{kq} = n$ *and* $k \geq 4$*, then Problem* (Main) *subsumes the* $(k-1)$*-*Clique *problem* [4] *on graphs with* $n-1$ *vertices.*

(b) *For any constant* $M \geq 3$*, there exists a number* $c(M) > 0$ *for which the following holds. For any* d_{kq} *such that* $\exp(c(M) \cdot d_{kq}) \leq n-1$ *and any* $k \geq M+1$*, Problem* (Main) *subsumes the problem of finding a largest clique in a graph with* $n-1$ *vertices,* $\exp(c(M) \cdot d_{kq})$ *disjoint cliques, and all cliques have size at least* $k-M$ *and at most* $k-1$*.*

The proof of Proposition 4 appears in Appendix A.5. Proposition 4 implies concrete limitations on the theoretical results we can expect for solving Problem (Main):

- By Proposition 4 (a), Problem (Main) inherits the hardness of the k-Clique problem, which is known to be NP-hard (when k is allowed to grow with n) [93]. Thus, we should not expect to find an exact algorithm which runs in $O(n^C)$ for some $C > 0$ independent of k.

- Moreover, it is known [48] that no exact algorithm can run in time $n^{o(k)}$ assuming *Exponential Time Hypothesis* holds. Thus, absent additional assumptions, we can only expect to *approximately* solve Problem (Main) in sub-linear time with respect to n.

- One natural assumption to make is that d_{kq} is small (this is typically the case in practice), and indeed our main result will be parameterized by d_{kq} and only non-trivial when $d_{kq} = o(\log n)$. By Proposition 4 (b), if $d_{kq} = \Omega(\log n)$, Problem (Main) is at least as hard as finding the largest clique in a graph with n vertices and $\Omega(n)$ disjoint cliques. In particular, each clique in such a graph is itself a candidate maximum clique, and any algorithm must effectively search over $\Omega(n)$ disjoint candidates to determine the largest, since there is no structural overlap between the cliques that an algorithm could exploit to narrow the search space. This indicates that an exact algorithm in sub-linear time with respect to n cannot exist when $d_{kq} = \Omega(\log n)$.

Hardness Parametrized by $\mathbf{rank}_+(W)$. Following on the above discussion, we require some additional assumptions to ensure that Problem (Main) can be solved, even approximately, in sub-linear time with respect to n. One such "assumption"

[4] For a (undirected, unweighted) graph, the k-Clique problem requires deciding if a clique of size k exists, and finding one if so.

will be that d_{kq} is small. We do not state this as a formal assumption, but rather our main result will be parameterized by d_{kq}.

Similarly, we also require some additional assumptions to ensure that Problem (Main) can be solved, even approximately, in polynomial time with respect to k. Our main result will be parameterized by a rank-type quantity pertaining to the matrix $W = \text{softmax}(QK^T)$. Now $QK^\top$ is by definition of rank d_{kq}, and while the softmax operator does not preserve rank exactly, it is known that W can be well-approximated by a matrix with rank polynomial in d_{kq} (see [3, 76]). Thus, for example, if d_{kq} is constant, then W is approximately low-rank.

Due to the softmax operator, the matrix W is entry-wise non-negative. This allows us to consider a structural parameter known as the *non-negative rank* of W, denoted $\text{rank}_+(W)$. The non-negative rank of a matrix $W \in \mathbb{R}^{n\times n}_{\geq 0}$ is defined as the smallest integer $\text{rank}_+(W)$ such that W can be written as a product of two non-negative matrices:

$$W = AB^\top, \text{ where } A, B \in \mathbb{R}^{n\times \text{rank}_+(W)}_{\geq 0}.$$

Equivalently, W can be expressed as the sum of $\text{rank}_+(W)$ non-negative rank-one matrices. Such a representation is called a *non-negative factorization* of W. Clearly, this notion is stronger than standard matrix rank; in particular, we always have $\text{rank}(W) \leq \text{rank}_+(W) \leq n$. For more properties on non-negative rank, see [53]. Non-negative rank has many applications in various fields, including data mining, combinatorial optimization, quantum mechanics, and more. In our case, it turns out that the non-negative rank of W is a key parameter that quantifies the hardness of Problem (Main). Formally, we present the following proposition:

Proposition 5. *If* $\text{rank}_+(W) = 2$*, Problem* (Main) *admits no* $(1-\epsilon)$*-approximation scheme*

(a) *with runtime* $f(1/\epsilon)\, k^{O(1)}$ *for any function* f*, assuming the* k*-*Clique *problem is not* Fixed-Parameter Tractable [5] *(Corollary of Theorem 6 in [103]), and*

(b) *with runtime* $f(1/\epsilon)\, k^{o(1/\epsilon)}$ *for any function* f*, assuming Exponential Time Hypothesis holds (Corollary of Theorem 5.1 in [87]).*

[5] A problem is said to be Fixed-Parameter Tractable (FPT) if it can be solved in time $f(k)n^{O(1)}$ for some function f, where k is a chosen problem parameter. A widely held conjecture is that the k-Clique problem is not FPT.

For general $\text{rank}_+(W)$*, Problem* (Main) *admits no* $(1-\epsilon)$*-approximation scheme with runtime*

$$k^{o\left(\frac{\text{rank}_+(W)}{\epsilon \log^2(\text{rank}_+(W)/\epsilon)}\right)} \text{ or } k^{o(\sqrt{\text{rank}_+(W)})},$$

assuming Gap Exponential Time Hypothesis *holds* [6] *(Corollary of [60]).*

The proof of Proposition 5 appears in Appendix A.5. Our proof is based on a reduction from Problem (Main) to the well-known *Multi-dimensional Knapsack Problem* (MDKP), formally defined in Appendix A.5. Proposition 5 shows that, when viewing k as the parameter in Problem (Main), any algorithm achieving a $(1-\epsilon)$-approximation must incur runtime with exponential dependence on k that necessarily involves both $\text{rank}_+(W)$ and $1/\epsilon$ in a non-trivial way.

2.3 Main Result

From the discussions in the previous section, recall that while our algorithm's runtime is parametrized by the non-negative rank of W, achieving sub-linear dependence on n and polynomial dependence on k requires that the non-negative rank of W be small.[7] Importantly, our main result does not require W itself to have low non-negative rank, but only that W can be well approximated entry-wise by a low non-negative rank matrix.

Formally, suppose there exists a non-negative matrix $W' \in \mathbb{R}^{n\times n}_{\geq 0}$ such that

$$1-\gamma \leq \frac{W_{ij}}{W'_{ij}} \leq 1+\gamma \quad \text{for all } i,j,$$

with $\text{rank}_+(W') = r_+$.[8] Then our main result is parametrized by r_+ and γ. Just as in the case of the parameter d_{kq}, our guarantee is non-trivial when $r_+ = O(1)$.

Before stating our main result, we introduce some notation. For a vector x, let $x_{\max}$ and $x_{\min}$ denote the largest and smallest entry of x, respectively. Likewise, for a matrix X, let $X_{\max}$ and $X_{\min}$ denote its largest and smallest entry, respectively. If X

[6] The Gap Exponential Time Hypothesis (Gap-ETH) asserts that, for some constant $\epsilon > 0$, distinguishing between satisfiable 3SAT formulas and those that are not even $(1-\epsilon)$-satisfiable requires exponential time.

[7] Computing a non-negative matrix factorization is in general NP-hard [166]: there is a rich literature on this subject that has yielded multiple algorithms (see [108, 172] for surveys). We will not be concerned with this runtime in analyzing Problem (Main), as Q and K are given beforehand, and thus we view this as amortized across multiple instances of Problem (Main).

[8] Actually, we only require a particular sub-matrix to be of non-negative rank r_+. This sub-matrix is of significantly smaller size, corresponding to the entries which survives a certain pruning procedure.

has rows $x_i^\top$, we define

$$\|X\|_{2,\infty} \;=\; \max_i \|x_i\|_2,$$

that is, the matrix norm induced by the vector 2- and ∞-norms.

We are now prepared to state our main result, which is that our algorithm (to be described in the next section) achieves the following:

Theorem 1. *Let k-ANN(n, d, k, ϵ) be the expected amortized runtime of an ϵ-Approximate k-Nearest Neighbor algorithm, which we assume is concave in n. Let τ be the number of distinct functions among $f_1, \ldots, f_n$. Suppose there exists $W' \in \mathbb{R}_+^{n \times n}$ such that $1 - \gamma \le W_{ij}/W'_{ij} \le 1 + \gamma$ for all i, j, and W' has non-negative rank r_+ with an explicit non-negative factorization. Let* ALG *denote the objective value returned by Algorithm 9 for Problem* (Main)*.*

Given any $\epsilon > 0$, Algorithm 9 achieves

$$\text{ALG} \ge (1 - g(\epsilon) - 2g(\gamma) - 2g((1+\gamma)\epsilon)) \cdot \text{OPT} - (\epsilon + h(\epsilon))k.$$

Moreover, its expected amortized runtime is

$$O\left(\epsilon^{-2d_{kq}} \cdot \tau \cdot k\text{-ANN}\left(\frac{n}{\tau}, d_v, k, \epsilon\right) + \epsilon^{-2d_{kq} r_+^2/\epsilon}(1+\gamma)^{r_+}(\tau k)^{r_+^2/\epsilon}\right),$$

where the Big-Oh depends only on $\|Q\|_{2,\infty}$, $\|K\|_{2,\infty}$, $(Vu)_{\max}$, $W_{\min}$, and $\max_{i \in [n]}\{f_i((Vu)_{\max})\}$.

Some remarks will be useful for parsing and interpreting this result:

- The concavity assumption on k-ANN(n, d, k, ϵ) is imposed for analytical convenience. It suffices that the runtime be sub-linear in n, as any sub-linear function admits a concave sub-linear upper bound and hence can be handled within our analysis. Many ϵ-Approximate k-Nearest Neighbor algorithms satisfy these conditions; see Lemma 1 for one such example.

- In practice, the number of distinct functions τ is typically very small, often just one.

- In our algorithm's performance guarantee, the multiplicative dependence on g and the additive dependence on h arise from the parametrization of f_i:

  $$f_i(x - \epsilon) \;\ge\; (1 - g(\epsilon))\, f_i(x) - h(\epsilon) \quad \text{for all } x.$$

  The additional ϵk additive term comes from the use of the ϵ-Approximate k-Nearest Neighbor algorithm.

- By Observation 1, if each f_i is L-Lipschitz, our algorithm achieves

$$\text{ALG} \geq \text{OPT} - C\epsilon Lk$$

for some universal constant $C > 0$.

- If $d_{kq} = o(\log n)$ and $r_+ = O(1)$, the amortized runtime of our algorithm can be simplified to

$$\tilde{O}\left(kn^{1-c\epsilon/k} + k^{C/\epsilon}\right).$$

- Similar to how a small d_{kq} implies low (standard) rank approximation, it also implies that W can be approximated entry-wise by a matrix of low non-negative rank. In particular, we show that if the rows of Q and K can be grouped into ℓ clusters, then such a W' can be constructed with $r_+ = \ell(\ell+1)/2$ and a small γ.

Proposition 6. *Let $W = \text{softmax}(QK^\top)$. Suppose $I_1, \ldots, I_\ell$ form a partition of $[n]$ such that for every $\ell' \in [\ell]$ and $i, i' \in I_{\ell'}$, we have $\|q_i - q_{i'}\|_2 \leq \delta$ and $\|k_i - k_{i'}\|_2 \leq \delta$. Then we can construct $W' \in \mathbb{R}^{n\times n}_{\geq 0}$ such that*

(a) *$1-\gamma \leq W_{ij}/W'_{ij} \leq 1+\gamma$ for all i, j, where $\gamma = 17\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}$,*

(b) *and W' has non-negative rank $r_+ = \ell(\ell+1)/2$ with an explicit non-negative factorization.*

To obtain the clusters, one approach is to directly partition the row spaces of Q and K, and group the rows of Q and K according to this partition. Since d_{kq} is assumed to be small, the number of clusters is also small. This yields the following corollary.

Corollary 1. *Let $W = \text{softmax}(QK^\top)$. Given any $\delta > 0$, we can construct $W' \in \mathbb{R}^{n\times n}$ such that $1 - \gamma \leq W_{ij}/W'_{ij} \leq 1 + \gamma$ for all i, j where $\gamma = 17\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}$, and W' has non-negative rank*

$$r_+ = \lceil 4 \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}/\delta \rceil^{4d_{kq}}$$

with an explicit non-negative factorization.

The proofs of Proposition 6 and Corollary 1 appear in Appendix A.6. Notice that if $d_{kq} = O(1)$, Corollary 1 shows that $r_+ = O(1)$ for some small γ. In this case our main result is non-trivial.

- In practice, the embedding dimension d_{kq} of items is usually much smaller than the number of items n, so $d_{kq} = o(\log n)$ often holds. In addition under the condition that $r_+ = O(1)$, for any fixed $\epsilon > 0$, our algorithm gives a $1 - \epsilon$ approximation algorithm with expected amortized runtime that is sub-linear in the total number of items n, and polynomial in the number of items to recommend k.
- Our algorithm's amortized runtime is practical. In reality companies and online platforms can easily have millions of products, so even algorithms with runtime polynomial in n would not be able to complete the personalized recommendation tasks in real-time. The number of items that companies can recommend to a single user is usually on the scale of tens (such as displaying products on a webpage), so algorithms with runtime exponential in k also would not be able to complete the personalized recommendation tasks in real-time.

Real-time personalization algorithms typically operate in two phases: *retrieval* and *ranking*. In phase one (retrieval), the algorithm efficiently selects a small subset of promising candidate items from a large item pool. This is typically achieved using ANN algorithms over learned item embeddings. In phase two (ranking), an optimization problem, which is determined by the specific personalization model, is approximately solved over the retrieved subset to produce the final recommendations.

Our algorithm also follows the retrieval and ranking structure, but introduces novel methods in both the retrieval and ranking phases. The two pieces in our algorithm's expected amortized runtime directly correspond to the expected amortized runtime of our algorithm's two phases: the expected amortized runtime of phase one is

$$O\left(\epsilon^{-2d_{kq}} \cdot \tau \cdot k\text{-ANN}\left(\frac{n}{\tau}, d_v, k, \epsilon\right)\right),$$

and the runtime of phase two is

$$O\left(\epsilon^{-2d_{kq}r_+^2/\epsilon}(1+\gamma)^{r_+}(\tau k)^{r_+^2/\epsilon}\right).$$

2.4 Algorithm and Proof of Main Theorem

Before diving into the details of our algorithm, we rewrite Problem (Main) with slightly different notations. Let $q_i^\top$ and $k_i^\top$ denote the i-th rows of Q and K, respectively. Let $W = \mathrm{softmax}(QK^\top)$, and let $w_i^\top$ be the i-th row of W. For vectors $a, b \in \mathbb{R}^n$, let $a \odot b \in \mathbb{R}^n$ denote the element-wise product of a and b, i.e., $(a \odot b)_i = a_i b_i$ for every $i \in [n]$. We make the following observation.

Observation 2. *Problem* (Main) *is equivalent to the following Problem* (P)*:*

$$\text{(P)} \qquad \max \quad f_{\mathsf{P}}(x) = \sum_{i=1}^{n} x_i f_i \left(\frac{(w_i \odot Vu)^\top x}{w_i^\top x} \right)$$
$$\text{s.t.} \quad x \in \{0,1\}^n, \quad 1 \le e^\top x \le k$$

where $e \in \mathbb{R}^n$ is the all-ones vector, and $x_i = 1$ indicates that item i is selected into the set S.

The proof of Observation 2 appears in Appendix A.8, and the pseudo-code of our main algorithm appears in Appendix A.7. From this point onward, we will work with P, using its notation in place of Problem (Main).

Phase One (Retrieval)

In phase one, our algorithm aims to identify a small subset $I \subset [n]$ of items such that the optimal objective value of P does not decrease significantly when restricted to I. To achieve this, our algorithm first partitions items offline based on their query vectors q_i, key vectors k_i, and reward functions f_i. Since both q_i and k_i lie in a space of dimension d_{kq}, the number of partitions is much smaller than n. Items in the same partition are designed to behave similarly under the self-attention layer. That is, they produce similar outputs when interacting with other items. When a user u arrives, the algorithm applies an ϵ-Approximate k-Nearest Neighbor algorithm (as defined in Section 2.2) within each partition to select at most k items whose value vectors have the highest approximate inner product similarity with u. Because items within the same partition respond similarly under attention, user preferences within each partition are primarily determined by similarity to the user vector. Thus, our algorithm retrieves a small subset I containing high-reward items tailored to the user.

Proposition 7 (Phase One)**.** *Suppose we have an ϵ-Approximate k-Nearest Neighbor algorithm with expected amortized runtime k-ANN(n, d, k, ϵ). Let τ be the number of distinct functions among $f_1, \ldots, f_n$. Given any $\epsilon > 0$, Algorithm 11 returns an index set $I \subset [n]$ such that*

$$|I| = \tau k \left\lceil \frac{140 \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\})^2}{(Vu)_{\max} \cdot \epsilon} \right\rceil^{2d_{kq}},$$

and the optimal value to the following Problem $\mathsf{P}(I)$

$$\text{(P}(I)\text{)} \qquad \max \quad f_{\mathsf{P}(I)}(x) = \sum_{i=1}^{n} x_i f_i \left(\frac{(w_i \odot Vu)^\top x}{w_i^\top x} \right)$$
$$\text{s.t.} \quad x \in \{0,1\}^n,\ x_i = 0 \text{ for } i \notin I,\ 1 \le e^\top x \le k$$

satisfies

$$\mathrm{OPT}_{\mathsf{P}(I)} \geq (1 - g(\epsilon))\mathrm{OPT}_{\mathsf{P}} - kh(\epsilon).$$

Moreover, suppose k-ANN(n, d, k, ϵ) is concave in n. Then the expected amortized runtime of Algorithm 11 is

$$\left\lceil \frac{140\big(\max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}\big)^2 (Vu)_{\max}}{\epsilon} \right\rceil^{2d_{kq}} \tau \cdot k\text{-ANN}\left(\frac{n}{\tau},\, d_v,\, k,\, \frac{\epsilon}{35\, \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}\, (Vu)_{\max}}\right).$$

Suppose $d_{kq} = o(\log n)$ and $\|Q\|_{2,\infty}, \|K\|_{2,\infty}, (Vu)_{\max}$ are all viewed as constants, then for any given constant $\epsilon > 0$, we have $|I| = \tau k n^{o(1)}$. Moreover, suppose k-ANN (n, d_v, k, ϵ) is sub-linear in n when d_v, k, ϵ are fixed, then the expected amortized runtime of Algorithm 11 is also sub-linear in n.

Phase Two (Ranking)

In phase two, our algorithm approximately solves $\mathsf{P}(I)$, which is P over the retrieved subset of items $I \subset [n]$. Without loss of generality, assume $I = [m]$. By the first remark of Proposition 7, we may treat $m = kn^{o(1)}$ under mild assumptions. Then since $x_i = 0$ for $i > m$, we may consider only the first m entries of Vu and the top-left $m \times m$ principal sub-matrix of W. Therefore, with slight abuse of notation, we redefine $Vu \in \mathbb{R}^m$ to include its first m entries, and $W, W' \in \mathbb{R}_+^{m\times m}$ to be the top-left $m \times m$ principal sub-matrices of the corresponding matrices, respectively. Moreover, note that the quantity

$$\frac{(w_i \odot Vu)^\top x}{w_i^\top x}$$

remains unchanged if the vector w_i is multiplied by a non-zero constant. Thus, rescaling the rows of W does not change $\mathsf{P}(I)$. Because $W \in \mathbb{R}_+^{m\times m}$ is now the $m \times m$ principal sub-matrix, the sum of its rows is not normalized to 1. So for simplicity of exposition, we assume each row of W is rescaled so that $\sum_{j=1}^m W_{ij} = 1$, and each row of W' is rescaled accordingly so that $1 - \gamma \leq W_{ij}/W'_{ij} \leq 1 + \gamma$ for all i, j. Then we may rewrite $\mathsf{P}(I)$ as

$$(\mathsf{P}(I)) \qquad \max \quad f_{\mathsf{P}(I)}(x) = \sum_{i=1}^m x_i f_i\left(\frac{(w_i \odot Vu)^\top x}{w_i^\top x}\right)$$
$$\text{s.t.} \quad x \in \{0,1\}^m,\ 1 \leq e^\top x \leq k.$$

From this point onward, we will work with this new form of $\mathsf{P}(I)$.

Our algorithm begins by replacing W with a low non-negative rank surrogate W' and showing that solving the problem under this approximation is sufficient. Rather than exhaustively enumerating all possible solutions, our algorithm then focuses on a restricted collection of partial solutions that retain the key structural information. The nonlinear terms in the objective are handled by introducing a family of auxiliary linearized problems, which can be further simplified through discretization. This reduction ensures that only a small number of auxiliary linearized problems need to be solved.

To address each auxiliary linearized problem, our algorithm employs a rounding procedure that converts fractional linear-programming solutions into valid discrete ones. At this stage, the central trade-off emerges: exploring too many partial solutions increases runtime beyond practical limits, while exploring too few places excessive burden on the rounding step, leading to higher approximation error. By carefully balancing this trade-off, the ranking phase achieves both computational efficiency through controlled exploration and strong accuracy by minimizing the loss introduced during rounding.

Proposition 8. *Suppose there exists $W' \in \mathbb{R}_+^{n\times n}$ such that $1-\gamma \le W_{ij}/W'_{ij} \le 1+\gamma$ for all i, j, and W' has non-negative rank r_+ with an explicit non-negative factorization. Given any $\epsilon > 0$, Algorithm 16 achieves*

$$\begin{aligned}\mathrm{ALG}_{\mathsf{P}(I)} \ge\ & (1 - g(2\gamma(Vu)_{\max}))^2(1 - g(c_{\epsilon,\gamma,W'_{\min}}))^2\mathrm{OPT}_{\mathsf{P}(I)} \\ & - k(1 - g(2\gamma(Vu)_{\max}))(2\epsilon(1 - g(c_{\epsilon,\gamma,W'_{\min}})) + g(c_{\epsilon,\gamma,W'_{\min}})h(c_{\epsilon,\gamma,W'_{\min}}) \\ & + (1 - g(c_{\epsilon,\gamma,W'_{\min}}))^2 h(2\gamma(Vu)_{\max}) + h(2\gamma(Vu)_{\max})),\end{aligned}$$

where

$$c_{\epsilon,\gamma,W'_{\min}} = \frac{(1+\gamma)\epsilon}{W'_{\min}},$$

with runtime

$$\begin{aligned}& r_+ m \log_2 m + \lambda m^{\lambda} + \lambda r_+^2 m^{\lambda r_+} \\ & \quad + \left\lceil \left(\frac{4(1+\gamma)k}{\epsilon W'_{\min}}\right)^{r_+} \right\rceil \left\lceil \frac{(Vu)_{\max} - \min\{0, (Vu)_{\min}\}}{\epsilon} \right\rceil^{r_+} \\ & \quad \cdot \left\lceil \frac{\max_{i\in[m]}\{f_i((Vu)_{\max})\}}{\epsilon} \right\rceil \cdot \lambda' m^{\lambda r_+ + \lambda'} T_{\mathrm{LP}}.\end{aligned}$$

where $\lambda = \lceil (2r_+ + 2)(Vu)_{\max}/\epsilon \rceil$ and $\lambda' = \lceil (2r_+ + 2)\max_{i\in[m]}\{f_i((Vu)_{\max})\}/\epsilon \rceil$. Here, $T_{\mathrm{LP}} = \mathrm{LP}(m, 3m + r_+ + 2) + \mathrm{LP}(m, 2m + 2r_+ + 2)$ and $\mathrm{LP}(m, n)$ is the runtime of solving a linear program with m variables and n constraints.

Our proof of Proposition 8 appears in Appendix A.9.

2.5 Experimental Results

We conducted two sets of experiments with the following findings.

Representation. Simple transformers captured user preferences nearly as accurately as deeper transformer models. They achieved 14.1% higher accuracy than the best non-attention baselines (e.g., logistic regression, random forest, support vector machines), and only 2.5% lower accuracy than general multi-layer transformers, demonstrating strong representational power.

Optimization. Our two-phase algorithm (retrieval and ranking) efficiently optimizes simple-transformer-based recommendations. Compared against k-Nearest Neighbors for retrieval and Beam Search for ranking under a fixed candidate budget, our algorithm achieved objective values 20.86% and 20.56% higher than the respective benchmark combinations, outperforming natural baselines in both phases.

We used two dataset. The first dataset was the Spotify Million Playlist Dataset [47]. Spotify is one of the largest music streaming platforms, with over 640 million monthly active users, including 252 million paying subscribers. The dataset comprises one million user-generated playlists created on the Spotify platform between January 2010 and October 2017. Each playlist includes features such as the playlist title and the titles of the tracks it contains.

The second dataset was the Trivago Session-based Hotel Recommendations Dataset [101]. Trivago is a global hotel search platform that operates 55 localized websites across more than 190 countries, providing access to over two million hotels. The dataset consists of user sessions related to hotel search and booking, encompassing approximately 730,000 unique users and 340,000 unique hotels across roughly 900,000 sessions. Each session includes information on user interactions with hotels, such as clicks and checkouts. In addition, the dataset contains various hotel attributes, including price, city, and other relevant features.

Representation

In this set of experiments, our goal was to show that simple transformers were able to learn from user behaviors and predict user preferences with high accuracy. More specifically:

Spotify. In the Spotify experiment, we designated playlists containing 20 songs as "true" playlists. To construct "fake" playlists, we took the first 15 songs from each true playlist and appended 5 randomly selected songs. The number of true and fake playlists was balanced to be equal. Given a playlist, the task was to classify it as either true or fake. The performance of an algorithm was evaluated based on the average classification accuracy.

Trivago. In the Trivago experiment, each session provided information on a user's interactions with the first 15 hotels. The task was to predict the user's interactions with the subsequent 5 hotels in the same session. The performance of an algorithm was evaluated based on the average prediction accuracy.

We compared three classes of machine-learning algorithms on these prediction tasks:

- **Non-Attention Models:** This class included well-known machine learning algorithms that do not incorporate self-attention mechanisms, such as random guessing, logistic regression, support vector machines, and nearest neighbors. These models disregarded any potential sequential structure in the data.
- **Simple Transformers:** This class consisted of transformer architectures with a single self-attention layer, followed by linear layers and activation functions.
- **General Transformers:** This class contained more complex transformer architectures with multiple self-attention layers and potentially deeper network structures.

Below we give the architecture of the simple transformers used in both experiments.

Architecture used in Spotify. Let x_i denote the word2vec embedding of the i-th song in a playlist, after being processed by a linear layer. Each user vector is modeled as the average of the embeddings of the first 15 songs in the playlist, that is, $u = \sum_{j=1}^{15} x_j$. Let $\mathrm{SA}_{Q,K,V}(\cdot)$ be the self-attention layer with learned parameters Q, K, V, and let $f_i(\cdot) = f(\cdot)$ be an activation function composed of a linear transformation and a logistic function, both parameterized and learned during training. Let S be the set of indices corresponding to the 16th through 20th songs in the playlist. Then, the output of the simple transformer is given by

$$\sum_{i=1}^{5} f(\mathrm{SA}_{Q,K,V}(X_S)_i^\top u),$$

where X_S denotes the input embeddings corresponding to songs in set S. General transformer models extend this architecture by incorporating additional self-attention layers.

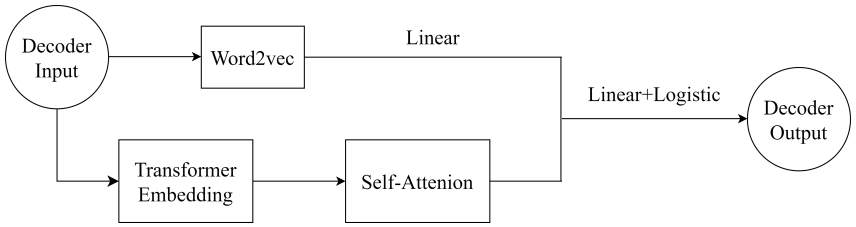


Figure 2.1: Architecture of the simple transformer used in the Spotify experiment.

Architecture used in Trivago. Let x_i denote the learned embedding of the i-th hotel. Each user vector is modeled as the average of the embeddings of the first 15 hotels the user engaged within a given session, that is, $u = \sum_{j=1}^{15} x_j$. Let $\mathrm{SA}_{Q,K,V}(\cdot)$ denote a self-attention layer with learned parameters Q, K, and V, and let $f_i(\cdot) = f(\cdot)$ be an activation function composed of a linear transformation and a logistic function, both parameterized and learned during training. Let S be the set of indices corresponding to the 16th through 20th hotels in the session. Then, the output of the simple transformer is given by

$$\sum_{i=1}^{5} f(\mathrm{SA}_{Q,K,V}(X_S)_i^\top u),$$

where X_S denotes the input embeddings corresponding to hotels in set S. The simple transformer architecture consisted solely of a decoder with one self-attention layer. General transformer models extended this architecture by incorporating an encoder, as well as additional self-attention layers in both the encoder and decoder.

The experimental results are presented in the tables below.

	Random Forest	Logistic Regression	Support Vector Machine	Simple Transformer	General Transformers
Spotify	0.518	0.520	0.334	0.702	0.726
Trivago	0.271	0.530	0.531	0.631	0.742

Table 2.1: Average accuracy of different machine-learning models on Spotify and Trivago.

In Table 2.1, the simple transformer outperformed non-attention models by average accuracy margins of 0.182 on Spotify and 0.20 on Trivago, while achieving performance comparable to general transformers. On Trivago, it outperformed several

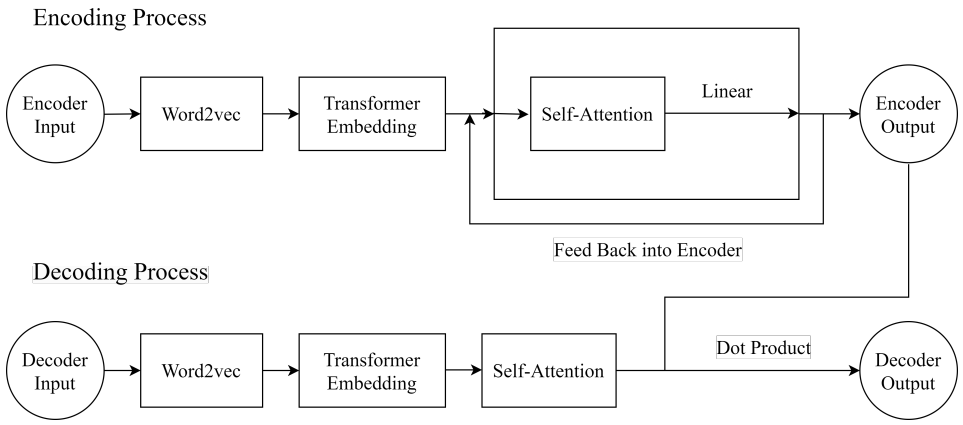


Figure 2.2: Architecture of the transformer used in the Trivago experiment. The simple transformer only contained the decoder, that is, a single self-attention layer.

Dec. Layers \ Enc. Layers	1	2	4
1	0.590	0.602	0.596
2	0.654	0.692	0.700
4	0.724	0.742	0.675

Table 2.2: Average accuracy of general (full encoder–decoder) transformers with various numbers of self-attention layers on Trivago.

more complex architectures and was only 2.4% below the average accuracy of all general transformers. Since the optimal architecture is not known a priori, simple transformers offer a strong and robust choice, delivering substantially higher accuracy than non-attention models while nearly matching general transformers.

Optimization

In the previous set of experiments, we demonstrated that simple transformers could empirically capture user preferences on both datasets. In this set of experiments, we turned to the task of personalized recommendation based on simple transformers. We treated the parameters Q, K, and V learned in the previous experiments as ground truth and solved Problem (Main). Each instance corresponded to an arriving user. More specifically:

Spotify. In the Spotify experiment, we were given 15 songs as input, and the task was to recommend an additional 5 songs to complete a 20-song playlist. The given 15 songs were treated as a representation of the user. The corresponding user vector u was computed by first averaging the word2vec embeddings of the 15 songs, and then applying a linear transformation to project the result into the value space. The

key, query, and value vectors of each song were obtained from the parameters learned in the previous experiments.

Trivago. In the Trivago experiment, we were given 15 user interactions as input, and the task was to recommend 5 additional hotels to maximize the booking rate. The user vector u was computed by averaging the learned embeddings of the 15 hotels with which the user had interacted. The key, query, and value vectors of each hotel were similarly obtained from the parameters learned in the previous experiments.

Recall that our algorithm operates in two phases: retrieval and ranking. Our algorithm performs these two phases as described below:

Phase One (Retrieval): Our algorithm partitions the row space of the query matrix Q and the key matrix K offline, with the number of partitions treated as a tunable hyperparameter. Then, when a user vector u arrives, for each group of items whose query and key vectors both belong to the same corresponding partition, we apply the k-Nearest Neighbor algorithm and retain the k items with the highest base reward $v^\top u$. All retained items are then passed to the ranking phase.

Phase Two (Ranking): Our algorithm first enumerates all possible combinations of the top c highest-reward items to include in a candidate solution, referred to as *valid tuples* in the proof of Proposition 8, where c is a tunable hyperparameter. For the remaining items, the algorithm solves the residual subproblem by evaluating a fixed number of auxiliary problems, denoted as $\mathsf{P}(X, t)$ in the same proof. The number of auxiliary problems solved was tuned to control the total number of candidate solutions generated.

We compared each phase of our algorithm against a natural benchmark:

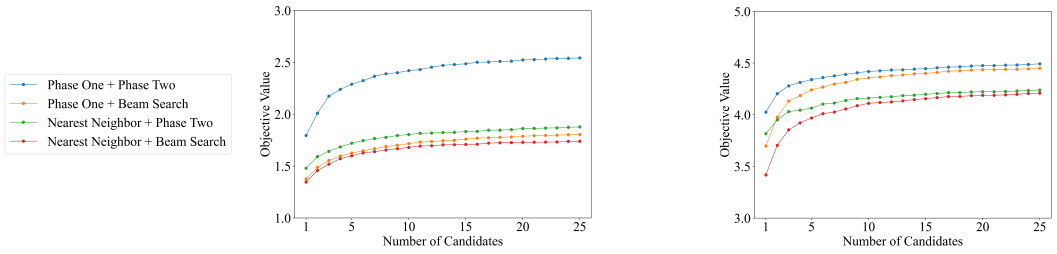
k-Nearest Neighbor (Retrieval): The k-Nearest Neighbor algorithm served as the benchmark for the retrieval phase. It ignored any potential sequential effects and instead ranked items solely based on their base rewards, computed as $v^\top u$. The algorithm then greedily selected the same number of items as our algorithm's retrieval phase to pass on to the ranking phase.

Beam Search (Ranking): Beam Search served as the benchmark for the ranking phase. Beam Search is a greedy-type heuristic commonly used in practice. Each candidate solution generated by Beam Search is specified by a k-tuple $(b_1, \ldots, b_k)$.

To select the ℓ-th item in the candidate solution, the algorithm evaluated each item not yet included by computing the incremental gain in the objective value if the item were added. It then selected the item corresponding to the b_ℓ-th highest increment and added it to the candidate solution. In particular, the tuple $(1, \ldots, 1)$ corresponds to the fully greedy solution. The values of $b_1, \ldots, b_k$ were tuned based on the desired number of candidate solutions.

Combining our retrieval and ranking phases with benchmark methods yields four algorithms, whose performance is shown in Figure 2.3. Our complete algorithm consistently outperformed all others across all candidate budgets.

Fixing the ranking phase (to either our Phase Two or Beam Search), our Phase One outperformed k-Nearest Neighbor by an average of 20.86%, demonstrating superior retrieval. Conversely, fixing the retrieval phase (to either our Phase One or k-Nearest Neighbor), our Phase Two outperformed Beam Search by 20.56%, highlighting its effectiveness in ranking. Overall, our algorithm achieved strong empirical performance in both phases, combining high efficiency with high accuracy.

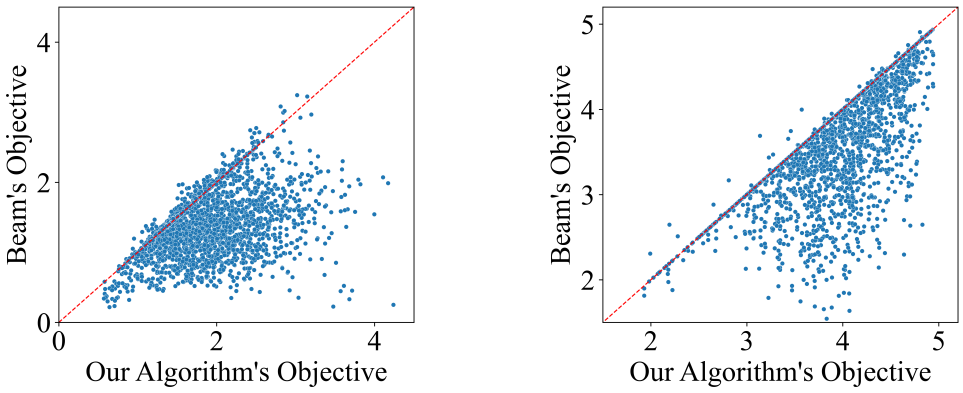


(a) Spotify (b) Trivago

Figure 2.3: Performances of four algorithms. The x-axis is the number of candidate solutions generated by each algorithm, and the y-axis is the objective value of the current best candidate solution. Each figure is averaged across 100 instances.

We further compare our Phase Two with Beam Search using scatter plots, fixing Phase One as the retrieval step. For ranking, both methods generated candidate solutions with the same computational budget: our Phase Two greedily solved a fixed number of auxiliary problems, while Beam Search branched the same number of times.

Each point in Figure 2.4 compares matched candidate solutions, with the x-axis showing our objective value and the y-axis that of Beam Search. Our Phase Two outperformed Beam Search in 90.92% of instances, with an average improvement of 29.01%, demonstrating substantial gains in the ranking phase.



(a) Spotify

(b) Trivago

Figure 2.4: Scatter plots of candidate solutions. Each point corresponds to a pair of matched candidate solutions produced by our algorithm and Beam Search. The x-axis represents our algorithm's objective value and the y-axis represents the Beam Search candidate solution's objective value. Each plot contains 2500 data points given by 25 candidate solutions in each of the 100 instances.

2.6 Conclusion

In conclusion, we study real-time personalization using simple transformers, a single-self-attention-layer architecture that is both expressive and computationally efficient. We show that simple transformers capture key user preference structures, including sequential variety effects and pairwise complementarity and substitution, and develop a sub-linear-time algorithm achieving near-optimal performance. Empirically, simple transformers outperform non-transformer baselines, remain competitive with deeper transformers, and our algorithm improves objective values over standard methods such as k-Nearest Neighbor and Beam Search.

Chapter 3

ONLINE RESOURCE ALLOCATION WITH PREDICTIONS UNDER UNKNOWN ARRIVAL MODEL

Online decision-makers often have access to predictions about future quantities such as arrivals, demand, or inventories, generated by methods ranging from simple time-series forecasts to modern machine-learning models. Because prediction accuracy is unknown a priori, blindly following such predictions can be harmful. We address this challenge by developing algorithms that leverage predictions in a manner robust to unknown prediction accuracy.

We study the Online Resource Allocation Problem, a general model for online decision-making in which limited resources must be allocated to a sequence of arrivals. Prior work has characterized optimal performance under both stochastic and adversarial arrivals, and designed algorithms that achieve these bounds without knowing the underlying arrival model. Building on this framework, we introduce predictions in the form of resource shadow prices and define prediction accuracy as the distance between predicted and true shadow prices.

We provide a tight lower bound characterizing how well any algorithm can exploit predictions by following them when accurate and ignoring them when inaccurate, without knowing either the prediction accuracy or the arrival model. We then propose an algorithm that matches this bound and empirically validate its performance using large-scale real data from the retailer *H&M*.

3.1 Introduction

Allocating a limited set of resources to satisfy different requests as they arrive is a key process in many operations problems. For example, airlines need to decide whether or not to accept a certain offer for a seat at a given price, while the total number of seats is limited [18, 160]; online retailers must choose which products to display to a browsing customer, taking into account inventory levels [70, 121]; internet search engines auction off impressions to advertisers with limited budgets [64, 130]. The *Online Resource Allocation Problem* is a generic model for all of these settings. In the problem, requests arrive sequentially, each request consisting of multiple actions to choose from, and each action generating some reward and consuming some subset

of resources. Actions are selected online, i.e. without knowing future requests. Resources are limited, and the objective is to maximize the total reward received across all time periods. While the Online Resource Allocation Problem is ubiquitous in practice, we briefly highlight several motivating examples:

- **Network Revenue Management:** In airline revenue management, resources correspond to seats on future flights. Requests may involve multiple seats (e.g., group bookings or multi-leg itineraries) and have highly variable prices due to numerous fare classes.
- **Assortment Optimization:** In online retail, a seller selects assortments to display during a customer's browsing session (e.g., search results or recommendations). Each display opportunity is a request, rewards are expected profits from customer choices, and resources correspond to product inventories.
- **Online Matching (AdWords, Online Auctions):** Online matching models two-sided markets such as AdWords and auctions. Arrivals correspond to online nodes (e.g., impressions), while resources correspond to capacities of offline nodes (e.g., advertiser budgets).

Broadly, there are two main approaches to Online Resource Allocation. The traditional approach assumes a model for arrivals and designs algorithms with optimal worst-case guarantees, most commonly under either a *stochastic* model, where arrivals are drawn i.i.d. from an unknown distribution, or an *adversarial* model, where no assumptions are made. [22] shows that the best possible worst-case performance can be achieved simultaneously under both models without knowing the true arrival model. Interpreting stochastic and adversarial arrivals as stationary and nonstationary processes, respectively, their algorithm learns effectively in stationary settings while remaining robust to arbitrary nonstationarity. However, this notion of optimality is with respect to worst-case guarantees and may be overly pessimistic in practice.

The second, arguably more modern approach, is to utilize some sort of *predictions* on the future arrivals. Here we use the term "prediction" in the broadest possible sense, ranging from simple time-series forecasting models, to state-of-the-art machine learning algorithms based on large amounts of data, to human judgement, and even combinations of all of the above. The de facto approach in practice is to take these predictions as fact (in a way we will make formal momentarily). Naturally, the

performance of this approach relies heavily on the accuracy of the predictions, which is not guaranteed: Figure 3.1, taken from [5], shows this for the relatively simple task of forecasting daily visits to two stores.

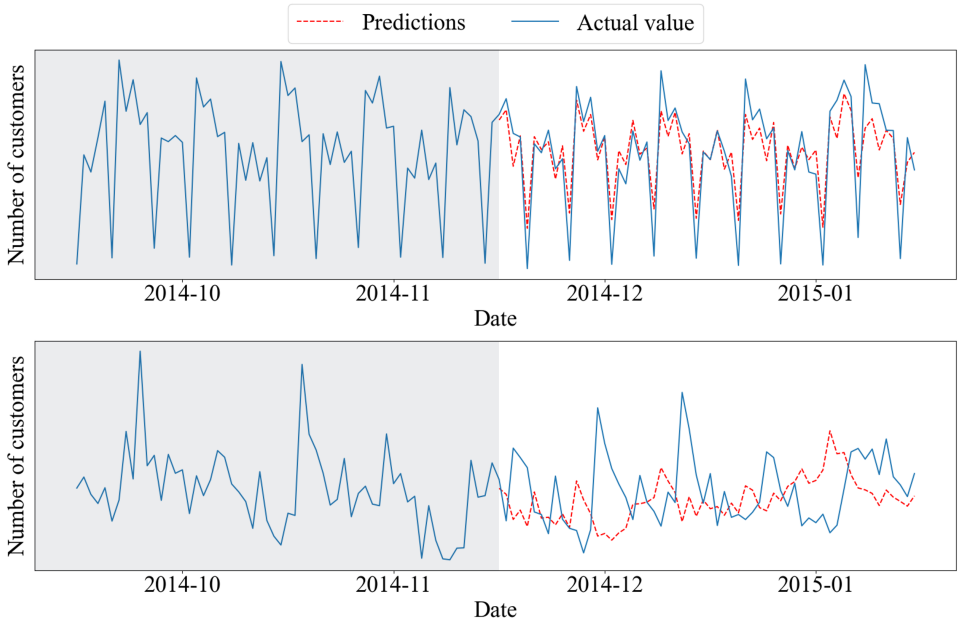


Figure 3.1: (Figure and caption from [5]) Daily number of customers (in blue), from September 2014 to January 2015, at two different stores in the Rossmann drug store chain. Predictions (in red), starting November 2014, are generated using Exponential Smoothing with the same fitting process. The store in the upper sub-figure has substantially more accurate predictions ($R^2 = 0.88$) than that of the lower sub-figure ($R^2 = 0.11$).

To summarize so far, the Online Resource Allocation Problem admits algorithms with optimal worst-case guarantees (for both stochastic and adversarial arrival models, simultaneously), and these algorithms can be significantly better or worse than following predictions, depending on the prediction quality. This suggests the opportunity to design an algorithm that leverages predictions *optimally*, in the sense that the predictions are utilized when accurate, and ignored when inaccurate. Ideally, such an algorithm should operate without knowledge of (a) the accuracy of the predictions and (b) the method with which they are generated. *This is precisely what we seek to accomplish in this paper.*

Online Resource Allocation with Predictions

The primary purpose of this paper is to develop an algorithm that optimally incorporates *predictions* (defined in the most generic sense possible) into the Online

Resource Allocation Problem. Without predictions, the *Online Resource Allocation Problem* consists of a finite horizon of T time periods and a limited number of m types of resources. At each time period, a decision must be made which will consume a certain set of resources and yield a certain reward. The form of these individual decision problems changes over time and is unknown in advance.

Following [22], we consider both stochastic and adversarial arrival models. Under the stochastic model, performance is measured by *regret*, defined as the difference between the total reward of an optimal offline algorithm (which knows the entire arrival sequence) and that of the online algorithm. Our goal is to achieve *sublinear* regret ($o(T)$), which ensures asymptotically optimal per-period performance. Under the adversarial model, sublinear regret is impossible in the worst case, and performance is instead measured by the *competitive ratio*, defined as the ratio between the optimal offline reward and the algorithm's reward; an α-competitive algorithm guarantees at least a $1/\alpha$ fraction of the offline optimum.

Without predictions, [10] showed that any algorithm incurs at least $\Omega(T^{1/2})$ regret under stochastic arrivals, while [21] proved that under adversarial arrivals any algorithm has competitive ratio at least α^*, where α^* depends on simple problem parameters. These bounds are matched simultaneously by [22]. Our objective is to design algorithms that improve upon these worst-case guarantees when predictions are accurate, while retaining the same guarantees when predictions are inaccurate.

To that end, we introduce the notion of *predictions*. Our prediction is of the form of an m-dimensional vector $\hat{\mu}$ whose coordinates represent a predicted *shadow price* for each of the m resources. In particular, $\hat{\mu}$ may carry some information about the future (realized) arrival sequence. We will show that this form of prediction satisfies certain nice properties including that it (a) immediately translates to a decision policy, and (b) there always exists "perfect" predictions which achieve near-optimal reward.

We measure the quality of any prediction $\hat{\mu}$ by its ℓ_1 distance to the closest perfect prediction μ^*.[1] Specifically, we use an *accuracy parameter* $a \geq 0$, defined as the largest a such that $||\hat{\mu} - \mu^*||_1 \in O(T^{-a})$. Notice that when $a = 0$ the prediction is effectively useless, and as a increases the prediction becomes more accurate. We call this problem *Online Resource Allocation with Predictions*. Our primary challenge will be to design algorithms with performances that are robust in the prediction quality *without* having access to a.

[1]The choice of the ℓ_1 norm follows naturally from our analysis, though any ℓ_p norm where $p \in [1, 2]$ yields similar performance guarantees for our algorithms.

A Simple Example

Figure 3.2: Two potential arrival sequences for an online resource allocation problem with a single resource (two lemons) and two time periods. The left (right) sequence falls under the stochastic (adversarial) arrival model.

Before outlining our contributions, it is worth describing a simple example to illustrate the challenge we face in incorporating predictions of unknown accuracy. Consider the example in Figure 3.2, which depicts two potential arrival sequences for an online resource allocation problem with a single resource (two lemons) and two time periods. In both sequences, a single lemon may be sold for \$1 in the first time period. This same offer occurs in the second time period for the left sequence, but the right sequence offers \$2 for two lemons (this offer may not be split). Note that the left (right) sequence falls under the stochastic (adversarial) arrival model, and critically, an algorithm can not distinguish between the two sequences until the second time period. Still, a simple algorithm (accept all offers when feasible) achieves zero regret under the left sequence, and a competitive ratio of $1/2$ under the right sequence (incidentally, $\alpha^* = 2$ for this problem instance).

However, suppose now we introduce a prediction, whose implication is that the first offer should be rejected. Under the right sequence, this constitutes a "good" prediction, and so an algorithm ideally would follow this prediction and collect the optimal \$2. Under the left sequence, this constitutes a "bad" prediction, and so an algorithm ideally would ignore this prediction but still achieve good regret as the arrival model is stochastic. It is of course impossible to do both of these.

More generally, there are essentially four "worlds" we must consider, depending on whether the arrival model is stochastic or adversarial, and whether the predictions are accurate or inaccurate. This example demonstrates that we can not hope to achieve the best of all four worlds simultaneously.[2] Instead, we will find that just as the accuracy of the predictions is best characterized continuously between "perfect" and "bad" (via our accuracy parameter a), the arrival model is best characterized continuously along a carefully-defined interpolation between the stochastic and adversarial models.

Finally, we emphasize that under the stochastic arrival model, a prediction may legitimately carry information about the realized arrival sequence and thus be

[2]The language here is indeed in reference to the "best of both worlds" literature, e.g. [22].

statistically correlated with it. For example, suppose lemon demand depends on both predictable factors, such as weather or seasonality, and unpredictable idiosyncratic shocks, such as random household consumption variation. A prediction that accurately captures the predictable components, such as weather, will be correlated with the realized arrivals, even though it does not fully determine them. In this case, the remaining uncertainty arises only from the idiosyncratic shocks, which may have relatively low variance. This illustrates how informative predictions can naturally arise in stochastic environments and remain useful across all realizations, despite not being perfectly accurate.

Our Contributions

Our primary contributions can be summarized as follows.

1. A Nonstationary Arrival Model and a Lower Bound. We define a parameterized class of arrival models that interpolates between the stochastic and adversarial models. In particular, we define a precise measure of the stationarity of an arrival sequence (Definition 4), in terms of two values λ and δ, such that $(\lambda, \delta) = (0, 0)$ (loosely) corresponds to the stochastic model, and $(\lambda, \delta) = (1, 1)$ corresponds to the adversarial model (the two values are in general distinct, and have nice time-series interpretations in terms of trend and seasonality). Moreover, for any fixed arrival sequence, every δ corresponds to some λ for which the arrival sequence is (λ, δ)-stationary. Therefore, δ can be viewed as a parameter set by us. Later we will see that the choice of λ and δ affects our algorithm's performance and runtime. In particular, our algorithm needs to calculate the offline optimum for $1/\delta$ times. Notably, this stationarity measure is defined for deterministic arrival sequences, and thus the corresponding (nonstationary) arrival models can be defined without positing a stochastic generative model.

The primary value of this new measure of stationarity is that it tightly characterizes the extent to which we can expect an algorithm to leverage predictions of unknown quality. Specifically, we prove the following lower bound:

Proposition 6 (Lower Bound, Informal)**.** *For any* $0 \le \lambda \le 1$ *and* $0 < \delta \le 1$*, and any algorithm, at least one of the following holds:*

(1) Under the stochastic arrival model, the algorithm incurs $\Omega(T)$ *worst-case regret;*

(2) Under the adversarial arrival model, with (λ, δ)-stationary arrivals, the algorithm's worst-case reward is at most

$$(1-\lambda)\max\left\{\frac{1}{\alpha^*}\text{OPT}, \text{PRD}\right\} - \Omega(\delta T).$$

Here recall that α^* is the best competitive ratio (without predictions) which we will specify later. OPT denotes the optimal offline reward. Following the actions induced by the predictions also yields a certain amount of reward, which we denote by PRD.

Now if $(\lambda, \delta) = (1, 1)$, i.e. the adversarial model with no restrictions, then Proposition 7 implies that we can not simultaneously achieve sub-linear regret under the stochastic arrival model and a meaningful reward under the adversarial model (our lemon example was already evidence of this). However, for smaller values of λ and δ, we can hope for sub-linear regret and an adversarial reward close to the best value of $\max\left\{\frac{1}{\alpha^*}\text{OPT}, \text{PRD}\right\}$.

2. An Optimal Algorithm. We construct an algorithm that *optimally* leverages predictions, in the sense that it achieves the lower bound of Proposition 7, without knowing the underlying arrival model (stochastic or adversarial) and without knowing the prediction accuracy. In particular, our main theoretical result is the following:

Theorem 2 (Upper Bound, Informal)**.** *Given a prediction $\hat{\mu}$ with (unknown) accuracy parameter a and given $0 < \delta \leq 1$, there exists an algorithm such that, under mild (and tight) assumptions, both of the following hold:*

(1) Under the stochastic arrival model, the algorithm incurs $\tilde{O}(T^{\frac{1}{2}-a})$[3] worst-case regret;

(2) Under the adversarial arrival model, with (λ, δ)-stationary arrivals, the algorithm's worst-case reward is at least

$$(1-\lambda)\max\left\{\frac{1}{\alpha^*}\text{OPT}, \text{PRD}\right\} - O(\delta T).$$

Our theoretical results are summarized in bold in Table 3.1, with a comparison to the problem with no predictions and the problem with predictions of known accuracy.

[3]The $\tilde{O}(\cdot)$ notation hides logarithmic factors. Technically the regret should be $\tilde{O}(\max\{T^{\frac{1}{2}-a}, 1\})$, since if $a > \frac{1}{2}$ the regret bound should be a constant. For the simplicity of exposition we drop the obvious regret bound of $O(1)$ in the introduction section.

Arrival Model	Without Predictions	With Predictions of Known Accuracy	With Predictions of Unknown Accuracy
Stochastic (regret)	$O(T^{\frac{1}{2}})$	$O(T^{\frac{1}{2}-a})$	$\boldsymbol{\tilde{O}(T^{\frac{1}{2}-a})}$
Adversarial (reward)	$\frac{1}{\alpha^*}$OPT	$\max\{\frac{1}{\alpha^*}\text{OPT}, \text{PRD}\}$	$\boldsymbol{(1-\lambda)\max\{\frac{1}{\alpha^*}\text{OPT}, \text{PRD}\} - \delta T}$

Table 3.1: Summary of our main theoretical results (in bold). Each entry has a corresponding algorithm that, without knowing the underlying arrival model, achieves the stated performance simultaneously for both stochastic arrivals and adversarial arrivals. Each entry also has a matching lower bound.

3. A Large-Scale Experiment. We evaluate the practical value of our model and algorithm using an H&M (Hennes & Mauritz AB) dataset containing two years of transactions for 105,542 products. The experiment corresponds to the assortment optimization problem and consists of three-month simulated horizons. We compare our algorithm against two natural baselines: the optimal algorithm without predictions and a policy that always follows the predictions. For each instance, these baselines define the worst- and best-achievable rewards, and we measure performance by the fraction of this gap captured by our algorithm; values close to 1 indicate near-best performance.

We generate predictions of varying quality using three popular forecasting methods. The average optimality gap achieved by our algorithm is 0.68 with Prophet forecasts, 0.58 with ARIMA forecasts, and 0.53 with Exponential Smoothing, demonstrating robust performance across prediction qualities.

Literature Review

Online Resource Allocation. Online resource allocation has been extensively studied under different arrival models. Under stochastic arrivals (i.i.d. or random order), the goal is to achieve sublinear regret, as in [1, 57, 58, 68, 75, 97]. Under adversarial arrivals, sublinear regret is impossible, and performance is measured by competitive analysis; classic AdWords results include [40, 130], which achieve a $(1 - 1/e)$-competitive ratio.

Recent work studies algorithms that perform well under both models without knowing the arrival process. [133] achieved the optimal adversarial competitive ratio and improved stochastic performance for AdWords, while [22] proposed a mirror-descent algorithm achieving optimal stochastic regret and adversarial competitive ratio for Online Resource Allocation. Our algorithm also attains optimal performance under

both models, but differs fundamentally in design and analysis and is the first to robustly leverage predictions.

Our approach is based on gradient descent in the dual space, which has been widely applied in binary integer programming [114], online linear programming [113], bandits with knapsacks [115], online stochastic optimization [88], and online resource allocation [22]. We further draw on parameter-free optimization [43], using adaptive step sizes to exploit predictions without knowing their quality.

Algorithms with Predictions. Motivated by advances in data and machine learning, there is growing interest in augmenting online algorithms with predictions to go beyond worst-case guarantees. This paradigm has been applied to revenue optimization [20, 72, 137], caching [123, 149], online matching [89, 107], online scheduling [106, 146], the secretary problem [9, 62, 63], and the nonstationary newsvendor [5]. Most of this literature uses competitive analysis and characterizes optimal consistency–robustness trade-offs.

In contrast, under stochastic arrivals we perform regret analysis and show near-optimal worst-case regret without knowing prediction quality. Other regret-based analyses with predictions include [5, 81, 137].

The closest works are [20] and [72], which study strict special cases of Online Resource Allocation and treat prediction quality as binary. Under this assumption, they achieve optimal consistency–robustness trade-offs. In contrast, we quantify prediction quality and provide tight guarantees for arbitrary accuracy.

3.2 Model: The Online Resource Allocation with Predictions

In this section, we first formally define the **Online Resource Allocation with Predictions** problem, and then describe two standard arrival models (stochastic and adversarial) as well as their respective performance metrics.

Problem Formulation

Online Resource Allocation. Consider a problem over T time periods labeled $t = 1, \ldots, T$. Assume there are m different types of resources. The total number of resources available is denoted by ρT, where $\rho \in \mathbb{R}_+^m$ is a non-negative m-dimensional vector. At each time period t, the decision-maker receives an *arrival* $\gamma_t = (r_t, g_t, \mathcal{X}_t) \in \mathcal{S}$. Here, $r_t : \mathcal{X}_t \to \mathbb{R}_+$ is a non-negative reward function, $g_t : \mathcal{X}_t \to \mathbb{R}_+^m$ is a non-negative resource consumption function, $\mathcal{X}_t \subset \mathbb{R}_+^d$ is a

compact action space, and $\mathcal{S}$ denotes the set of all possible arrivals.[4] Note that we impose no convexity assumptions: $r_t(\cdot)$ can be non-concave, $g_t(\cdot)$ can be non-convex, and $\mathcal{X}_t$ can be non-convex or discrete. At each arrival γ_t, without knowing any of the future arrivals, an action $x_t \in \mathcal{X}_t$ must be selected, which yields $r_t(x_t)$ reward and consumes $g_t(x_t)$ resources. The objective is to maximize the total reward subject to the resource constraint. Finally, we assume that $\mathcal{X}_t$ always contains a 0 vector representing a "void" action that consumes no resources and yields no rewards: $r(0) = 0$ and $g_t(0) = 0$. This ensures that there is always a feasible action available. This is the problem we will refer to as **Online Resource Allocation (without predictions)**.

We introduce some notations that will appear in our results later on (though our algorithms will not depend on these parameters). We denote by $\underline{\rho} = \min_{j\in[m]} \rho_j > 0$ the lowest resource parameter and $\bar{\rho} = \max_{j\in[m]} \rho_j = ||\rho||_\infty$ the highest resource parameter. Similarly, let $\bar{r} \geq 0$ be a constant which satisfies $\max_{x\in\mathcal{X}} r(x) \leq \bar{r}$ for every $(r, g, \mathcal{X}) \in S$, and let $\bar{g} \geq \underline{g} > 0$ be constants satisfying $\underline{g} \leq ||g(x)||_\infty \leq \bar{g}$ for every $x \in \mathcal{X}$ and $x \neq 0$.

Primal and Dual. For any arrival sequence $\gamma = (\gamma_1, \ldots, \gamma_T)$, we use $\mathrm{OPT}(\gamma)$ to denote the *offline/hindsight optimum*, which is the reward of the optimal solution when γ is known in advance:

$$\mathrm{OPT}(\gamma) := \max_{x_t\in\mathcal{X}_t} \sum_{t=1}^{T} r_t(x_t) \quad \text{s.t.} \quad \sum_{t=1}^{T} g_t(x_t) \leq \rho T. \tag{3.1}$$

As we will describe momentarily, it will be natural to consider predictions in terms of the dual space, so the Lagrangian dual problem of Eq. 3.1 plays a key role. Let $\mu \in \mathbb{R}^m_+$ be the vector of dual variables, where each μ_j can be thought of as the shadow price of resource j. We define

$$r^*_t(\mu) := \sup_{x\in\mathcal{X}_t} \{r_t(x) - \mu^\top g_t(x)\} \tag{3.2}$$

as the optimal opportunity-cost-adjusted reward of request γ_t, where the opportunity cost is calculated according to the shadow prices μ.[5] Note that $r^*_t(\mu)$ is a generalization

[4] We assume throughout the paper that the reward functions and the resource consumption functions are deterministic for any given action, but our algorithms also apply when the rewards and/or consumed resources are random.

[5] We will assume that the primal optimization problems in Eq. 3.2 admit an optimal solution. This is to simplify the exposition: our results still holds if we have an approximate of the optimal solution (see [22]).

of the convex conjugate of $r_t(x)$ that takes the resource consumption function $g_t(x)$ and the action space $\mathcal{X}_t$ into account. In particular, when $g_t(x) = x$ and $\mathcal{X}_t$ is the whole space, $r_t^*(\mu)$ becomes the standard convex conjugate. For fixed arrivals γ, we define the Lagrangian dual function $D(\mu \mid \gamma) : \mathbb{R}_+^m \to \mathbb{R}$ to be

$$D(\mu \mid \gamma) := \sum_{t=1}^{T} r_t^*(\mu) + \mu^\top \rho T. \tag{3.3}$$

This allows us to move the constraints of Eq. 3.1 to the objective, which is easier to work with. By weak duality, $D(\mu \mid \gamma)$ provides an upper bound on $\mathrm{OPT}(\gamma)$. Moreover, even without any convexity assumptions, the duality gap of our problem is upper bounded by a constant that is independent from the time horizon T. This can be shown via Shapley-Folkman Theorem (see Proposition 5.26 of [34] for a detailed explanation). We formally state and prove the weak duality result and the duality gap result in Appendix B.1. We equip the primal space of the resource constraints $\mathbb{R}^m$ with the ℓ_∞ norm $||\cdot||_\infty$, and the Lagrangian dual space with the ℓ_1 norm $||\cdot||_1$. Such choices of norms come naturally from our analysis. Similar performance guarantees of our algorithms with the dependence on the number of resources[6] can be obtained using the ℓ_p norm for the primal space and the ℓ_q norm for the primal space with $1/p + 1/q = 1$ and $p \in [2, \infty]$.

Predictions. So far, we have presented the problem of Online Resource Allocation without predictions. As described in the introduction, it is often the case that when this problem is faced in practice, some notion of a "prediction" can be made which might guide us in selecting actions. Such predictions can come from a diverse set of sources ranging from simple human judgment, to forecasting algorithms built on previous demand data, to more-sophisticated machine learning algorithms trained on feature information. The process of sourcing or constructing such predictions is orthogonal to our work. Instead, we treat these predictions as given to us endogenously (and in particular, we make no assumption on the accuracy of these predictions), and attempt to use these predictions optimally.

Notice that from Equation (3.2), at each time t, given a dual variable $\mu \in \mathbb{R}_+^m$ there is a natural action to take, namely the action

$$x_t^\mu \in \mathrm{argmax}_{x \in \mathcal{X}_t, \sum_{s=1}^{t-1} g_s(x_s) + g_t(x) \le \rho T} \left\{ r_t(x) - \mu^\top g_t(x) \right\}.^7$$

[6] The number of resources m is viewed as a constant throughout the paper.

[7] We again assume this optimization problem and other similar-style optimization problems in this paper admit an optimal solution to simplify the exposition. When the right hand side contains more than one action, we naturally choose the action that has the highest reward.

In words, x_t^μ is the "greedy" action that, subject to the resource constraint, maximizes the opportunity-cost-adjusted reward according to the shadow prices μ. Therefore each dual variable μ essentially corresponds to an algorithm, which is simply taking the "greedy" action x_t^μ at each time period. Below we formally define this algorithm for any dual variable μ, which we call the *Dual-Adjusted Greedy Algorithm* (GRD_μ):

Algorithm 1: Dual-Adjusted Greedy Algorithm GRD_μ

Inputs: Dual variable μ, total time periods T, initial resources $G_1 = \rho T$
for $t = 1, \ldots, T$ **do**
  Receive request $(r_t, g_t, \mathcal{X}_t)$
  Make the primal decision x_t and update the remaining resources G_t:

$$x_t \in \arg\max_{x \in \mathcal{X}_t, g_t(x) \le G_t} \{r_t(x) - \mu^\top g_t(x)\}$$
$$G_{t+1} \leftarrow G_t - g_t(x_t).$$

end

Let $R(\text{GRD}_\mu \mid \gamma)$ denote the reward obtained by GRD_μ with arrival sequence γ and dual variable μ.[8] For an arrival sequence γ, we say a dual variable $\mu^*(\gamma)$ is a *"perfect" dual variable with respect to* γ if, on arrival sequence γ, the reward obtained by $\text{GRD}_{\mu^*(\gamma)}$ is at most an additive constant away from $\text{OPT}(\gamma)$ (hence essentially optimal). To simplify notations, from now on we will use μ^* in place of $\mu^*(\gamma)$. It can be shown that there always exists a "perfect" dual variable:

Proposition 1 ("Perfect" Dual Variable)**.** *For any arrival sequence* γ*,*

$$\max_{\mu \in \mathbb{R}_+^m} R(\text{GRD}_\mu \mid \gamma) + (\bar{g}/\underline{g} + 1)(m+1)\bar{r} \ge \text{OPT}(\gamma).$$

The proof of Proposition 1 appears in Appendix B.1, and utilizes the Shapley-Folkman Theorem. In words, Proposition 1 shows that there exists a dual variable μ^* that is essentially optimal to follow.

With the understanding of the key role that dual variables play in our problem, we formally introduce the notion of predictions. Because dual variables induce actions, they are natural quantities to predict. For an arrival sequence γ, we assume that before the first time period, the decision-maker receives a **prediction** $\hat{\mu}(\gamma) \in \mathbb{R}_+^m$ of the dual variable. Similar as μ^*, to simplify notations, from now on we will use $\hat{\mu}$ in place of $\hat{\mu}(\gamma)$. We measure the **prediction error** of $\hat{\mu}$ by $||\hat{\mu} - \mu^*||_1$, its ℓ_1 distance

[8] Later we will formally extend the notion of $R(\text{ALG} \mid \gamma)$ to any algorithm ALG.

from μ^*. [9] We quantify the prediction error using the **accuracy parameter**, which is the smallest $a \in [0, \infty]$ such that

$$||\hat{\mu} - \mu^*||_1 \leq \kappa T^{-a}.$$

Here $\kappa > 0$ is a scaling constant that we can choose, and any κ that ensures $a \in [0, \infty]$ can be chosen without affecting our performance bound asymptotically.[10] Note that $\hat{\mu}$ and μ^* both depend on γ, so the definition makes sense even if γ is itself random. The two extreme cases of the prediction error are (1) $a = 0$, in which case $\hat{\mu}$ is almost a constant away from μ^*, so the prediction is effectively useless; and (2) $a = \infty$, in which case $\hat{\mu} = \mu^*$, so the prediction is perfect. We will always assume that a is unknown to the decision-maker.

In reality, a prediction $\hat{\mu}$ is unlikely to be completely useless. We make the following technical assumption on the prediction quality:

Assumption 1 (Non-trivial Prediction)**.** *There exists a (known) function $\epsilon(T) = o(1)$ such that $||\hat{\mu} - \mu^*||_1 = o(\epsilon(T))$.*

Note that Assumption 1 does not eliminate the case $a = 0$. In practice $\epsilon(T)$ can be chosen to be a function close to 1 without hurting the algorithm's performance guarantee.

Aside: Predicting the Dual Variables. In many applications, historical data (e.g., arrivals from a previous month) is available and can be used to *warm start* dual variables, computed offline without knowing the arrival order and interpreted as predictions (shadow prices) for resource values. While imperfect, such data often provides informative signals.

Predicting dual variables is a long-standing idea in both theory and practice. For example, in online matching under random order, [57, 167] compute dual variables from an initial prefix of arrivals and use them to obtain a $1 + \epsilon$ approximate matching for the remainder, an approach once used by Yahoo! for ad matching. More broadly,

[9] In the case that multiple perfect dual variables exist, we take μ^* to be the perfect dual variable that is closest to $\hat{\mu}$.

[10] As a technical aside, there is a natural choice of κ: Proposition 2 in [22] implies that it is enough to only consider dual variables that lie in the m-dimensional rectangle $[0, \mu_1^{\max}] \times \cdots \times [0, \mu_m^{\max}] \in \mathbb{R}_+^m$ where $\mu_j^{\max} = \bar{r}/\rho_j + 1$. Therefore we may assume without loss of generality that the prediction $\hat{\mu}$ we receive lies inside this rectangle (otherwise we could project $\hat{\mu}$ onto this rectangle). Thus setting $\kappa = ||\mu^{\max}||_1$ ensures $a \in [0, \infty]$.

dual prediction is common in dual-based algorithms for online scheduling [106], online matching [59, 107], online covering [23], and ad auctions [98].

In these problems, including ours, the offline optimum depends only on the *dual* of the sample path, which lies in $\mathbb{R}_+^m$, rather than the full arrival sequence, whose dimension grows with the horizon T. As a result, predicting the dual variable is often easier than predicting the entire sample path.

By contrast, under stochastic arrivals, prior work assumes predictions of per-period arrival distributions [19, 88], measured via total variation and Wasserstein distances. Such predictions are independent of realized arrivals and thus encode less information about the sample path. Indeed, even perfect distributional predictions in the sense of [19, 88] can still lead to $\Omega(\sqrt{T})$ regret due to inherent variance. In contrast, we allow predicted dual variables to depend on future arrivals, potentially encoding richer information through statistical association, for example via observable intermediary variables.

Arrival Models and Performance Metrics

An online algorithm ALG, at each time period t, takes an action x_t (potentially randomized, but deterministic here to save on notation) based on the prediction $\hat{\mu}$, the current request $(r_t, g_t, \mathcal{X}_t)$ and the previous history $\mathcal{H}_{t-1} := \{r_s, g_s, \mathcal{X}_s, x_s\}_{s=1}^{t-1}$, i.e., $x_t = \text{ALG}\left(r_t, g_t, \mathcal{X}_t \mid \hat{\mu}, \mathcal{H}_{t-1}\right)$. We denote the reward received by an algorithm on an arrival sequence γ as

$$R(\text{ALG} \mid \gamma) = \sum_{t=1}^{T} r_t\left(x_t\right).$$

This notation is defined similarly to the notation $R(\text{GRD}_\mu \mid \gamma)$ which we defined for the Dual-Adjusted Greedy Algorithm. For the prediction $\hat{\mu}$, we use the *Prediction Algorithm* to represent the special case of the Dual-Adjusted Greedy Algorithm where the dual variable is $\hat{\mu}$, and we let $\text{PRD}(\gamma) = R(\text{GRD}_{\hat{\mu}} \mid \gamma)$. As stated in Proposition 1, if $a = \infty$, i.e., if $\hat{\mu} = \mu^*$, then $\text{PRD}(\gamma) + (\bar{g}/\underline{g} + 1)(m+1)\overline{r} \geq \text{OPT}(\gamma)$, which shows the Prediction Algorithm is essentially optimal if we have a perfect prediction.

Note that for any sequence of dual variables $\mu_1, \ldots, \mu_T$, following μ_t at time period t gives a series of actions $x_1, \ldots, x_t$. We define the *depletion time* of resource j by following $\mu_1, \ldots, \mu_T$ to be the first time period such that the remaining amount of resources j is less than $\underline{g}$, that is, after this time period no actions that consumes

resource j is feasible (if this never happens we set the depletion time to be T). We will use the depletion time to quantify the behavior of $\hat{\mu}$. Intuitively, dual variables close to $\hat{\mu}$ induce similar actions in most time periods as long as $\hat{\mu}$ is not always on the "boundary" of decisions, and hence their depletion time should be similar. We make this idea formal using the following assumption on the depletion time.

Assumption 2 (Non-degenerate Prediction)**.** *Fix an arrival sequence γ and a prediction $\hat{\mu}$ for γ, there exists a constant $\zeta > 0$ satisfying the following: for any sequence of dual variables $\mu_1, \ldots, \mu_T$ where $\mu_t \in \mathbb{R}_+^m$ and $||\hat{\mu} - \mu_t||_1 \leq \zeta$ for all t, the difference between the depletion time of resource j by following $\mu_1, \ldots, \mu_T$ and by following $\hat{\mu}$ is in $o(T)$ for every resource j.*

Assumption 2 roughly states that, for a sequence of dual variables that is close to the prediction, the action induced by the sequence of dual variables and the action induced by the prediction deplete resources at similar times. This assumption is reasonable and mild for the following reasons: in reality, most actions sets are discrete (such as {accept,reject}, $\mathbb{N}$, etc.). Therefore for most $\hat{\mu}$, as long as it is not at the "boundary" (which is often a measure-zero set), dual variables close to $\hat{\mu}$ all induce the same action. Moreover, in practice it is also unlikely for the "boundaries" at each time period to be the same across a majority of time periods since $r_t(\cdot)$ and $g_t(\cdot)$ vary over time, in which case Assumption 2 is satisfied with any prediction $\hat{\mu}$. Finally, perturbing each input with some small noise also turns a degenerate prediction into a non-degenerate one:

Proposition 2. *Let the action set of each time period be $\mathcal{X}_t = \{0, 1\}$. For any arrival sequence γ and any prediction $\hat{\mu} \neq 0$, and any arbitrarily small $\sigma > 0$, let γ' be an arrival sequence obtained by randomly perturbing the reward functions r_t to be r'_t in the following way: $r'_t(0) = r_t(0)$ and $r'_t(1) = r_t(1) + \epsilon_t$ where $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ are i.i.d. random variables. Then γ and $\hat{\mu}$ satisfies Assumption 2 almost surely (with respect to the random perturbation). In particular, for sufficiently small $\zeta > 0$, Assumption 2 is satisfied with probability $1 - \exp(-c\zeta T)$ for some constant $c > 0$.*

The proof of Proposition 2 appears in Section B.1. Proposition 2 formalizes the intuition that perturbing each input with some small noise may turn a degenerate prediction into a non-degenerate one. Although Proposition 2 only considers $\mathcal{X} = \{0, 1\}$ and perturbs the reward functions, similar results can be shown for more general $\mathcal{X}$ and perturbations on the resources consumption functions.

There are two primary arrival models when studying online problems: the *stochastic (i.i.d) arrival model* and the *adversarial arrival model*, both which we define formally below. Our goal is to design algorithms that have good performances under both arrival models, and for predictions of different qualities. Additionally, our algorithms should be *oblivious* to the arrival model and the prediction quality, i.e., they should have good performance without knowing the arrival model and the prediction quality.

Stochastic (i.i.d.) Arrival Model. The arrivals are drawn independently from an (unknown) underlying probability distribution $\mathcal{P} \in \Delta(\mathcal{S})$, where $\Delta(\mathcal{S})$ is the space of all probability distributions over $\mathcal{S}$. We measure the performance of an algorithm by its **regret**. Given an underlying arrival distribution $\mathcal{P}$, the regret incurred by an algorithm ALG under $\mathcal{P}$ is defined as $\mathbb{E}_{\gamma\sim\mathcal{P}^T}[\mathrm{OPT}(\gamma) - R(\mathrm{ALG} \mid \gamma)]$. We will be concerned with the **worst-case regret** over all distributions in $\Delta(\mathcal{S})$: we define the regret of ALG to be

$$\mathrm{Regret}(\mathrm{ALG}) = \sup_{\mathcal{P}\in\Delta(\mathcal{S})} \mathbb{E}_{\gamma\sim\mathcal{P}^T}[\mathrm{OPT}(\gamma) - R(\mathrm{ALG} \mid \gamma)].$$

Note that if the regret $\mathrm{Regret}(\mathrm{ALG})$ is sub-linear in T, then algorithm ALG is essentially optimal on average as T goes to infinity.

Since $\gamma \sim \mathcal{P}^T$ is a random object, we assume that the prediction $\hat{\mu}$ is defined on the same probability space as γ, and that

$$\|\hat{\mu} - \mu^*\|_1 = O(T^{1/2-a})$$

holds for every realization of γ.

To illustrate that this condition can hold for some $a > 0$, suppose the optimal dual variable admits the decomposition

$$\mu^* = \mu_1 + \Psi_1(\nu_1, \ldots, \nu_n) + \Psi_2(\omega_1, \ldots, \omega_n),$$

where $\mu_1 \in \mathbb{R}^m_+$ is a fixed quantity in the dual space, $\Psi_1, \Psi_2 : \mathbb{R}^n \to \mathbb{R}^m$ are fixed functions, and $\nu_1, \ldots, \nu_n$ and $\omega_1, \ldots, \omega_n$ are random variables induced by the randomness of the arrival sequence γ. Assume that the prediction $\hat{\mu}$ has access to $\nu_1, \ldots, \nu_n$, so that

$$\hat{\mu} = \mu_1 + \Psi_1(\nu_1, \ldots, \nu_n).$$

In this case, the prediction error $\|\hat{\mu} - \mu^*\|_1$ depends only on the randomness in $\omega_1, \ldots, \omega_n$, and not on $\nu_1, \ldots, \nu_n$. Consequently, if the variance of $\omega_1, \ldots, \omega_n$ is

small, we may have

$$\|\hat{\mu} - \mu^*\|_1 = O(T^{1/2-a})$$

for some $a > 0$, uniformly over all realizations of γ.

In practice, the prediction $\hat{\mu}$ is not assumed to depend causally on γ, but is expected to be statistically related to γ, for instance through shared observable co-variates.

Adversarial Arrival Model. The arrivals are arbitrary and adversarial, generated by an oblivious adversary, meaning that the arrival sequence is fixed prior to execution and does not depend on the algorithm's internal randomness or actions. Unlike the stochastic arrival model, regret here can be shown to grow linearly with T for any algorithm, so it is less meaningful to study the order of regret over T. Instead, we use **competitive ratio** as the performance metric. We say that an algorithm ALG is asymptotically ***α*-competitive** if $\alpha \geq 1$ is the smallest number such that

$$\limsup_{T\to\infty} \sup_{\gamma\in\mathcal{S}^T} \left\{ \frac{1}{T} \left(\frac{1}{\alpha} \mathrm{OPT}(\gamma) - R(\mathrm{ALG} \mid \gamma) \right) \right\} \leq 0.^{11}$$

In words, if an algorithm is asymptotically α-competitive, then it can obtain at least $1/\alpha$ fraction of the optimal reward in hindsight as T goes to infinity.[12]

[21] proved that, without predictions, the lowest competitive ratio that any algorithm can achieve is $\alpha^* = \max\{\sup_{(r,g,\mathcal{X})\in\mathcal{S}} \sup_{j\in[m],x\in\mathcal{X}} g(x)_j/\rho_j, 1\}$. [22] gave a mirror descent algorithm that achieves this competitive ratio. This is, loosely speaking, the best we might hope to achieve with "bad" predictions. On the other hand, we can always obtain $\mathrm{PRD}(\gamma)$ by following the prediction, which may exceed $\mathrm{OPT}(\gamma)/\alpha^*$ with "good" predictions (indeed, as we have seen in Proposition 1, $\mathrm{PRD}(\gamma)$ can be as large as $\mathrm{OPT}(\gamma)$).

If we knew the prediction quality beforehand, we could obtain the maximum of the two by simply choosing the better approach (this is in fact the best we can hope for). Using this as the benchmark, we will compare an algorithm's reward to this maximum. That is, for an algorithm ALG, we will analyze the following quantity:

$$\limsup_{T\to\infty} \sup_{\gamma\in\mathcal{S}^T} \left\{ \frac{1}{T} \left(\max\left\{ \frac{1}{\alpha^*} \mathrm{OPT}(\gamma), \mathrm{PRD}(\gamma) \right\} - R(\mathrm{ALG} \mid \gamma) \right) \right\}.$$

[11] This definition serves as an implicit assumption that $\mathrm{OPT}(\gamma)$ grows linearly in T. When $\mathrm{OPT}(\gamma)$ is sub-linear in T, by our definition every algorithm is 1-competitive.

[12] We assume the arrival sequence $\gamma = (\gamma_1, \ldots, \gamma_T)$ is fixed in advance. Our results still hold if the arrival sequence is chosen by a non-oblivious or adaptive adversary who does not know the internal randomization of the algorithm.

A Measure of Stationarity

Ideally, one would hope to develop an algorithm that achieves the "best" performance under both stochastic and adversarial arrivals respectively without knowing the prediction quality and the underlying arrival model. However, we will show in the next section that this is provably not achievable by any algorithm.

For an arrival sequence γ, its *stationarity* is closely related to the "difficulty" of solving the instance it created. As examples, an arrival sequence generated independently from the same underlying distribution can be considered as completely stationary, an arrival sequence that has certain seasonality/periodicity with small trend (e.g. generated from time series models) is less stationary, and an arrival sequence that is adversarially chosen (e.g. the lower bound instance) is completely nonstationary. Intuitively, an arrival sequence is more stationary if certain parts of the sequence with the same length are "similar" to each other. In this subsection we formalize this idea and develop a measure of arrival sequences' stationarity. We then use it to quantify algorithms' performances.

For a time interval from time periods s to time period t, let $\gamma_{s:t} = (\gamma_s, \ldots, \gamma_t)$ denote the arrival sequence from time period s to time period t. We define the $\gamma_{s:t}$-*subproblem* to be the problem instance where the arrival sequence is $\gamma_{s:t}$ and the total amount of resources is $\rho(t - s + 1)$, i.e., scaled down proportionally. In particular, the (offline) optimum of the $\gamma_{s:t}$-*subproblem* is:

$$\mathrm{OPT}(\gamma_{s:t}) := \max_{x_{t'} \in \mathcal{X}_{t'}} \sum_{t'=s}^{t} r_{t'}(x_{t'}) \quad \text{s.t.} \quad \sum_{t'=s}^{t} g_{t'}(x_{t'}) \le \rho(t - s + 1).$$

Similarly, we use $R(\mathrm{GRD}_\mu \mid \gamma_{s:t})$ to denote the amount of reward obtained by the Dual-Adjusted Greedy Algorithm with dual variable μ on the $\gamma_{s:t}$-*subproblem*.

Definition 4 (Measure of Stationarity)**.** Given the total number of available resources ρT, an arrival sequence $\gamma = (\gamma_1, \ldots, \gamma_T)$ is (λ, δ)-*stationary* for some $0 < \delta \le 1$ and $0 \le \lambda \le 1$ if for every $\mu \in \mathbb{R}_+^m$:

$$\min_{k=1,\ldots,\lfloor \frac{1}{\delta} \rfloor} \left(R(\mathrm{GRD}_\mu \mid \gamma_{1:k\delta T}) + R(\mathrm{GRD}_\mu \mid \gamma_{k\delta T+1:T})\right) \ge (1 - \lambda) R(\mathrm{GRD}_\mu \mid \gamma).$$

Intuitively, γ being (λ, δ)-stationary (roughly) means when we break γ into two subproblems at any time period that is a multiple of δT, the rewards obtained by these two subproblems sum up to be at least $1 - \lambda$ portion of the reward obtained by γ.

A few remarks are in order:

- δ measures the number of possible partition time periods that makes the subproblems similar to the original problem. On the extremes, δ close to 0 means that every subproblem is similar to the original problem, and $\delta = 1$ imposes no restrictions on γ. As examples, if γ is generated i.i.d. from some underlying distribution, δ can be arbitrarily close to 0, and if γ is periodic with small period, δ can be small.

- λ measures the loss in the partition, which can be viewed as the similarity of the subproblems to the original problem. On the extremes, $\lambda = 0$ means γ can be partitioned at time periods $k\delta T$ without losing much rewards, and $\lambda = 1$ imposes no restrictions on γ. As an example, γ having small "trend" (i.e., the infinity-norm of the vector of possible rewards is similar across all time periods) implies small λ.

- Smaller δ and smaller λ both represent more stationarity. Note that there is no single fixed (λ, δ) pair for an arrival sequence, but rather each choice of δ gives a corresponding λ, and smaller δ yields larger λ, i.e. the subproblems become less similar as the partition becomes more granular. The role of δ and λ will become clear when we state our main theorem, and we will not need to know the value of λ in our algorithm.

- Unlike usual stochastic definitions of stationarity (such as in [88] and [116]), here it is defined for deterministic arrival sequences. We show in the proposition below that if the arrivals are stochastic (i.i.d.), then the arrival sequence is $(\delta, 0)$-stationary for any $\delta > 0$ with high probability. This shows our definition of stationarity is compatible with stochastic definitions of stationarity.

  Proposition 3. *If an arrival sequence γ is generated under the stochastic (i.i.d.) arrival model, then γ is (δ, λ)-stationary for any constants $\delta, \lambda > 0$ with probability at least $1 - O(T^{-2})$.*

  The proof of Proposition 3 appears in Appendix B.1.

- In most stochastic definitions of stationarity (such as in [88] and [116]), the nonstationary measure is meant to characterize the instances that are possible to achieve sub-linear regret. All instances that incur linear regret are viewed as completely nonstationary. In our paper, the nonstationary measure is meant to characterize the instances that are impossible to achieve sub-linear regret

(or equivalently, that are impossible to achieve competitive ratio that is 1). All instances that incurs sub-linear regret are viewed as completely stationary.

- Unlike usual definitions of stationarity, our definition is problem- (i.e. resource-) dependent. For example, $\rho = 0$ and ρ sufficiently large both imply δ can be arbitrarily small and $\lambda = 0$, since any partition of the arrival sequence gives the same reward.

3.3 Main Results

In this section, we first present previous results for the Online Resource Allocation problem *without* predictions, and then give our main results on the full problem (*with* predictions) along with matching lower bounds.

Prior Results: Online Resource Allocation without Predictions

[22] studied the no-prediction version of our problem and gave a mirror descent algorithm which achieves the "best" achievable performance under both arrival models without knowing the underlying arrival model. We discuss their algorithm in detail in Appendix B.2. They proved the following performance guarantee for their *Mirror Descent Algorithm* (MDA):

Proposition 4 (Theorem 1 and Theorem 2 in [22])**.** *Consider the Mirror Descent Algorithm* (MDA)*. It holds that:*

(1) If the arrivals are stochastic,

$$\text{Regret}(\text{MDA}) = O(T^{\frac{1}{2}});$$

(2) If the arrivals are adversarial,

$$\limsup_{T\to\infty} \sup_{\gamma\in\mathcal{S}^T} \left\{ \frac{1}{T} \left(\frac{1}{\alpha^*} \text{OPT}(\gamma) - R(\text{MDA} \mid \gamma) \right) \right\} \leq 0.$$

Proposition 4 shows that the Mirror Descent Algorithm achieves $O(T^{\frac{1}{2}})$ regret and is α^*-competitive, which are both optimal [10, 21].

Prior Results: Lower Bounds

As a final step before describing our results, we present previous lower bounds for the full problem with predictions and *known* arrival model.

Stochastic Arrivals. Without predictions, the best achievable regret by any algorithm is $O(T^{\frac{1}{2}})$ [10]. With predictions, [141] gave the following lower bound on the best achievable regret with known accuracy parameter a:

Proposition 5 (Corollary of Theorem 2 in [141])**.** *Under stochastic arrival model, given a prediction $\hat{\mu}$ with accuracy parameter a, no algorithm can achieve regret better than $O(\max\{T^{\frac{1}{2}-a}, 1\})$, even if a is known.*

Adversarial Arrivals. Without predictions, the best achievable reward by any algorithm (taken the worst-case γ across all problem instances) is $\frac{1}{\alpha^*}\mathrm{OPT}(\gamma)$ [21]. On the other hand, simply following the actions induced by the prediction at each time yields reward $\mathrm{PRD}(\gamma)$. As we have seen in Proposition 1, for good predictions $\mathrm{PRD}(\gamma)$ can be as high as $\mathrm{OPT}(\gamma)$. Hence we have the following lower bound under adversarial arrivals:

Proposition 6 (Corollary of Theorem 3.1 in [21])**.** *Under adversarial arrival model, given a prediction with accuracy parameter a, no algorithm can achieve (worst-case γ across all problem instances) reward higher than $\max\{\frac{1}{\alpha^*}\mathrm{OPT}(\gamma), \mathrm{PRD}(\gamma)\}$, even if a is known.*

Our Results: Online Resource Allocation with Predictions

We are finally prepared to state our main result, which to develop a single algorithm that achieves the optimal performance using predictions, without knowing the underlying arrival model and the prediction accuracy.

Theorem 2 (Upper Bound)**.** *Assume that Assumptions 1 and 2 hold. Given a prediction $\hat{\mu}$ with (unknown) accuracy parameter a and given $0 < \delta \leq 1$, there exists an algorithm* (MainALG) *such that:*

(1) If the arrivals are stochastic,

$$\mathrm{Regret}(\mathrm{MainALG}) = \tilde{O}(\max\{T^{\frac{1}{2}-a}, 1\});$$

(2) If the arrivals are adversarial and (λ, δ)-stationary,

$$\limsup_{T\to\infty} \sup_{\gamma\in\mathcal{S}^T} \left\{ \frac{1}{T}\Big((1-\lambda)\max\left\{ \frac{1}{\alpha^*}\mathrm{OPT}(\gamma),\ \mathrm{PRD}(\gamma) \right\} - R(\mathrm{MainALG} \mid \gamma) \Big) \right\} \leq \delta\bar{r}.$$

A few remarks are in order:

- The performance guarantee under adversarial arrivals is better for smaller λ and δ, which matches the intuition that one can hope to achieve better performance with more stationary arrivals.
- If the arrivals are known to be adversarial, which we will discuss in the next section (Algorithm 2 and Proposition 9), there exists an algorithm that achieves

$$\limsup_{T\to\infty} \sup_{\gamma\in\mathcal{S}^T} \left\{\frac{1}{T}\left(\max\left\{\frac{1}{\alpha^*}\mathrm{OPT}(\gamma), \mathrm{PRD}(\gamma)\right\} - R(\mathrm{ALG} \mid \gamma)\right)\right\} \le 0$$

  That is, we are able to not suffer from nonstationarity. This is because the performance requirement is much higher for stochastic arrivals (sub-linear regret), which requires a conservative consumption of resources and hence obtains less rewards when the arrivals are highly nonstationary. This idea is elaborated in the lower bound construction below.

We provide a lower bound which shows Theorem 1 is tight in the sense that for any algorithm that achieves sub-linear regret under stochastic arrivals, one cannot replace λ with any number smaller and still get meaningful guarantees under adversarial arrivals. The proof of Proposition 7 appears in Appendix B.3. The lower bound construction consists of two instances, stochastic with bad prediction and adversarial with good prediction, that are provably indistinguishable for a certain period of time.

Proposition 7 (Lower Bound)**.** *For any $0 \le \lambda' < \lambda \le 1$, $0 < \delta \le 1$, and $K > 1$, there exists a sequence of instances γ of increasing time horizon T that satisfies Assumptions 1 and 2, such that for any algorithm* (ALG)*, at least one of the following holds:*

(1) The arrivals are stochastic, and

$$\mathrm{Regret}(\mathrm{ALG}) = \Omega(T);$$

(2) The arrivals are adversarial and (λ, δ)-stationary, and

$$\limsup_{T\to\infty} \left\{\frac{1}{T}\left((1-\lambda')\max\left\{\frac{1}{\alpha^*}\mathrm{OPT}(\gamma), \mathrm{PRD}(\gamma)\right\} - R(\mathrm{ALG} \mid \gamma)\right)\right\} > K\delta\bar{r}.$$

3.4 Algorithm and Proof of Main Result

In this section, we first present two algorithms that utilize the prediction in an optimal way for the two arrival models respectively. Then we combine these two algorithms to a single algorithm that is oblivious to both the prediction quality and the arrival model, which completely solves the Online Resource Allocation with Prediction problem.

Our algorithms for each arrival model utilize mirror descent, which take an initial dual variable, a step-size, and a reference function[13] as inputs. At each time period t, the algorithms take the action induced by the current dual variable μ_t, and performs a first-order update on the dual variable. With prediction $\hat{\mu}$, a natural initialization of the dual variable is to set $\mu_1 = \hat{\mu}$, i.e., the algorithms start by assuming the prediction is accurate. Then, the algorithms use adaptive step sizes η_t in mirror descent steps depending on the arrival model and the prediction's behavior.

Algorithm for the Stochastic Arrival Model

Let $\hat{\mu}$ be a prediction with accuracy parameter a, i.e., $||\hat{\mu} - \mu^*||_1 \leq \kappa T^{-a}$. By Proposition 5, no algorithm can achieve regret better than $O(\max\{T^{\frac{1}{2}-a}, 1\})$ even if a is known. As a comparison, we can show that the optimal fixed step size for the Mirror Descent Algorithm is $\eta \sim T^{-\frac{1-a}{2}}$ using similar method as the proof of Proposition 4, which incurs $O(\max\{T^{\frac{1-a}{2}}, 1\})$ regret. Therefore, mirror descent with fixed step size is sub-optimal even if the prediction quality is known. This suggest us to use adaptive step sizes. The step size we use is drawn from [43] in their work in parameter-free optimization. [14]

Algorithm 18, which we call the *Stochastic Arrival Algorithm* (SA), initializes the dual variable at the prediction $\hat{\mu}$ and updates the dual variable at each time period through mirror descent with fine-tuned step sizes. A detailed description of Algorithm 18 appears in Appendix B.4.

Proposition 8. *Consider the Stochastic Arrival Algorithm* (SA) *under the stochastic arrival model. Given a prediction $\hat{\mu}$ with (unknown) accuracy parameter a, it holds that:*

$$\text{Regret(SA)} = \tilde{O}(\max\{T^{\frac{1}{2}-a}, 1\}).$$

[13] For completeness, in Appendix B.2 we state the standard assumptions on choosing the reference function $h(\cdot)$ for mirror descent algorithms [30, 39, 119, 120].

[14] It follows the line of work in the more general online learning problem of parameter-free regret minimization [45, 54–56, 132, 158].

The proof of Proposition 8 can be found in Appendix B.5. By Proposition 5, the Stochastic Arrival Algorithm achieves optimal worst-cast regret up to logarithm factors.

Algorithm for the Adversarial Arrival Model

Different from the stochastic arrival model, under the adversarial arrival model it is impossible to achieve sub-linear worst-case regret. Instead, we directly compare the reward obtained by our algorithm to the maximum reward of two natural benchmark algorithms: the Mirror Descent Algorithm (which is optimal when the prediction quality is low) and the Prediction Algorithm (which is optimal when the prediction quality is high). That is, for an algorithm ALG, we will analyze the following quantity:

$$\limsup_{T\to\infty} \sup_{\gamma\in\mathcal{S}^T} \left\{ \frac{1}{T} \left(\max\left\{ \frac{1}{\alpha^*}\mathrm{OPT}(\gamma), \mathrm{PRD}(\gamma) \right\} - R(\mathrm{ALG} \mid \gamma) \right) \right\}.$$

We give Algorithm 2 for the adversarial arrival model, which we call the *Adversarial Arrival Algorithm* (AA). It performs mirror descent with fixed step size $\eta \sim \epsilon(T)/T$.

Algorithm 2: Adversarial Arrival Algorithm (AA)

Inputs: Prediction $\hat{\mu}$, total time periods T, initial resources $G_1 = \rho T$, reference function $h(\cdot) : \mathbb{R}^m \to \mathbb{R}$, upper bound function $\epsilon(T) = o(1)$, and step-size $\eta \sim \epsilon(T)/T$

Initialize $\mu_1 \leftarrow \hat{\mu}$

for *t from* 1 *to* T **do**

Receive request $(r_t, g_t, \mathcal{X}_t)$

Make the primal decision x_t and update the remaining resources G_t:

$$x_t \in \arg\max_{x\in\mathcal{X}_t, g_t(x)\le G_t} \left\{ r_t(x) - \mu_t^\top g_t(x) \right\}$$
$$G_{t+1} \leftarrow G_t - g_t(x_t).$$

Obtain a sub-gradient of the dual function:

$$\phi_t \leftarrow -g_t(x_t) + \rho.$$

Update the dual variable by mirror descent:

$$\mu_{t+1} \leftarrow \arg\min_{\mu\in\mathbb{R}_+^m} \phi_t^\top \mu + \frac{1}{\eta} V_h(\mu, \mu_t),$$

where $V_h(x, y) := h(x) - h(y) - \nabla h(y)^\top (x - y)$ is the Bregman divergence.

end

Proposition 9. *Assume that Assumptions 1 and 2 hold. Consider the Adversarial Arrival Algorithm* (AA) *under the adversarial arrival model. Given a prediction $\hat{\mu}$ with (unknown) accuracy parameter a, it holds that:*

$$\limsup_{T\to\infty} \sup_{\gamma\in\mathcal{S}^T} \left\{ \frac{1}{T} \left(\max\left\{ \frac{1}{\alpha^*}\mathrm{OPT}(\gamma), \mathrm{PRD}(\gamma) \right\} - R(\mathrm{AA} \mid \gamma) \right) \right\} \le 0.$$

The proof of Proposition 9 can be found in Appendix B.5. Proposition 9 states that the Adversarial Arrival Algorithm achieves the maximum of the two benchmark algorithms without knowing the prediction quality. Proposition 9 is tight by Proposition 6.

Main Algorithm: Detection of Nonstationarity

With the Stochastic Arrival Algorithm and the Adversarial Arrival Algorithm, we are ready to present our main algorithm, which merges the two algorithms and works for both arrival models.

Algorithm 3: Main Algorithm

Inputs: Prediction $\hat{\mu}$, total time periods T, initial rewards $R_1 = 0$, initial resources $G_1 = 0$, reference function $h(\cdot) : \mathbb{R}^m \to \mathbb{R}$, upper bound function $\epsilon(T) = o(1)$, constant L which is specified in Equation (B.30) in Appendix B.6, constant $0 < \delta \leq 1$, and initial step-size η_1

for *t from* 1 *to* T **do**
- Receive request $\gamma_t = (r_t, g_t, \mathcal{X}_t)$
- **if** $t = k\lfloor \delta T \rfloor + 1$ *for some* $k = 0, \ldots, \lceil 1/\delta \rceil - 1$ **then**
  - **if** $R_t + L \log(T)\sqrt{T} \geq \mathrm{OPT}_{t-1}(\gamma_1, \ldots, \gamma_{t-1})$ **then**
    - Release resources for the next $\lfloor \delta T \rfloor$ time periods: $G_t \leftarrow G_t + \lfloor \delta T \rfloor \rho$
    - Take action x_t given by the Stochastic Arrival Algorithm with the following inputs: total time periods $\lfloor \delta T \rfloor$, initial dual variable $\mu_{k\lfloor \delta T \rfloor + 1} = \hat{\mu}$, initial resources $G_{k\lfloor \delta T \rfloor + 1}$, reference function $h(\cdot)$, and initial step-size η_1
    - Update resources: $G_{t+1} \leftarrow G_t - g_t(x_t)$
    - Update rewards: $R_{t+1} \leftarrow R_t + r_t(x_t)$.
  - **else**
    - **break**
  - **end**
- **end**
- Release all resources: $G_t \leftarrow G_t + \rho(T - t + 1)$
- Use the Adversarial Arrival Algorithm with initial dual solution $\hat{\mu}$, remaining resources G_t, and step size $\eta \sim \epsilon(T)/T$.

end

The Main Algorithm initially assumes stochastic arrivals and runs the Stochastic Arrival Algorithm, cautiously releasing resources to avoid over-consumption. Simultaneously, it monitors observed arrivals to test this assumption: under stochastic arrivals, the accumulated reward should be comparable to the optimal offline reward under the underlying distribution. If the observed reward is significantly lower, the

algorithm infers with high probability that arrivals are not stochastic and switches to the Adversarial Arrival Algorithm for the remaining periods. If arrivals are adversarial but sufficiently stationary, this test may not trigger; however, in that case the Stochastic Arrival Algorithm remains effective, and no switch is needed.

Theorem 1 (Upper Bound)**.** *Consider the Main Algorithm* (MainALG)*. Assume that Assumptions 1 and 2 hold. Given a prediction $\hat{\mu}$ with (unknown) accuracy parameter a and given $0 < \delta \leq 1$, it holds that:*

(1) If the arrivals are stochastic,

$$\text{Regret}(\text{MainALG}) = \tilde{O}(\max\{T^{\frac{1}{2}-a}, 1\});$$

(2) If the arrivals are adversarial and (λ, δ)-stationary,

$$\limsup_{T\to\infty} \sup_{\gamma\in\mathcal{S}^T} \left\{ \frac{1}{T} \left((1-\lambda) \max\left\{ \frac{1}{\alpha^*}\text{OPT}(\gamma), \text{PRD}(\gamma) \right\} - R(\text{MainALG} \mid \gamma) \right) \right\} \leq \delta \bar{r}.$$

The proof of Theorem 1 appears in Appendix B.6. Theorem 1 is tight by Proposition 7.

3.5 Experiments

Finally, we evaluate our algorithm empirically using both synthetic and real data. The main takeaway is that our algorithm is robust to prediction quality: its reward is consistently close to the better of the Mirror Descent Algorithm (which is worst-case optimal without predictions) and the Prediction Algorithm.

In both experiments, we consider sequential assortment optimization, modeling an online retailer making in-cart recommendations. Each arrival corresponds to a customer checkout, where the algorithm recommends a subset of products of fixed cardinality. Each recommended product is purchased with a customer-specific probability, generating revenue. In the Online Resource Allocation with Predictions framework, actions correspond to assortments, resources to product inventories, and rewards to expected profits.

Each experimental instance consists of fixed initial inventories, an arrival sequence, and a prediction of the shadow price for each product, with prediction quality varying across instances. For each instance, we compare three total rewards: those obtained by our Main Algorithm, the Mirror Descent Algorithm (MDA), and the Prediction Algorithm (PRD).

Our primary performance metric is the *optimality gap*. For an instance I, let $R(\text{PRD} \mid I)$, $R(\text{MDA} \mid I)$, and $R(\text{MainALG} \mid I)$ denote the rewards of PRD, MDA,

and the Main Algorithm, respectively. We define

$$\text{GAP}(I) = \frac{R(\text{MainALG} \mid I) - \min\{R(\text{PRD} \mid I), R(\text{MDA} \mid I)\}}{\max\{R(\text{PRD} \mid I), R(\text{MDA} \mid I)\} - \min\{R(\text{PRD} \mid I), R(\text{MDA} \mid I)\}}.$$

This metric measures how close the Main Algorithm is to the better benchmark, normalized so that GAP = 1 corresponds to matching the better benchmark and GAP = 0 to matching the worse one (a random choice between PRD and MDA yields GAP = 0.5).[15]

Synthetic Experiment

We began with a set of smaller, synthetic experiments with 25 products over 1000 time periods, and the task of recommending 2 products at a time. We assumed customers belong to one of 25 "types." The process we used to randomly generate the product prices, the initial inventory levels, and the customer type-specific purchase probabilities, is described in Appendix B.7. Each time period corresponds to a single arriving customer (drawn uniformly from the 25 types), or no arrival. We used three types of arrival sequences, with the probability of a customer arrival changing over time: **stationary** with a fixed arrival probability of 0.7, **nonstationary** with an arrival probability linearly increasing from 0.4 to 1.0, and **adversarial** with an arrival probability of 0.0 during the first 300 periods and 1.0 afterward. We randomly generated predictions of varying qualities by computing the optimal shadow prices, and adding mean-zero Gaussian noise with standard deviations of 500 (**bad**), 5 (**good**), and 0 (**perfect**).

The results are summarized in the top half of Table 3.2, which for nine random ensembles of instances (depending on arrival model and prediction quality) reports both the median GAP, and the proportion of instances for which the GAP was at least 0.5 (for both values, higher is better). Recall that no algorithm can be expected to achieve high GAP values (say above 0.5) for all nine ensembles simultaneously. We see that our algorithm generally performs better with higher prediction quality and higher stationarity. Now one issue with GAP as a performance metric is that it can be quite erratic when the Mirror Descent and Prediction algorithms generate similar rewards (as their difference is the denominator in GAP), and these are arguably the instances in which GAP "matters" the least from a practical standpoint. Therefore, in the bottom half of Table 3.2 we removed the instances for which the two rewards are within 25% of each other. The remaining instances are exactly the instances

[15] Technically, GAP may lie outside $[0, 1]$.

Median	1 − CDF(0.5)	Stochastic		Nonstationary		Adversarial	
	Perfect Predictions	0.81	0.63	0.83	0.64	0.65	0.56
	Good Predictions	0.77	0.62	0.81	0.64	0.64	0.56
	Bad Predictions	0.54	0.52	0.45	0.49	0.22	0.40

Median	1 − CDF(0.5)	Stochastic		Nonstationary		Adversarial	
	Perfect Predictions	0.71	0.65	0.72	0.66	0.64	0.60
	Good Predictions	0.67	0.63	0.72	0.67	0.52	0.54
	Bad Predictions	0.58	0.55	0.49	0.49	0.36	0.40

Table 3.2: Summary of synthetic experiments. For each of three levels of prediction quality (the rows), and each of three generative arrival models (the columns), two summary statistics are reported over a random ensemble of instances: (left) the median GAP, and (right) the proportion of instances for which the GAP was at least 0.5. (Top) Results over all instances. (Bottom) Results over instances for which the rewards of the Mirror Descent and Prediction algorithms differ by at least 25%.

where robustness matters the most. We see that our algorithm is more robust to bad predictions on these instances.

H&M Experiment

We used a dataset from H&M (a fast-fashion clothing retailer), which contains the online transactions of 105,542 products from 2018 to 2020, along with product features. Because most products have zero or few transactions in two years, we selected the products with the top 5000 number of total transactions for our experiment, which includes 13,697,790 transactions. Our task was to recommend three products.

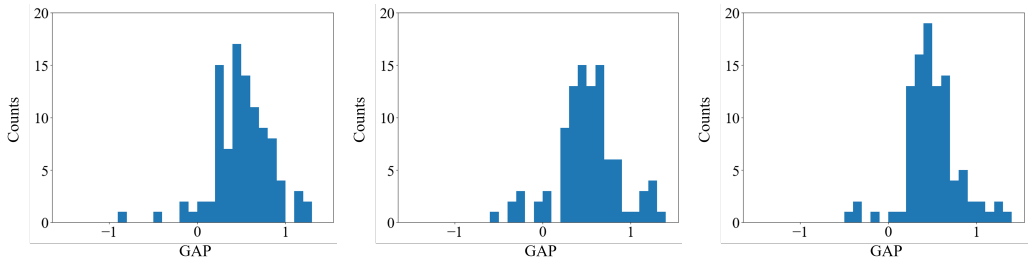
Each instance runs across three month's transactions from the data. The time horizon for each instance was the maximum number of transactions per day (103,473) multiplied by the total number of days (90), so that each day contained 103,473 time periods, each having zero or one arriving customer. To estimate customer-specific purchase probabilities, we used (customer, transaction time, product A, product B, price of product A, price of product B)-tuples and trained a random forest algorithm with the corresponding features of this tuple (a 209-dimensional vector after encoding) to estimate the probability that the customer, who brought product A at that certain time period with the certain price, would also buy product B if recommended.

To generate predictions, we used three popular forecasting methods ranging from classical algorithms to the state-of-the art:

- **Prophet:** A recent algorithm developed by Facebook [162] based on a (piecewise-linear) trend and seasonality decomposition, known to work well in practice with minimal tuning. Tuning parameters: software default.
- **Exponential Smoothing (Holt Winters):** A classic algorithm based on a (linear) trend and seasonality decomposition, known for its simplicity and robust performance. It is frequently used as a benchmark in forecasting competitions [125]. Tuning parameters: seasonality of length.
- **ARIMA:** Another classic algorithm that is rich enough to model a wide class of nonstationary time-series. Tuning parameters: (p, q, r).

These experiments were run on an N2D Series machine on Google Cloud's Compute Engine, with 224 vCPUs and 896GBs of memory. The total computation time was around 140 hours.

The results are summarized in Figure 3.3, which contains histograms of the GAPs across an ensemble of 100 instances (for varying three-month periods in the data), separately for each forecasting algorithm. The average GAP is 0.68 on instances with Prophet forecasts, 0.58 on instances with ARIMA forecasts, and 0.53 on instances with Exponential Smoothing forecasts. Because the average GAPs are large with all three forecasting methods, our algorithm performs close to the better one of the Prediction algorithm and the Mirror Descent Algorithm, showcasing its robustness to the unknown prediction accuracy.



(a) GAPs of Prophet (b) GAPs of ARIMA (c) GAPs of Exponential Smoothing

Figure 3.3: Histograms of GAPs with different forecasting methods, each containing 100 instances.

3.6 Conclusion

In this paper, we propose a new model for incorporating predictions into the Online Resource Allocation Problem. We first develop separate algorithms for stochastic and adversarial arrival models: under stochastic arrivals, the algorithm achieves nearly

optimal minimax worst-case regret, while under adversarial arrivals, it achieves a nearly optimal reward. Both algorithms operate without prior knowledge of prediction quality. Building on these, we propose a unified algorithm that attains the same guarantees under the appropriate arrival model, without knowing either the arrival model or the prediction quality in advance. The key idea is to initially treat arrivals as stochastic while continuously performing hypothesis tests to detect adversarial behavior.

Chapter 4

THE NONSTATIONARY NEWSVENDOR WITH (AND WITHOUT) PREDICTIONS

The classic newsvendor model yields an optimal decision for a "newsvendor" selecting a quantity of inventory, under the assumption that the demand is drawn from a known distribution. Motivated by applications such as cloud provisioning and staffing, we consider a setting in which newsvendor-type decisions must be made sequentially, in the face of demand drawn from a stochastic process that is both unknown and nonstationary. All prior work on this problem either (a) assumes that the level of nonstationarity is known, or (b) imposes additional statistical assumptions that enable accurate *predictions* of the unknown demand. Our research tackles the Nonstationary Newsvendor without these assumptions, both with and without predictions.

We first, in the setting without predictions, design a policy which we prove (via matching upper and lower bounds) achieves order-optimal regret – ours is the first policy to accomplish this without being given the level of nonstationarity of the underlying demand. We then, for the first time, introduce a model for generic (i.e. with no statistical assumptions) predictions with arbitrary accuracy, and propose a policy that incorporates these predictions without being given their accuracy. We upper bound the regret of this policy, and show that it matches the best achievable regret had the accuracy of the predictions been known.

Our findings provide valuable insights on inventory management. Managers can make more informed and effective decisions in dynamic environments, reducing costs and enhancing service levels despite uncertain demand patterns. This study advances understanding of sequential decision-making under uncertainty, offering robust methodologies for practical applications with nonstationary demand. We empirically validate our new policy with experiments based on three real-world datasets containing thousands of time-series, showing that it succeeds in closing approximately 74% of the gap between the best approaches based on nonstationarity and predictions alone.

4.1 Introduction

The newsvendor problem is a century-old model [65] that remains fundamental to the practice of operations management. In its original instantiation, a "newsvendor" is tasked with selecting a quantity of inventory before observing the demand for that inventory, with the demand itself randomly drawn from a *known* distribution. The newsvendor incurs a per-unit underage cost for unmet demand, and a per-unit overage cost for unsold inventory. The objective is to minimize the total expected cost, and the classic result is that the optimal inventory level is a certain problem-specific quantile (depending only on the underage and overage costs) of the demand distribution.

This paper is concerned with a modern instantiation of the same model, consisting of a *sequence* of newsvendor problems over time, each with *unknown* demand distributions that *vary* over time. While this version of the problem is arguably ubiquitous in practice today, it may be worth highlighting a few motivating examples:

- **Cloud Provisioning:** Consider a website which provisions computational resources from a commercial cloud provider to serve its web requests. Such provisioning is typically done *dynamically*, say on an hourly basis, with the aim of satisfying incoming requests at a sufficiently high service level. Thus, the website faces a single newsvendor problem every hour, with an hourly "demand" that can (and does) vary drastically over time.
- **Staffing:** A more traditional example is staffing, say for a brick-and-mortar retailer, a call center, or an emergency room. Each day (or even each shift) requires a separate newsvendor problem to be solved, with demand that is highly nonstationary.

Despite its ubiquity, this problem is far from resolved, precisely because the demand (or sequence of demand distributions) is both *nonstationary* and *unknown* – indeed, the repeated newsvendor with stationary, but unknown demand was solved by [111], and the same setting with known, but nonstationary demand can be treated simply as a sequence of completely separate newsvendor problems. At present, there are by and large two existing approaches to this problem:

1. **Limited Nonstationarity:** One approach is to design policies which "succeed" under limited nonstationarity, i.e. the cost incurred by the policy should be parameterized by some carefully-chosen measure of nonstationarity (e.g. quadratic variation), and nothing else. This approach has proved fruitful

across a diverse set of problems ranging from dynamic pricing [95] to multi-armed bandit problems [36] to stochastic optimization [37]. Most relevant here, the recent work of [96] applies this lens to the newsvendor setting (we will discuss this work in detail momentarily). This approach yields policies with theoretical guarantees that are quite robust – no assumption on the demand (beyond the limited nonstationarity) is required. However, this is far removed from practice, where the next approach is more common.

2. **Predictions:** The second approach is to utilize some sort of *predictions* of the unknown demand. These predictions can be generated from simple forecasting algorithms for univariate time-series, all the way to state-of-the-art machine learning models that leverage multiple time-series and additional feature information. Therefore these predictions may contain much more information than past demand data points, such as various features/contexts, or even black-box type information that is non-identifiable. In addition to being the de facto approach in practice, the use of predictions in newsvendor-type problems is well-studied, and in fact provable guarantees exist for many specific prediction-based approaches [24, 83, 143, 181]. All such guarantees rely on (at the very least) the demand and potential features being generated from a known family of stochastic models, so that the framework and tools of statistical learning theory can be applied. Absent these statistical assumptions, it is unclear a priori whether the resulting predictions will be sufficiently accurate to outperform robust policies such as those generated in the previous approach. As a concrete example of this, see Figure 4.1, which demonstrates on a real set of retail data that prediction accuracy may vary drastically and unexpectedly, even when those predictions are generated according to the same procedure and applied during the same time period.

To summarize, the repeated newsvendor with unknown, nonstationary demand (which from here on we refer to as the *Nonstationary Newsvendor*) admits policies with nontrivial guarantees, which can be made significantly better or worse by following predictions. This suggests the opportunity to design a policy that uses predictions *optimally*, in the sense that the predictions are utilized when accurate, and ignored when inaccurate. Ideally, such a policy would run without knowledge of (a) the accuracy of the predictions and (b) the method with which they are generated. *This is precisely what we accomplish in this paper.*

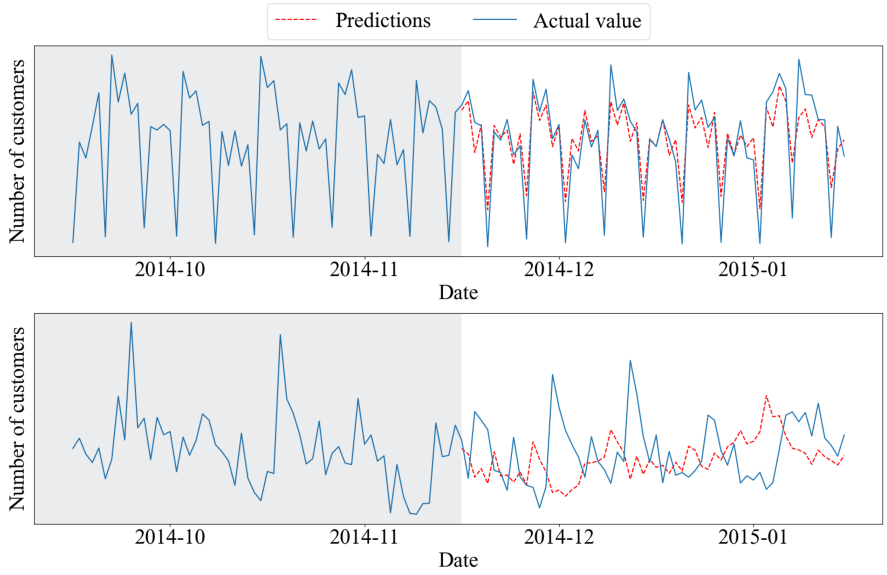


Figure 4.1: Daily number of customers (in blue), from September 2014 to January 2015, at two different stores in the Rossmann drug store chain. Predictions (in red), starting November 2014, are generated using Exponential Smoothing with the same fitting process. The store in the upper sub-figure has substantially more accurate predictions ($R^2 = 0.88$) than that of the lower sub-figure ($R^2 = 0.11$).

The Nonstationary Newsvendor, with and without Predictions

The primary purpose of this paper is to develop a policy that optimally incorporates *predictions* (defined in the most generic sense possible) into the *Nonstationary Newsvendor* problem. Naturally, a prerequisite to this is a fully-solved model of the Nonstationary Newsvendor without predictions. At present this prerequisite is only partially satisfied (via the work of [96]), so a nontrivial portion of our contributions will be to fully solve this problem.

Without predictions, the Nonstationary Newsvendor consists of a sequence of newsvendor problems indexed by periods $t \in 1, \ldots, T$, each with *unknown* demand distribution D_t. The level of nonstationarity is characterized via a *variation parameter* $v \in [0, 1]$, where $v = 0$ essentially amounts to stationary demand, and $v = 1$ is effectively arbitrary (in a little more detail: a deterministic analogue of quadratic variation is applied to the sequence of means $\{\mathbb{E}[D_1], \ldots, \mathbb{E}[D_T]\}$, and $v \in [0, 1]$ is the exponent such that this quantity equals T^v). Finally, we measure the performance of any policy using *regret*, which is the expected difference in the total cost incurred by the policy versus that of an optimal policy that "knows" the demand distributions. At minimum we aim to design a policy that achieves *sub-linear* (i.e. $o(T)$) regret, as such a policy would incur a per-period cost that is on average no worse than the

optimal, as T grows. We will in fact design policies which achieve *order-optimal* regret with respect to the variation parameter v.

To this base problem, we introduce the notion of predictions. In each period we receive a prediction a_t of the mean demand $\mu_t = \mathbb{E}[D_t]$ before selecting the order quantity. Our predictions are generic: no assumption is made on how they are generated. We measure the accuracy of the predictions through an *accuracy parameter* $a \in [0, 1]$, defined such that $\sum_{t=1}^{T} |a_t - \mu_t| = T^a$. Notice that when $a = 0$ the predictions are almost perfect, and when $a = 1$ the predictions are effectively useless. We will characterize a precise threshold on a (which depends on v) that determines when the predictions should be utilized. Our primary challenge will be to design a policy that makes use of the predictions only when they are sufficiently accurate, and *without* having access to a. As to the variation parameter v, we will separately consider policies which do and do not have access to v – this distinction will turn out to be *the* critical factor in classifying what is and is not achievable.

Our Contributions

Our primary contributions can be summarized as follows.

1. Nonstationary Newsvendor (without predictions): We completely solve the Nonstationary Newsvendor problem. This consists of first constructing a policy and proving an upper bound on its regret:

Theorem 2 (Informal)**.** *There exists a policy which achieves* $\tilde{O}(T^{(3+v)/4})$ *regret*[1] *without knowing* v.

We then show that this regret is minimax optimal up to logarithmic factors:

Proposition 10 (Informal)**.** *No policy can achieve regret better than* $O(T^{(3+v)/4})$*, even if* v *is known.*

As alluded to earlier, [96] previously initiated the study of the Nonstationary Newsvendor. Our results are distinct in terms of both modeling and theoretical contributions. We will expound these distinctions more carefully later on.

- **Modeling:** The most crucial difference in our model is that we allow both the demand and the set of possible ordering quantities to be *discrete*. This is

[1] The $\tilde{O}(\cdot)$ notation hides logarithmic factors.

certainly of practical concern (e.g. physical inventory, employees, and virtual machines are all indivisible units of demand), but moreover we will show that the results of [96] *require* both the demand and set of feasible ordering quantities to be continuous. Thus, there is no overlap in our theoretical results.

- **Results:** [96] succeed in designing a policy that achieves order-optimal regret, but crucially, their policy requires that the variation parameter v be known. In addition to being concerning from a practical standpoint, this leaves open the theoretical question of what exactly is achievable in settings for which v is unknown. Our results show that the *same* regret can be achieved without knowing v.

2. Nonstationary Newsvendor with Predictions: We construct a policy that *optimally* leverages predictions, i.e. it is robust to unknown prediction accuracy. To be precise, the previous contribution offers a policy that achieves $\tilde{O}(T^{(3+v)/4})$ regret, and predictions yield a simple policy that achieves $O(T^a)$ regret, so we would expect that the best possible regret is the minimum of these two quantities. We show this formally:

Proposition 11 (Informal)**.** *No policy can achieve regret better than* $O(T^{\min\{(3+v)/4,a\}})$*, even if* v *and* a *are known.*

Our main algorithmic contribution is a policy which achieves this lower bound (up to log factors) *without* knowing the prediction accuracy:

Theorem 3 (Informal)**.** *There exists a policy which achieves regret* $\tilde{O}(T^{\min\{(3+v)/4,a\}})$*, knowing* v*, and without knowing* a*.*

Finally, since our policy relies on knowledge of the variation parameter v, the remaining question is whether the same regret is achievable if both v and a are unknown. We show that in fact predictions *cannot* be incorporated in any meaningful way in this case:

Proposition 12 (Informal)**.** *If* v *and* a *are unknown, then no policy can achieve regret better than* $O(T^{\max\{(3+v)/4,a\}})$ *for all* $v, a \in [0, 1]$*.*

Our theoretical results are summarized in the Table 4.1. Each entry has a corresponding policy that achieves the stated regret, along with a matching lower bound.

	Without predictions	With predictions of unknown accuracy
Known variation	$\tilde{O}(T^{(3+v)/4})$	$\tilde{O}(T^{\min\{(3+v)/4,a\}})$
Unknown variation	$\tilde{O}(T^{(3+v)/4})$	$\tilde{O}(T^{\max\{(3+v)/4,a\}})$

Table 4.1: Summary of main theoretical results. Each entry has a corresponding policy that achieves the stated regret, along with a matching lower bound.

	No Prediction	**Prediction**	**Our Policy**
Upper store	\$28,303	\$14,454	\$14,454
Lower store	\$23,460	\$35,600	\$23,899

Table 4.2: Continuation of Figure 4.1: costs incurred by an optimal policy which makes no use of predictions, a policy which relies entirely on predictions, and our policy.

3. Empirical Results: Finally, we demonstrate the practical value of our model (namely the Nonstationary Newsvendor with Predictions) and our policy via empirical results on three real-world datasets that span our motivating applications above: daily web traffic for Wikipedia.com (of various languages), daily foot traffic across the Rossmann store chain, and daily visitors at a certain Japanese restaurant. These datasets together contain over one thousand individual time-series on which we generate predictions of varying quality, using four different popular forecasting and machine learning algorithms. We apply our policy, and compare its performance against the two most-natural baseline policies: our optimal policy without predictions, and the simple policy which always utilizes the predictions (these correspond to the two "existing approaches" described previously). A snapshot of our results, for the Rossmann stores depicted in Figure 4.1, is given in Table 4.2.

More generally, on any given experimental instance (i.e. a time-series and a set of predictions), the minimum (maximum) of the costs incurred by these two baselines can be viewed as the best (worst) we can hope for. Thus we measure performance in terms of the proportion of the gap between these two costs incurred by our policy, so if this "optimality gap" is close to 0, our policy performs almost as good as the better one of the two baselines. Note that *randomly* selecting between the two baseline policies yields an (expected) optimality gap of 0.5. We find that in the Rossmann dataset, the average optimality gap is 0.26 when the predictions are accurate, and 0.28 when the predictions are inaccurate. In the Wikipedia dataset, the average optimality gap is 0.40 when the predictions are accurate, and 0.07 when the

predictions are inaccurate. In the Restaurant dataset, the average optimality gap is 0.10 when the predictions are accurate, and 0.39 when the predictions are inaccurate. This demonstrates that our policy performs well, irrespective of the quality of the predictions.

Literature Review

The earliest works on the newsvendor model assume that the demand distribution is known [11, 154]. This has since been relaxed, the resulting approaches being divided into parametric and nonparametric ones. Among parametric approaches much work is Bayesian, where a prior distribution is assumed over the parameters. [153] applied the Bayesian approach to inventory models, and later this was studied in many works [15, 85, 91, 118]. [117] introduced another parametric approach called operational statistics which, unlike the Bayesian approach, does not assume any prior knowledge on the parameter values, instead using past demand observations to directly estimate the optimal ordering quantity.

Nonparametric approaches have been developed in recent years. The first example is the sample average approximation (SAA) method, first proposed by [100] and [157]. [110] applied SAA to the newsvendor problem, and [111] improved significantly upon their bounds. Other non-parametric approaches include stochastic gradient descent algorithms [41, 84, 104] and the concave adaptive value estimation (CAVE) method [71, 145]. With the development of machine learning, [24] and [143] propose machine learning/deep learning algorithms using demand features and historical data.

All the above studies treat the newsvendor in a static environment. There are two common approaches to nonstationarity. The first is to model (stochastically) the nonstationarity and utilize past demand observations according to the model. One common way is to model the nonstationarity as a Markov chain. For example, [163] applied this idea to inventory management, and [14, 46] applied this idea to revenue management. Another approach is to bound the nonstationarity via a variation budget, which has been applied to stochastic optimization [37], dynamic pricing [95], multi-armed bandit [36], newsvendor problem [96], among others. Some of these works are applicable in the sense that our problem can be mapped to their settings (e.g. multi-armed bandit such as [37, 52, 92, 122]), but these connections do not appear to be fruitful. In particular, the multi-armed bandit papers cited above typically consider a *limited-feedback* setting rather than the *full-feedback* setting

explored in this work. Related to feedback, while our study provides a complete characterization of the regret behavior for the nonstationary newsvendor problem with *uncensored* demand, practical applications often involve *censored* demand. The nonstationary newsvendor problem under censored demand is an interesting direction for future research.

Beyond the bandit literature, it is worth mentioning recent work on online convex optimization (OCO) with limited nonstationarity. When the level of nonstationarity is *known*, the standard first-order OCO algorithms can be modified with carefully chosen restarts and updating rules ([37],[178], [51]). There are also recent works that concern *unknown* nonstationarity, such as [180], [16], [17], and [82]. Finally, as mentioned before, [96] is particularly relevant, so we delay a careful comparison to Sections 4.2 and 4.3.

The second common practice is to use predictions. A recent line of work looks to help decision-making by incorporating predictions into online optimization problems such as revenue optimization [4, 20, 137], caching [123, 149], online scheduling [106], and the secretary problem [62]. In this paper we combine the nonstationarity framework and the prediction framework on the newsvendor problem.

Finally, most previous works involving algorithms with predictions analyzed algorithms' performances using competitive analysis (e.g. [9, 20, 89, 124]) and obtained optimal *consistency-robustness* trade-offs, where *consistency* is an algorithm's competitive ratio when the prediction is accurate, and *robustness* is the competitive ratio regardless of the prediction's accuracy. However, competitive ratio transfers to a regret bound that is linear in T. In contrast, we do regret analysis under this framework and design an algorithm that has near-optimal worst-case regret without knowing the prediction quality. Other papers with regret analyses under the prediction model include [137] (revenue optimization in auctions), [77] (Thompson sampling), [81] (constrained online two-stage stochastic optimization), and [4] (online resource allocation).

4.2 Model: The Nonstationary Newsvendor (without Predictions)

We begin this section with a formal description of the **Nonstationary Newsvendor**, along with a comparison to the problem of the same name from [96]. Consider a sequence of newsvendor problems over T time periods labeled $t = 1, \ldots, T$. At the beginning of each time period t, the decision-maker selects a quantity $q_t \in Q$, where

Q is a fixed subset of $\mathbb{R}^+$ bounded above by a quantity we denote as $Q_{\max}$.[2] Then the period's demand d_t is drawn from an (unknown) demand distribution D_t, which depends on the time period t. These demand distributions are independent over time. Finally a cost is incurred – specifically, there is a (known) per-unit underage cost $b_t \in [0, b_{\max}]$ and a (known) per-unit overage cost $h_t \in [0, h_{\max}]$, so that the total cost is equal to

$$b_t(d_t - q_t)^+ + h_t(q_t - d_t)^+,$$

where $x^+ = \max\{0, x\}$. The decision-maker observes the realized demand d_t,[3] and thus the cost. Note that requiring $q_t \in Q$ does not impose any restriction on *modeling*, since Q could simply be selected to be $\mathbb{R}$ (as in much of the literature). In fact, introducing Q allows for modeling important practical concerns such as batched inventory or even simply the integrality of physical items. As we will discuss momentarily, this is a non-trivial concern insofar as theoretical guarantees are concerned.

To complete our description of the Nonstationary Newsvendor, we will need to (a) impose a few assumptions on the demand distributions, and then (b) describe how "nonstationarity" is quantified. These are, respectively, the subjects of the following two subsections.

Demand Distributions

We will assume that the demand distributions come from a known, parameterized family of distributions $\mathcal{D}$:

Assumption 3. *Every demand distribution D_t comes from a family of distributions $\mathcal{D}$ satisfying the following:*

(a) $\mathcal{D} = \{\mathcal{D}_\mu : \mu \in [\mu_{\min}, \mu_{\max}]\}$, that is $\mathcal{D}$ is parameterized by a scalar μ taking values in some bounded interval.

(b) Each distribution $\mathcal{D}_\mu \in \mathcal{D}$ is sub-Gaussian.[4]

Assumption 3 is fairly minimal. Parsing it in reverse: the sub-Gaussianity in part (b) allows for many commonly-used variables, such as the Gaussian distribution

[2] All of our results carry through if Q is allowed to depend on t.

[3] The demand is not censored here, as is the case in all of the motivating examples in the introduction. The censored version of our problem is an interesting, but separate subject.

[4] A random variable X is *sub-Gaussian* with *sub-Gaussian norm* $\|X\|_{\psi_2}$ if $\mathbb{P}(|X| > x) \leq 2\exp\left(-x^2/\|X\|_{\psi_2}^2\right)$ for all $x \geq 0$. For sub-Gaussian variables, we have $\mathbb{E}[|X|] \leq 3\|X\|_{\psi_2}$.

and any bounded random variable, while letting us eventually apply Hoeffding-type concentration bounds. Part (a) is particularly minimal at the moment, as μ represents an arbitrary parameterization of $\mathcal{D}$, but will become meaningful when combined with Assumption 4. The choice of the symbol "μ" might suggest that μ represents the mean of $\mathcal{D}_\mu$, and indeed this is what we will assume from here on. But it should be emphasized that our taking $\mu = \mathbb{E}[\mathcal{D}_\mu]$ is strictly for notational convenience (because we will frequently need to refer to the means of these distributions): if μ were any other parameterization of $\mathcal{D}$, we could simply define a mapping from μ to the mean values.

Now define $C(\mu, b, h, q)$ to be the expected newsvendor cost when selecting quantity $q \in Q$, given underage/overage costs b and h, and demand distribution $\mathcal{D}_\mu$:

$$C(\mu, b, h, q) = \mathbb{E}_{d \sim \mathcal{D}_\mu}[b(d-q)^+ + h(q-d)^+].$$

The critical assumption, with respect to the parameterization in Assumption 3(a), is that the expected cost is well-behaved as a function of μ:

Assumption 4. *For every $b \in [0, b_{\max}]$, $h \in [0, h_{\max}]$, and $q \in Q$, the function $C(\cdot, b, h, q)$ is Lipschitz on its domain $[\mu_{\min}, \mu_{\max}]$, i.e. there exists $\ell \in \mathbb{R}^+$ such that for every $\mu_1, \mu_2 \in [\mu_{\min}, \mu_{\max}]$, we have*

$$|C(\mu_1, b, h, q) - C(\mu_2, b, h, q)| \leq \ell |\mu_1 - \mu_2|.$$

Note that in the above description, the Lipschitz constant ℓ may depend on b, h, and q, but by continuity, there exists a single ℓ so that the above holds for all b, h, q simultaneously.

Some useful examples of families $\mathcal{D}$ satisfying Assumptions 3 and 4 are the following:[5]

1. $\mathcal{D}_\mu \sim \mathcal{N}(\mu, \sigma^2)$, the family of normal distributions with fixed variance σ^2. In this case, $\ell = O(\sigma(b_{\max} + h_{\max}))$. A relaxation is that the variances may vary (continuously) with μ.
2. $\mathcal{D}_\mu = \mu + \epsilon$, where ϵ is any mean-zero, sub-Gaussian variable.

[5]Unfortunately, Assumption 4 is not guaranteed to hold. For example, for the family of distributions

$$\mathcal{D}_\mu \sim \begin{cases} \mu, & \mu \in [0,1] \\ \mu + \text{Bernoulli}(0.5) - 0.5, & \mu \in (1,2] \end{cases},$$

the function $C(\mu, 1, 1, 1)$ is discontinuous (and thus not Lipschitz) at $\mu = 1$.

3. The Poisson distribution is frequently used to model demand (since arrivals are often modeled as a Poisson process). While the Poisson distribution is *not* sub-Gaussian, any reasonable truncation satisfies our assumptions. For example, $\mathcal{D}_\mu \sim \min\{\text{Poisson}(\mu), K\mu\}$, for some constant K. Here, K can be taken to be large enough so that the truncation happens with small probability (in fact, this probability is $O(e^{-K\mu})$).

To understand the reasoning behind Assumption 4, consider the problem faced at some time t. The optimal choice for the decision-maker here is

$$q_t^* \in \text{argmin}_{q\in Q} C_t(\mu_t, q), \tag{4.1}$$

where μ_t is the mean of D_t (i.e. $D_t \sim \mathcal{D}_{\mu_t}$), and $C_t(\mu, q) = C(\mu, b_t, h_t, q)$ to simplify the notation.[6] Since D_t is unknown, it is likely that some $q_t \neq q_t^*$ will ultimately be selected, and we could measure the sub-optimality of this decision (i.e. regret, to be defined soon): $C_t(\mu_t, q_t) - C_t(\mu_t, q_t^*)$. It would be natural then to try to characterize this suboptimality as a function of $|q_t - q_t^*|$, but in fact all of the algorithms we will consider "work" by making an estimate $\hat{\mu}_t$ of μ_t, and then selecting $\hat{q}_t \in \text{argmin}_{q\in Q} C_t(\hat{\mu}_t, q)$. So motivated, the purpose of Assumption 4 is to allow us to "translate" error in our estimate of μ_t to (excess) costs. The following structural lemma makes this precise, and will be used throughout the paper.

Lemma 2. *Fix any b and h (we will suppress them from the notation). For any $D_{\mu_1}, D_{\mu_2} \in \mathcal{D}$, let $q_1^* \in \text{argmin}_{q\in Q} C(\mu_1, q)$ and $q_2^* \in \text{argmin}_{q\in Q} C(\mu_2, q)$. Then we have*

$$C(\mu_1, q_2^*) - C(\mu_1, q_1^*) \leq 2\ell|\mu_1 - \mu_2|.$$

Lemma 2 states that estimation error of the mean μ_t translates *linearly* to excess cost. The proof of Lemma 2 appears in Appendix C.1.

Aside: Comparison to [96]: The final component in describing the Nonstationary Newsvendor is defining a proper quantification of nonstationarity. Before doing so, we delineate the *modeling* differences between our Nonstationary Newsvendor and that of [96]. There are two primary differences:

1. The demand distributions D_t in [96] are assumed to be of the form $D_t = \mu_t + \epsilon_t$, where μ_t is the mean of D_t that drifts across time and ϵ_t is the noise distribution

[6] As a sanity check, the classical result for the newsvendor problem [11, 154] states that if $Q = \mathbb{R}$, then q_t^* is the $b_t/(b_t + h_t)$-th quantile of D_t.

that is i.i.d., continuous, and bounded. Effectively, the demand distributions fall into a *non-parametric* family of distributions with the same "shape". In contrast, our demand distributions fall into a *parametric* family of distributions, though not necessarily of the same "shape".

2. Our set of allowed order quantities Q is bounded, but otherwise arbitrary. In particular, it need not contain the optimal unconstrained order quantity $\operatorname{argmin}_{q\in\mathbb{R}} C(\mu, q)$ for each μ (or any μ, for that matter). [96] assume $Q = \mathbb{R}^+$.[7]

Besides the practical reasons why discrete quantities arise in practice (non-divisible items, batched inventory, etc.), the primary consequence of either of the two differences above is that they preclude a critical lemma used in [96] (and in fact by [110]) which states that $C(\mu_1, q_2^*) - C(\mu_1, q_1^*)$ (as defined in our Lemma 2) scales as $(q_1^* - q_2^*)^2$. This scaling does *not* necessarily hold when either the demand distribution or Q is discrete. These relaxations in assumptions yield different lower bounds in the worst-case regret from [96], which we will discuss in detail later.

Demand Variation

Just as in [96] (and [95] before that), we measure the level of nonstationarity via a deterministic analogue of quadratic variation for the sequence of means $\mu_1, \ldots, \mu_T$. Specifically, define a *partition* of the time horizon $\{1, \ldots, T\}$ to be any subset of time periods $\{t_0, \ldots, t_K\}$ where $1 \le t_0 < \cdots < t_K \le T$. Here the subset can have any size between 1 and T, i.e. $0 \le K \le T - 1$. Then for any sequence of means $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_T\}$, its **demand variation** is

$$V_{\boldsymbol{\mu}} = \max_{0\le K\le T-1} \max_{\{t_0,\ldots,t_K\}\in\mathcal{P}} \left\{ \sum_{k=1}^{K} \left|\mu_{t_k} - \mu_{t_{k-1}}\right|^2 \right\}, \tag{4.2}$$

where $\mathcal{P}$ is the set of all partitions.

To motivate the use of partitions in the definition of $V_{\boldsymbol{\mu}}$, it is worth contrasting with a measure that may feel more natural, namely the sum of squared differences (*SSD*) between consecutive terms, $\sum_{t=2}^{T} (\mu_t - \mu_{t-1})^2$, which corresponds to taking the densest possible partition $\{1, 2, \ldots, T\}$. The maximum in the definition of $V_{\boldsymbol{\mu}}$ is *not* necessarily achieved by selecting the densest possible partition, but rather by setting $t_0, \ldots, t_K$ to be the periods when the sequence $\mu_1, \ldots, \mu_T$ changes direction. Thus, the demand variation penalizes *trends*, or consecutive increases/decreases, more so than the SSD. For example, the mean sequences $\boldsymbol{\mu_1} = \{1, 2, 3, 4, 5\}$ and $\boldsymbol{\mu_2} = \{1, 0, 1, 0, 1\}$,

[7] While not stated explicitly, the results in [96] only require Q to contain points arbitrarily close to every optimal unconstrained order quantity.

respective variations $V_{\mu_1} = (5-1)^2 = 16$ and $V_{\mu_2} = 1^2 + 1^2 + 1^2 + 1^2 + 1^2 = 5$, despite having identical SSDs.

All of our theoretical guarantees (upper and lower bounds) will be parameterized by $V_{\boldsymbol{\mu}}$. This quantity of course depends on T, and so it is natural to allow $V_{\boldsymbol{\mu}}$ to grow T. It will turn out that the most natural parameterization of this growth is via what we will simply call the **variation parameter** $v \in [0, 1]$, such that $V_{\boldsymbol{\mu}} = BT^v$, where B is some constant (which we take to be equal to one from here on). We denote the set of demand distribution sequences $\{D_1, \dots D_T\}$ whose means $\boldsymbol{\mu} = \{\mu_1, \dots, \mu_T\}$ satisfy $V_{\boldsymbol{\mu}} \leq T^v$ as

$$\mathcal{D}(v) = \{\{D_1, \dots D_T\} : D_t \in \mathcal{D} \text{ for all } t \text{ and } V_{\boldsymbol{\mu}} \leq T^v\}.$$

In the next section, we will show via a minimax lower bound that non-trivial guarantees are only achievable when $v < 1$, and provide an algorithm which achieves the same bound.

Aside: Time-Series Modeling: At this point, we have fully described our model for the Nonstationary Newsvendor. All that remains is to define our performance metric, which we will do in the next subsection. We conclude this subsection with an important practical consideration with respect to time-series models and our variation parameter.

Consider, as an example, the following class of time-series models:

$$d_t = R(t) + S(t) + \epsilon_t. \tag{4.3}$$

Here, $R(t)$ represents a deterministic (and usually simple, e.g. linear) function representing some notion of "trend," and $S(t)$ represents a deterministic, periodic function representing some notion of "seasonality." Finally, all stochastic behavior is captured by the random variables ϵ_t, which are assumed to be independent and mean-zero. This time-series model is classic, and yet drives forecasting algorithms (e.g. exponential smoothing) which are still competitive in modern forecasting competitions [125].

The above model raises an important practical issue: if there exists any (non-trivial) trend $R(\cdot)$ or seasonality $S(\cdot)$, then the demand variation of the sequence of means $\mu_t = R(t) + S(t)$ would scale at least as T, meaning $v = 1$ and no meaningful guarantee will be achievable. Our main observation is that time-series effects like trend and seasonality are easily detected and estimated, so that in any practical

setting, estimates $\hat{R}(\cdot)$ and $\hat{S}(\cdot)$ should be available, and used to "de-trend" and "de-seasonalize" the data. Concretely, the Nonstationary Newsvendor would take place on the sequence

$$\tilde{d}_t = d(t) - \hat{R}(t) - \hat{S}(t) = (R(t) - \hat{R}(t)) + (S(t) - \hat{S}(t)) + \epsilon_t.$$

The resulting sequence of means $\mu_t = (R(t) - \hat{R}(t)) + (S(t) - \hat{S}(t))$ does not stem from the trend and seasonality, but rather the error in estimating the trend and seasonality. It is this error that is assumed to be nonstationary, but with reasonable variation parameter.

Performance Metric: Regret

We conclude this section by formally defining our performance metric for any policy. A **policy** is simply a sequence of mappings $\pi = \{\pi_1, \ldots, \pi_T\}$, where each π_t is a mapping from $d_1, \ldots, d_{t-1}$ to an order quantity $q_t \in Q$ at time t (by convention, π_1 is a constant function).[8] We measure the performance of a policy by its **regret**. Fix a sequence of demand distributions $\boldsymbol{D} = \{D_1, \ldots, D_T\}$. Following the earlier notation from Eq. (4.1), the regret incurred by a policy which selects order quantities $q_1, \ldots, q_T$ is

$$\mathbb{E}^{\pi}_{\boldsymbol{D}} \left[\sum_{t=1}^{T} \left(C_t(\mu_t, q_t) - C_t(\mu_t, q_t^*) \right) \right],$$

where the expectation is with respect to the randomness of the realized demands. Recall that the demand distributions are independent, so q_t^* as defined in (4.1) depends only on D_t. In words, the regret measures the difference between the (expected) total cost incurred by the policy and that of a clairvoyant that knows the underlying demand distributions $\boldsymbol{D} = \{D_1, \ldots, D_T\}$.[9]

We will be concerned with the **worst-case regret** of a policy across families of instances (i.e. sequences of demand distributions) controlled by the variation parameter v:

$$\mathcal{R}^{\pi}(T) = \sup_{\boldsymbol{D} \in \mathcal{D}(v)} \mathbb{E}^{\pi}_{\boldsymbol{D}} \left[\sum_{t=1}^{T} \left(C_t(\mu_t, q_t) - C_t(\mu_t, q_t^*) \right) \right].$$

Note that if the worst-case regret $\mathcal{R}^{\pi}(T)$ of some policy is sublinear in T, then that policy is essentially cost-optimal on average as T goes to infinity. In the next section,

[8] Note that we are not considering randomized policies here, but all of our theoretical results (the lower bounds, in particular) hold even when randomization is allowed.

[9] Note that this is different from a clairvoyant that knows the realized demands $d_1, \ldots, d_T$. Such a clairvoyant would incur zero cost.

we will prove a lower bound on the achievable across all policies, and describe an algorithm which achieves this lower bound.

4.3 Solution to the Nonstationary Newsvendor (without Predictions)

This section contains a complete solution (i.e. matching lower and upper bounds on regret) to the Nonstationary Newsvendor. We begin with the lower bound:

Proposition 10 (Lower Bound: Nonstationary Newsvendor)**.** *For any variation parameter $v \in [0, 1]$, and any policy π (which may depend on the knowledge of v), we have*

$$\mathcal{R}^{\pi}(T) \geq cT^{(3+v)/4},$$

where $c > 0$ is a universal constant.

Proposition 10 is a corollary of a more general lower bound (Proposition 11 in the next section) – it will turn out the Nonstationary Newsvendor is a special case of the Nonstationary Newsvendor with Predictions – so the proof is omitted. Proposition 10 states that the regret of any policy is at least $\Omega(T^{(3+v)/4})$. It is useful to contrast this with two existing results:

1. **Stationary Newsvendor:** In the special case of i.i.d. demand, it is known that the optimal achievable regret is $\Theta(T^{1/2})$ – Example 1 of [35] demonstrates the lower bound, and the SAA method of [110, 111] achieves the upper bound. This point might appear to be incompatible with our result, which states a lower bound of $\Omega(T^{3/4})$ when $v = 0$, but in fact the case of $v = 0$ is more general than i.i.d. demand since it allows $O(T^0) = O(1)$ demand variation, while i.i.d. demand amounts to *zero* demand variation. Indeed, our proof of Proposition 10, for the special case of $v = 0$, utilizes instances for which the demand distribution is allowed to change $T^{1/2}$ times (by an amount of $T^{-1/4}$, resulting in $O(1)$ variation).

   As an aside, this discussion raises a natural question: is the disconnect here between stationary (i.i.d.) demand and variation parameter $v = 0$ a consequence of our use *quadratic* variation, and would the same disconnect arise for other measures of demand variation? In Appendix C.5, we answer both questions in the affirmative by showing that if the exponent 2 in the demand variation (Equation (4.2)) is instead some $\theta \geq 0$, then Proposition 10 generalizes to a lower bound of $\Omega(T^{(1+\theta+v)/(2+\theta)})$. Thus, for any $\theta > 0$, the

case of variation parameter $v = 0$ is meaningfully more general than stationary demand.

2. **Continuous Newsvendor:** A similar "story" plays out in the setting of [96], which recall (among other key differences with our model, as described in Section 4.2) requires the additional assumption that both the demand distributions and the possible order quantities be continuous. [96] show an optimal achievable regret of $\Theta(T^{(1+v)/2})$, which can be contrasted with the stationary (i.i.d.) setting for which an $\Omega(\log T)$ lower bound exists [35]. The following table summarizes these lower bounds:

	Continuous	**General**
Stationary (i.i.d.)	$\log T$	$T^{1/2}$
Nonstationary	$T^{(1+v)/2}$	$T^{(3+v)/4}$

In the next two subsections, we will first analyze a simple algorithm which achieves the lower bound of Proposition 10 when the variation parameter v is known, and then use this as a building block for an algorithm which achieves the same bound when v is unknown.

Upper Bound with Known Variation Parameter v

If we assume that v is known, then designing a policy which achieves regret matching Proposition 10 is fairly straightforward. In fact, a simple policy based on averaging a fixed number of past demand observations does the job ([96] use the same policy). That policy, which we call the **Fixed-Time-Window Policy** is defined in Algorithm 4.

The Fixed-Time-Window Policy uses a carefully-selected "window" size n that is on the order of $T^{(1-v)/2}$. At each time period t, it constructs an estimate $\hat{\mu}_t$ of the mean by averaging the observed demands from the previous n periods, and then selects the optimal order quantity corresponding to $\hat{\mu}_t$. Note that Algorithm 4 also includes a "scaling constant" κ – this should be thought of as a practical tuning parameter, but for the coming theoretical result, it can be chosen arbitrarily (e.g. $\kappa = 1$ suffices).

The following result bounds the worst-case regret of the Fixed-Time-Window Policy:

Lemma 3 (Upper Bound: Nonstationary Newsvendor with Known v)**.** *Fix any variation parameter $v \in [0, 1]$. The Fixed-Time-Window Policy π^{fixed} achieves*

Algorithm 4: Fixed-Time-Window Policy

Inputs: variation parameter $v \in [0, 1]$ and scaling constant $\kappa > 0$
Initialization: $n \leftarrow \lceil \kappa T^{(1-v)/2} \rceil$
for $t = 1, \ldots, n$ **do**
  select $q_t \in Q$ arbitrarily
end
for $t = n + 1, \ldots, T$ **do**
  $\hat{\mu}_t \leftarrow \frac{1}{n} \sum_{s=t-n}^{t-1} d_s$
  if $\hat{\mu}_t \notin [\mu_{\min}, \mu_{\max}]$ **then**
    round $\hat{\mu}_t$ to the nearest value in $[\mu_{\min}, \mu_{\max}]$
  end
  $q_t \leftarrow \arg\min_{q \in Q} C_t(\hat{\mu}_t, q)$
end

worst-case regret

$$\mathcal{R}^{\pi^{\text{fixed}}}(T) \leq C T^{(3+v)/4},$$

where

$$C \leq 3 \max\{b_{\max}, h_{\max}\}(\delta + Q_{\max}) + \ell\left(2\sqrt{\kappa} + \sqrt{\frac{\pi}{36e}} \frac{\delta}{\sqrt{\kappa}}\right)$$

and $\delta = \sup_{\mathcal{D}_\mu \in \mathcal{D}} \|\mathcal{D}_\mu\|_{\psi_2}$.

As promised, Lemma 3 shows that the Fixed-Time-Window Policy achieves regret that matches the lower bound in Proposition 10. Its proof can be found in Appendix C.2, and amounts to bounding the estimation error incurred by demand noise (which is worse for smaller time windows) and demand mean variation (which is worse for larger time windows). The exact time window used in the policy comes from balancing these two sources of error.

Upper Bound with Unknown Variation Parameter v

The lower bound in Proposition 10 holds for policies that "know" v. Naturally, it also holds for policies that do not know v, but an unanswered question at the moment is whether (a) the lower bound should be even larger when v is unknown, or (b) there exists a policy that matches Proposition 10 without knowing v. We show here that case (b) holds by constructing such a policy.[10] Our policy, which we call the **Shrinking-Time-Window Policy** (Algorithm 5), at a high level uses the Fixed-Time-Window Policy with the smallest variation parameter that is consistent with the demand observed so far. In more detail:

[10] As a final comparison to [96], they do not consider the unknown v setting.

1. It begins with a discrete set of candidate variation parameters $\mathcal{V} = \{v_1, \ldots, v_k\}$:

$$v_j = \left(1 + \frac{1}{\log T}\right)^{j-1} \frac{1}{\log T}, \quad j = 1, \ldots, k \tag{4.4}$$

   where k is chosen so that $v_{k-1} < 1 \leq v_k$. $\mathcal{V}$ is defined specifically so that the variation parameters are increasing ($v_{j-1} < v_j$), and so that it discretizes the interval $[0, 1]$ at a sufficiently fine granularity.

2. At any time period t, there is a "current" candidate parameter v_i (initialized to be v_1 at $t = 1$) that is assumed to be the true variation v, and so the corresponding Fixed-Time-Window Policy is applied: a time window of

$$n_i = \lceil \kappa T^{(1-v_i)/2} \rceil \tag{4.5}$$

   is used, and an estimate of μ_t is made:

$$\hat{\mu}_t^i = \frac{1}{n_i} \sum_{s=t-n_i}^{t-1} d_s, \quad \text{rounded to the nearest value in } [\mu_{\min}, \mu_{\max}]. \tag{4.6}$$

3. The index i of the "current" candidate parameter v_i is incremented at any period in which the policy gathers sufficient evidence that $v_i < v$. This is possible due to the following observation: if $v_i \approx v$, then by Lemma 3 we have that for any $v_j > v_i$, the regret incurred by the Fixed-Time-Window Policy corresponding to v_j is $O(T^{(3+v_j)/4})$, and thus the *cumulative difference* between the estimated mean demands ($|\hat{\mu}_t^i - \hat{\mu}_t^j|$) cannot exceed $O(T^{(3+v_j)/4})$. Thus if this is observed for some $v_j > v_i$, we can conclude that $v_i < v$, and i is incremented.

This policy's regret matches (up to log factors) the lower bound in Proposition 10:

Theorem 2 (Upper Bound: Nonstationary Newsvendor with Unknown v)**.** *For any variation parameter $v \in [0, 1]$, the Shrinking-Time-Window Policy $\pi^{\text{shrinking}}$ achieves worst-case regret*

$$\mathcal{R}^{\pi^{\text{shrinking}}}(T) \leq C T^{(3+v)/4} \log^{5/2} T,$$

where $C \leq 3 \max\{b_{\max}, h_{\max}\}(\delta + Q_{\max}) + 12e^{1/4}\ell\left(\gamma + \sqrt{\kappa}\right) + e^{1/4} C_{\text{Lemma 3}}$, and $C_{\text{Lemma 3}}$ is the constant in Lemma 3.

The proof of Theorem 2 can be found in Appendix C.3, but at a high level works as follows:

Proof Sketch of Theorem 2. First note that the total regret incurred during the first $T^{3/4}$ time periods is at most $O(T^{3/4})$. After that, the total regret incurred during

Algorithm 5: Shrinking-Time-Window Policy

Inputs: scaling constants $\kappa > 0$, and γ sufficiently large (Equation (C.2) in Appendix C.3)

Initialization: Set $\mathcal{V} = \{v_1, \ldots, v_k\}$ and $\{n_1, \ldots, n_k\}$ according to Equations (4.4) and (4.5)

for $t = 1, \ldots, T^{3/4}$ **do**
  select $q_t \in Q$ arbitrarily;
end

Initialize $i \leftarrow 1$ and $t_{\textbf{if}} \leftarrow T^{3/4} + 1$

for $t = T^{3/4} + 1, \ldots, T$ **do**
  if $\sum_{s=t_{\textbf{if}}}^{t} |\hat{\mu}_s^i - \hat{\mu}_s^j| \geq 2\left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) T^{(3+v_j)/4}$ for some $j > i$ **then**
    $i \leftarrow i + 1$
    $t_{\textbf{if}} \leftarrow t$
  end
  $q_t \leftarrow \arg\min_{q \in Q} C_t(\hat{\mu}_t^i, q)$.
end

the time periods in which the **if** condition in Algorithm 5 is triggered is at most $O(k)$, where k is the number of candidate variation parameters defined in Equation (4.4), and $k \approx \log^2 T$. Thus, it suffices to bound the total regret incurred *between* successive triggerings of the **if** condition. As a final reduction before proceeding, consider the smallest candidate variation parameter that is at least v, i.e. define ℓ to be the smallest index such that $v \leq v_\ell$. Because $v_\ell = \left(1 + \frac{1}{\log T}\right) v_{\ell-1}$, we have that T^{v_ℓ} is a constant multiple away from T^v. Thus, it will suffice to bound the total regret by $O(T^{(3+v_\ell)/4} \log^{5/2} T)$.

To do this, we first show that (with high probability) throughout the algorithm, the running index i never exceeds ℓ. To see this, consider the following steps:

1. For every $j \geq \ell$, because $v \leq v_j$, by Lemma 3 the Fixed-Time-Window Policy corresponding to the window size n_j has worst-case regret $O(T^{(3+v_j)/4})$. In addition, we can show with high probability (via Hoeffding's inequality) that $\sum_{s=n_j+1}^{T} |\hat{\mu}_s^j - \mu_s| \leq \left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) T^{(3+v_j)/4}$ for every $j \geq \ell$. We may assume this event occurs from now on.
2. For the sake of contradiction, suppose i exceeds ℓ at some period t, or equivalently, there is a period t in which $i \geq \ell$, and the **if** condition is triggered

with some $j > i$. Then

$$\begin{aligned}\sum_{s=t_{\mathbf{if}}}^{t} |\hat{\mu}_s^i - \hat{\mu}_s^j| &\overset{(a)}{\leq} \sum_{s=t_{\mathbf{if}}}^{t} |\hat{\mu}_s^i - \mu_s| + \sum_{s=t_{\mathbf{if}}}^{t} |\hat{\mu}_s^j - \mu_s| \\ &\overset{(b)}{\leq} \sum_{s=n_i+1}^{T} |\hat{\mu}_s^i - \mu_s| + \sum_{s=n_j+1}^{T} |\hat{\mu}_s^j - \mu_s| \\ &\overset{(c)}{\leq} \left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) \cdot T^{(3+v_i)/4} + \left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) \cdot T^{(3+v_j)/4} \\ &\overset{(d)}{<} 2\left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) \cdot T^{(3+v_j)/4},\end{aligned}$$

where (a) is the triangle inequality and (b) holds because $t_{\mathbf{if}} > T^{3/4}$ and $n_i, n_j \leq \kappa T^{1/2}$; since $j > i \geq \ell$, by our assumption in the previous step we get (c); (d) follows since $v_j > v_i$. But this directly contradicts our assumption that the **if** condition is triggered at period t. Therefore i never exceeds ℓ.

Now suppose two consecutive **if** conditions occur at times t' and t''. Between these periods, we may apply the negation of the **if** condition for any $j > i$, and since i never exceeds ℓ, we specifically can take $j = \ell$. This yields

$$\sum_{s=t'+1}^{t''-1} |\hat{\mu}_s^i - \hat{\mu}_s^\ell| < 2\left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) T^{(3+v_\ell)/4}.$$

Then we have

$$\begin{aligned}\sum_{s=t'+1}^{t''-1} |\hat{\mu}_s^i - \mu_s| &\overset{(a)}{\leq} \sum_{s=t'+1}^{t''-1} |\hat{\mu}_s^\ell - \mu_s| + \sum_{s=t'+1}^{t''-1} |\hat{\mu}_s^i - \hat{\mu}_s^\ell| \\ &\overset{(b)}{\leq} \sum_{s=n_i+1}^{T} |\hat{\mu}_s^\ell - \mu_s| + \sum_{s=t'+1}^{t''-1} |\hat{\mu}_s^i - \hat{\mu}_s^\ell| \\ &\overset{(c)}{<} \left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) \cdot T^{(3+v_\ell)/4} + 2\left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) \cdot T^{(3+v_\ell)/4} \\ &= 3\left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) \cdot T^{(3+v_\ell)/4},\end{aligned}$$

where (a) is the triangle inequality and (b) holds because $t' > T^{3/4}$ and $n_i \leq \kappa T^{1/2}$; the first part of (c) follows by our high-probability assumption, and the second part of (c) follows since the **if** condition is not triggered between time t' and time $t'' - 1$; Therefore between two consecutive **if** conditions, the worst-case regret incurred is $O(T^{v_\ell}\sqrt{\log T})$. Because the **if** condition can happen at most $k \approx \log^2 T$ times, the total worst-case regret of the Shrinking-Time-Window Policy is $O(T^{v_\ell} \log^{5/2} T) = O(T^v \log^{5/2} T)$. ■

This concludes our discussion of the Nonstationary Newsvendor. In the next section, we turn to the second subject of this paper, which is the same problem with predictions.

4.4 The Nonstationary Newsvendor with Predictions

As described in the introduction, it is likely that when the Nonstationary Newsvendor is faced practice, some notion of a "prediction" of future demand will be made. Such predictions can come from a diverse set of sources ranging from simple human judgement, to forecasting algorithms built on previous demand data, to more-sophisticated machine learning algorithms trained on feature information. The process of sourcing or constructing such predictions is orthogonal to our work. Instead, we treat these predictions as given to us endogenously (and in particular, we make no assumption on the accuracy of these predictions), and attempt to use these predictions optimally.

Model

The **Nonstationary Newsvendor with Predictions** problem assumes all of the setup, assumptions, and notation of the previous Nonstationary Newsvendor problem. In addition, at each time period t, we assume that the decision-maker receives a **prediction** a_t before selecting an order quantity $q_t \in Q$.[11] This prediction is meant to be an estimate of μ_t, and so we measure the **prediction error** of a sequence $\boldsymbol{a} = \{a_1, \ldots, a_T\}$ with respect to a sequence of means simply as

$$\sum_{t=1}^{T} |a_t - \mu_t|.$$

Note that unlike demand variation, we have not used partitions here (and in fact, introducing partitions would not have any effect since we are measuring *absolute* rather than squared differences). Intuitively, we do not want to require the sequence of errors to be meaningful time-series: the predictions are generic, and their accuracy is allowed to change rapidly. Just as for the demand variation, the prediction error is expected to grow with the time horizon T, and the proper parameterization of this growth is via an exponent: we call the **accuracy parameter** the smallest $a \in [0, 1]$ such that the prediction error is at most T^a. We will always assume that a is unknown to the decision-maker.

[11] We are taking the predictions to be entirely deterministic, so for example, a_t is not allowed to depend on the previously-observed demands $d_1, \ldots, d_{t-1}$. Our results hold if we extend to the setting in which the predictions are stochastic (and adapted to the demand filtration).

Algorithm 6: Prediction Policy

for $t = 1, \ldots, T$ **do**
  $\hat{\mu}_t \leftarrow a_t$ (if $\hat{\mu}_t \notin [\mu_{\min}, \mu_{\max}]$, round $\hat{\mu}_t$ to the nearest value in $[\mu_{\min}, \mu_{\max}]$)
  $q_t \leftarrow \arg\min_{q \in Q} C_t(\hat{\mu}_t, q)$.
end

Naturally, the notion of a **policy** π expands to include the predictions: $\pi = \{\pi_1, \ldots, \pi_T\}$, where each π_t is a mapping from $d_1, \ldots, d_{t-1}$ and $a_1, \ldots, a_t$ to an order quantity $q_t \in Q$. The simplest policy, which "should" be used if the prediction error is known to be sufficiently small, is to simply behave as if the predictions were perfect. We call this the **Prediction Policy** (Algorithm 6). The following observation collects a few (likely unsurprising) facts about the performance of this policy, with respect to worst-case regret (generalized in the "obvious" manner to incorporate prediction accuracy via the accuracy parameter a):

Observation 3 (**Upper and Lower Bounds: Prediction Policy**). *Fix any variation parameter $v \in [0, 1]$ and any accuracy parameter $a \in [0, 1]$.*

a) The Prediction Policy $\pi^{\text{prediction}}$ achieves worst-case regret

$$\mathcal{R}^{\pi^{\text{prediction}}}(T) \leq CT^a,$$

where $C \leq 2\ell$.

b) For any policy π (which may depend on the knowledge of a) that is solely a function of the predictions (i.e. does not depend on the observed demands), we have

$$\mathcal{R}^{\pi}(T) \geq cT^a,$$

where $c > 0$ is a universal constant.

Observation 3a) states that the Prediction Policy translates prediction error directly to regret (incidentally, it does this without "knowing" a). There are of course other ways in which the predictions could be used, but Observation 3b) essentially states that there is nothing to be gained by doing so (even if a is known). The proof of Observation 3a) appears in Appendix C.1. Observation 3b) is a direct corollary of Proposition 11, which is given in the next subsection.

Extreme Cases

What exactly is achievable for the Nonstationary Newsvendor with Predictions depends heavily on whether or not v and a are known to the policy. To see this, it is worth first considering the two extremes.

Case 1: Known v and a. A simple policy is available when v and a are both known. Compare the quantities $(3 + v)/4$ and a. If the former quantity is smaller, apply the Fixed-Time-Window Policy. If the latter is smaller, apply the Prediction Policy. Lemma 3 and Observation 3 together imply that this achieves a worst-case regret of $O(T^{\min\{(3+v)/4,a\}})$. This is optimal, as demonstrated by the following result:

Proposition 11 (Lower Bound: Known v and a)**.** *Fix any variation parameter $v \in [0, 1]$ and any accuracy parameter $a \in [0, 1]$. For any policy π (which may depend on the knowledge of v and a), we have*

$$\mathcal{R}^{\pi}(T) \geq cT^{\min\{(3+v)/4,a\}},$$

where $c > 0$ is a universal constant.

The proof of this result can be found in Appendix C.4, and relies on an explicit construction of a family of problem instances. Our construction breaks the total time horizon into cycles wherein the demand distribution is i.i.d.. We tune the length of each cycle to be small enough so that it is (provably) hard to detect the change in demand distributions and the predictions are essentially useless for most time periods in the cycle, and large enough so that the demand variation is within T^v and the prediction error is within T^a.

Case 2: Unknown v and a. At the opposite extreme, if v and a are both unknown, is it still possible to achieve $O(T^{\min\{(3+v)/4,a\}})$ worst-case regret? The answer is no:

Proposition 12 (Lower Bound: Unknown v and a)**.** *For any policy that does not depend on the knowledge of v or a, there exists a problem instance such that $a \neq (3 + v)/4$, and the policy incurs regret at least $cT^{\max\{(3+v)/4,a\}}$ on the instance, where $c > 0$ is a universal constant.*

Proposition 12 states that the best we can hope for, when v and a are unknown, is a worst-case regret of at least $\Omega(T^{\max\{(3+v)/4,a\}})$.[12] Note that Proposition 12 shows

[12] Indeed, it implies that no algorithm can achieve regret $O(T^{f(v,a)})$ for a function $f : [0, 1] \times [0, 1] \to [0, 1]$ satisfying $f(v, a) \leq \max\{(3 + v)/4, a\}$ for all $v, a \in [0, 1]$ and $f(v, a) < \max\{(3 + v)/4, a\}$ for some $v, a \in [0, 1]$.

there exists a pair of v and a, and a corresponding problem instance, such that this lower bound holds. This is in contrast to a result showing that *for any* pair of v and a, there exists a problem instance such that the lower bound holds, as is common in the literature (e.g. Theorem 1 of [95]). This lower bound is easily achieved, for example by applying the Shrinking-Time-Window Policy or the Prediction Policy (or any blind randomization of the two). The proof of Proposition 12 is in Appendix C.6. In contrast to Case 1, the lower bound construction here relies heavily on the fact that we do not know which one of $(3 + v)/4$ and a is smaller.

Final Solution

We have finally reached the problem which motivates this entire paper: designing an optimal policy for the Nonstationary Newsvendor with Predictions when the prediction error a is unknown. We will assume that v is known, since when v is unknown, Proposition 12 rules out the possibility of using the predictions to improve on what is already achievable without predictions. On the other hand, by Proposition 11, the absolute best we could hope for is a policy which achieves a worst-case regret of $O(T^{\min\{(3+v)/4,a\}})$. In words, we would like a policy which, without knowing a, achieves the same regret had a been known. Our main result is the design of such a policy.

Our policy is called the **Prediction-Error-Robust Policy (PERP)**, and is given in Algorithm 7. PERP utilizes the Fixed-Time-Window policy π^{fixed} in Section 4.3 as an estimate of the true mean to track the quality of the predictions over time.

Theorem 3 (Upper Bound: Known v and Unknown a)**.** *For any variation parameter $v \in [0, 1]$ and any accuracy parameter $a \in [0, 1]$, the Prediction-Error-Robust Policy π^{PERP} achieves worst-case regret*

$$\mathcal{R}^{\pi^{\text{PERP}}}(T) \leq \min\{3C_{\text{Lemma 3}}\sqrt{\log T} \cdot T^{(3+v)/4}, 2\ell \cdot T^{a}\},$$

where $C_{\text{Lemma 3}}$ is the constant in Lemma 3 (and 2ℓ matches the constant in Observation 3a).

The intuition behind PERP is to follow the predictions until a time that is late enough to have evidence that the prediction quality is bad (compared to the Fixed-Time-Window Policy), while early enough to not incur much regret caused by the poor quality of the predictions. Because we do not observe the true past mean μ_t after time period t, we naturally use $\hat{\mu}_t^{\text{fixed}}$ from the Fixed-Time-Window policy π^{fixed} as an

Algorithm 7: Prediction-Error-Robust Policy (PERP)

Inputs: variation parameter $v \in [0, 1]$ and scaling constants $\kappa > 0$, γ sufficiently large (Equation (C.3) in Appendix C.7)

Initialization: $n \leftarrow \lceil \kappa T^{(1-v)/2} \rceil$

for $t = 1, \ldots, n$ **do**
  $\pi_t \leftarrow \pi_t^{\text{prediction}}$
end
for $t = n+1, \ldots, T$ **do**
  $\hat{\mu}_t^{\text{fixed}} \leftarrow \frac{1}{n} \sum_{s=t-n}^{t-1} d_s$
  $\hat{\mu}_t^{a} \leftarrow a_t$
  if $\sum_{s=n+1}^{t} |\hat{\mu}_s^a - \hat{\mu}_s^{\text{fixed}}| \geq \left(\gamma \sqrt{\log T} + \sqrt{\kappa} + 1\right) \cdot T^{(3+v)/4}$ **then**
    $\pi_t \leftarrow \pi_t^{\text{fixed}}$
    break
  end
  else
    $\pi_t \leftarrow \pi_t^{\text{prediction}}$.
  end
end

estimation of μ_t, and in turn keep tracking the cumulative difference the prediction quality $|a_t - \mu_t|$. We carefully choose the parameters in π^{PERP} so that this estimation is not accurate only with a small probability, and we can identify the prediction quality is bad if this cumulative difference is too large. By Proposition 11, any policy can only achieve worst-case regret on the order of $T^{\min\{(3+v)/4, a\}}$, so PERP is order-optimal.

Aside: Unknown v and Known a. There are four possible scenarios depending on the knowledge of v and a: known/unknown v and known/unknown a. So far we have discussed three of them: known v and a (Proposition 11), unknown v and a (Proposition 12), and known v and unknown a (Theorem 3). For the sake of completeness, we discuss the remaining case of unknown v and known a in Appendix C.8, where we give a policy that achieves worst-case regret $\tilde{O}(T^{\min\{(3+v)/4, a\}})$. This is order-optimal by Theorem 3.

4.5 Experiments

Finally, we describe a set of experiments we performed to evaluate our policy (`PERP`) for the Nonstationary Newsvendor with Predictions. In all of our experiments, we compared `PERP` against the Shrinking-Time-Window Policy (`NO-PRED`) and the

Prediction Policy (PURE-PRED). The main takeaways are:

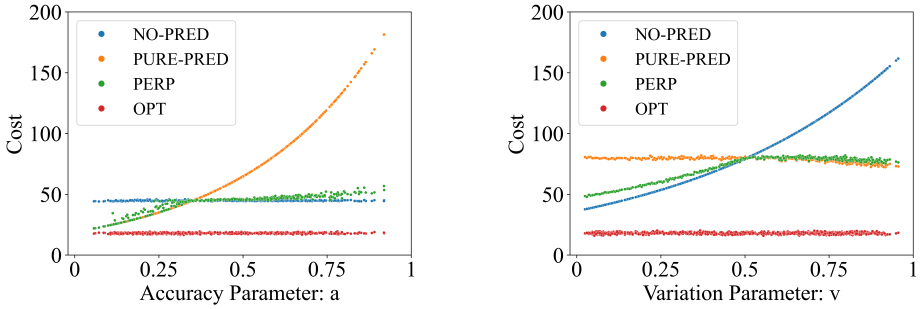
1. PERP's performance is robust with respect to the quality of the predictions, without knowing the prediction quality beforehand. Specifically, the (newsvendor) cost it incurs is consistently "close" to the lower of the costs incurred by NO-PRED and PURE-PRED.
2. PERP performs especially well when the absolute difference between the costs of NO-PRED and PURE-PRED is large, i.e. when the "stakes" are highest.

Experiments on Synthetic Data

The objective of our first batch of experiments was to fix one of the two theoretical parameters (v or a) and test PERP's performance as the other parameters changes. To generate demand sequences, we used the parametric time-series model that corresponds to triple exponential smoothing (Holt Winters), a classic model for time-series data in the family of Equation (4.3). We give the exact formulas, and our choices of parameters, in Appendix C.9; for more on triple exponential smoothing, see [175]. In our experiment, each demand sequence consisted of the demands for the next 365 time periods, with the realized demands generated as Poisson variables with the corresponding means. We ran two sets of experiments:

- **Fixed v:** We fixed a single set of parameters for the demand sequence and generated 1,000 different predictions of this demand sequence, each from a set of "predicted" parameters with different accuracy. Thus the variation parameter v was fixed, and the accuracy parameter a varied across instances.
- **Fixed a:** We generated 1,000 demand sequences by selecting the parameters randomly. We then generated predictions by changing each parameter 10% and using the corresponding sequence. Thus the variation parameter v varied across instances, but the accuracy parameter a was (roughly) fixed.

For each demand sequence and corresponding prediction, we ran NO-PRED, PURE-PRED, and PERP with equal overage and underage costs, and scaling constants $\kappa = \gamma = 1$. Because the experiment was synthetic, the true underlying demand distribution was known at each time period. Therefore we also ran OPT as a benchmark, which simply ordered the optimal quantile at each time period. The variation parameter v in PERP was calculated using the past demands of the pre-fixed 30 time periods by the definition in Section 2.2. We calculated the parameters v and a by their definitions (given in Section 2.2 and Section 4.1, respectively), scaled appropriately to make



(a) Fixed v (b) Fixed a

Figure 4.2: The costs of NO-PRED, PURE-PRED, PERP, and OPT when (a) the variation parameter v is fixed, and (b) the accuracy parameter a is fixed. Each dot represents the cost of the corresponding policy on a given instance.

them lie in $[0, 1]$. The resulted scatter plots are shown in Figure 4.2. In (a), v is fixed, so the cost of NO-PRED (blue dots) is approximately the same for all instances. The cost of PURE-PRED (orange dots) is approximately exponential in a, which follows by Observation 3a). In (b), a is fixed, so the cost of PURE-PRED is approximately the same for all instances. The cost of NO-PRED is approximately exponential in v, which follows by Theorem 2. Note that in both (a) and (b), the cost of PERP (green dots) is close to the minimal cost of NO-PRED and PURE-PRED across all instances, showing that PERP's performance is robust in both v and a.

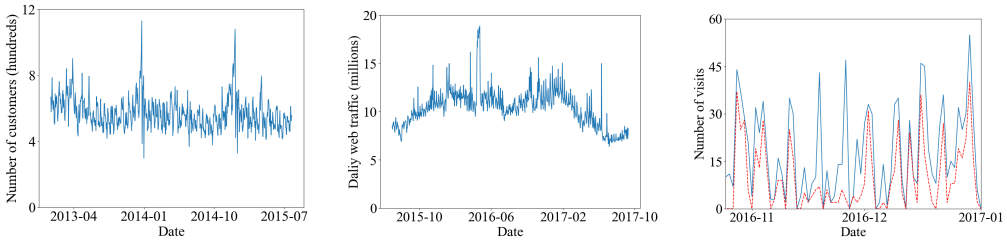
Experiments on Real Data

We used real-world datasets to represent the "demand" sequences in our experiments. Figure 4.3 depicts example time series from each of these datasets. All datasets include *multiple* daily time series and are publicly available:

- **Rossmann:**[13] Daily number of customers that visited each of 1,115 stores in the *Rossmann* drug store chain during a 781-day period in 2013-2015.
- **Wikipedia:**[14] Daily web traffic across *Wikipedia.com* pages of 9 different languages for an 803-day period from 2015 to 2017.
- **Restaurant:**[15] Daily number of visitors and online reservations across 185 restaurants in Japan, during a 478-day period in 2016-2017. We treated

[13]Available at https://www.kaggle.com/competitions/rossmann-store-sales/data
[14]Available at https://www.kaggle.com/competitions/web-traffic-time-series-forecasting/data
[15]Available at https://www.kaggle.com/competitions/recruit-restaurant-visitor-forecasting/

the number of visitors as the "demand," and the reservations as a predictive feature.



(a) Rossmann (b) Wikipedia (c) Restaurant

Figure 4.3: An example of a single time series from each dataset. In (c), the red dashed line represents an additional fetaure: daily online reservations.

Each *instance* of our experiment represented a single Nonstationary Newsvendor with Predictions problem, with the realized demands taken from a single time series in our data (a single Rossmann store, a single language on Wikipedia, or a single restaurant). The overage and underage costs were constant within each instance, and without loss of generality the two costs for an instance can be characterized by the corresponding critical quantile (specifically the ratio of the underage cost to the sum). The time horizon for each instance was a set number of days taken from the end of the time series, with the preceding days used to train one of four prediction methods. These predictions were also updated over the course of the instance at a set frequency. For the Wikipedia dataset, this yielded a total of 2,880 possible instances, all of which were tested. The Rossmann dataset has multiple orders of magnitude more instances, so we randomly sampled 1,000 from this set. For the Restaurant dataset, we used a single prediction method to generate two sets of predictions for each restaurant: one only utilized the number of past visitors and the other incorporated the number of reservations as a feature, which gave 740 instances. Table 4.3 describes all of the instances used.

For each instance, we applied `NO-PRED`, `PURE-PRED`, and `PERP` with scaling constants $\kappa = \gamma = 1$, and the variation parameter v in `PERP` was calculated using the past demands of the training data by the definition in Section 2.2. To generate predictions, we used four popular forecasting method ranging from classic to the state-of-the art:

- **Exponential Smoothing (Holt Winters):** A classic algorithm based on a (linear) trend and seasonality decomposition as in Equation (4.3), known for

	Rossmann	**Wikipedia**	**Restaurant**
Number of time series	1,115	9	185
Critical quantiles (%)	30,40,50,60,70	95,98,99,99.9	50
Experimental period (days)	300,400,500,600	300,400,500,600,700	100
Prediction update frequency (days)	2,4,10,20	2,4,10,20	5, 10
Total number of instances	1,000 (sampled)	2,880 (exhaustive)	740 (exhaustive)

Table 4.3: Description of experimental instances.

its simplicity and robust performance. It is frequently used as a benchmark in forecasting competitions [125]. Tuning parameter: seasonality of length 50.

- **ARIMA:** Another classic algorithm that is rich enough to model a wide class of nonstationary time-series. Tuning parameters: $(p, q, r) = (3, 2, 5)$.
- **Prophet:** A recent algorithm developed by Facebook [162] based on a (piecewise-linear) trend and seasonality decomposition as in Equation (4.3), known to work well in practice with minimal tuning. Tuning parameters: software default.
- **LightGBM:** A recent algorithm developed by Microsoft [94] based on tree algorithms. LightGBM formed the core of most of the top entries in the recent $100,000 M5 Forecasting Challenge [126]. Tuning parameters: software default.

For the Restaurant dataset, we used Prophet as the forecasting method, with and without the reservations as an additive linear feature. We treated the outputs of these methods as predictions of the *mean* demand. To estimate the demand distribution around this mean, we used the empirical distribution of the residuals of the same predictions on the training period.[16] In practice, even if the prediction quality is good,

[16]That is, if the training data consists of T_{train} periods, which without loss we index as $\{t = -T_{\text{train}} + 1, T_{\text{train}} + 2, \ldots, -1, 0\}$, then the demand distribution at any time t was estimated to be

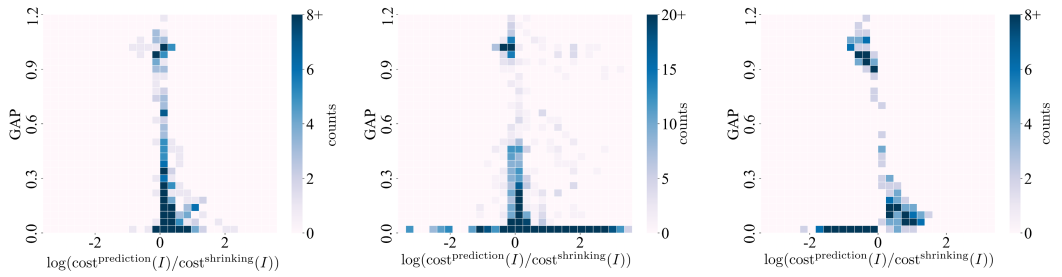
$$\hat{\mu}_t + \text{Uniform}\left(\{d_s - \hat{\mu}_s : s = -T_{\text{train}} + 1, T_{\text{train}} + 2, \ldots, -1, 0\}\right).$$

the predictions of the first few days might incur large costs due to noise/instability of the predictions, which may cause PERP to misidentify the prediction quality. Therefore we restricted PERP to following the predictions for the first 20 days, only allowing switches afterward.

Results. Each instance yields three total costs: one incurred by PERP, and two incurred by the benchmark algorithms (NO-PRED and PURE-PRED). The primary performance metric we report is a form of optimality gap. For an instance I, let $\text{cost}^{\text{PURE-PRED}}(I)$ be the cost of PURE-PRED, and similarly define $\text{cost}^{\text{NO-PRED}}(I)$, $\text{cost}^{\text{PERP}}(I)$. Then the optimality gap (GAP) of PERP is defined as

$$\text{GAP}(I) = \frac{\text{cost}^{\text{PERP}}(I) - \min\{\text{cost}^{\text{PURE-PRED}}(I), \text{cost}^{\text{NO-PRED}}(I)\}}{\left|\text{cost}^{\text{PURE-PRED}}(I) - \text{cost}^{\text{NO-PRED}}(I)\right|}.$$

If we think of PERP as trying to achieve the minimum of the costs incurred by the two benchmark policies, then GAP measures the excess cost that PERP incurs on top of this minimum, normalized so that GAP = 0 implies that the minimum has been achieved, and GAP = 1 implies that the maximum of the two costs was incurred.[17]



(a) Rossmann (b) Wikipedia (c) Restaurant

Figure 4.4: Histograms of GAPs across (a) 1,000 randomly-sampled instances on the Rossmann dataset, (b) 2,880 instances on the Wikipedia dataset, and (c) 740 instances on the Restaurant dataset.

Experiments on the datasets yielded the histograms in Figure 4.4. For each instance I, the value on the horizonal axis is $\log(\text{cost}^{\text{PURE-PRED}}(I)/\text{cost}^{\text{NO-PRED}}(I))$, which is greater than 0 if NO-PRED has a lower cost, and less than 0 if PURE-PRED has a lower cost. In the 1,000 Rossman instances NO-PRED had a lower cost 82.7% of the time, in the 2,880 Wikipedia instances NO-PRED had a lower cost 81.9% of the time, and in the 740 Restaurant instances NO-PRED had a lower cost 64.3% of the time. The values on the vertical axis are the GAPs. Note that most GAPs are small when the absolute

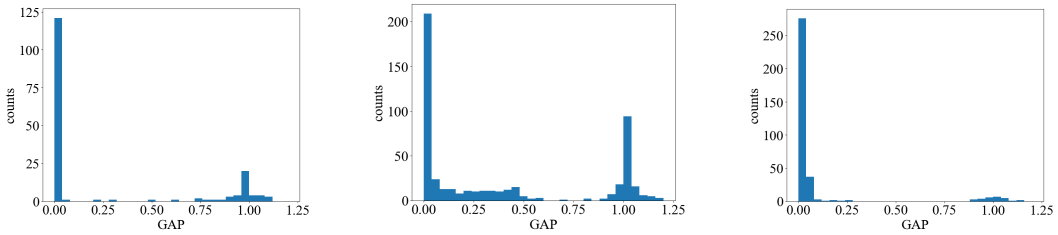
[17] GAP may technically be outside of $[0, 1]$.

values of the log difference are large. This shows PERP performs very well when the difference of costs between NO-PRED and PURE-PRED is large. On the other hand, there are instances where PERP has large GAPs, in particular there are instances with GAPs equal to 1 when the log difference of costs is close to 0. This happens because when the log difference of costs is close to 0, the cost of NO-PRED and the cost of PURE-PRED are close, so PERP may misidentify the prediction quality. Still, since the max cost and the min cost of the other two policies are close, even the GAPs are large in these instances, PERP does not perform badly.

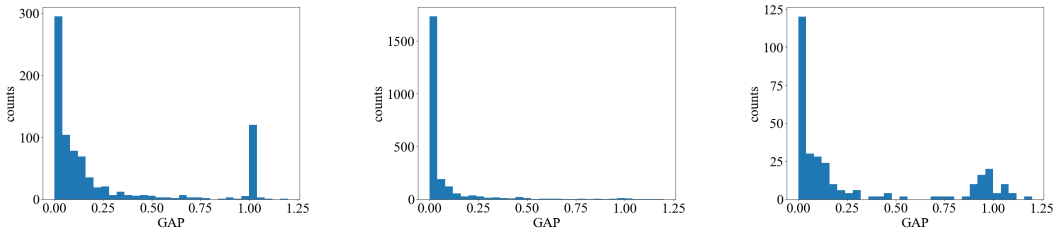
	Rossmann	**Wikipedia**	**Restaurant**
Average GAP with good predictions	0.26	0.40	0.10
Average GAP with bad predictions	0.28	0.07	0.39

Table 4.4: Summary of experimental results.

We further divide the instances according to which of NO-PRED and PURE-PRED had lower cost in Table 4.4 and Figure 4.5. For comparison, if we did not know the



(a) GAP with high prediction accuracy. Left to right: Rossmann, Wikipedia, Restaurant.



(b) GAP with low prediction accuracy. Left to right: Rossmann, Wikipedia, Restaurant.

Figure 4.5: Histograms of the GAPs for (a) 173 Rossmann instances (left), 522 Wikipedia instances (middle), 476 Restaurant instances (right) for which PURE-PRED has lower cost, and (b) 827 Rossmann instances (left), 2358 Wikipedia instances (middle), 264 Restaurant instances (right) for which NO-PRED has lower cost.

prediction quality beforehand, uniformly random choosing between NO-PRED and PURE-PRED has an expected GAP of 0.5. Therefore PERP outperforms this natural benchmark in all cases of all datasets.

4.6 Conclusion

We proposed a new model incorporating predictions into the nonstationary newsvendor problem. We first gave a complete analysis of the Nonstationary Newsvendor (without predictions) by proving a lower regret bound and developing the Shrinking-Time-Window Policy, which was the first policy that achieves the lower bound up to log factors without knowing the variation parameter. We then considered the Nonstationary Newsvendor with Predictions and proposed the Prediction-Error-Robust Policy, which does not need to know the prediction quality beforehand, and achieves nearly optimal minimax worst-cast regret.

Chapter 5

CONCLUSION

This thesis studies a central question in modern operations management: how can firms fully leverage the power of modern AI in sequential decision-making while retaining the rigor of optimization? Across a wide range of operational settings, AI models now provide increasingly rich forms of guidance, from demand forecasts and estimated opportunity costs to expressive neural architectures that capture complex behavioral interactions. Yet the availability of such models does not by itself resolve the underlying decision problem. Their outputs must still be translated into actions that are timely, scalable, and reliable. This thesis develops optimization foundations for AI-driven sequential decision-making, showing how modern AI can be incorporated into operational decisions in a way that is computationally tractable, robust to misspecification, and supported by formal performance guarantees.

A unifying theme across the thesis is that AI creates value for operations only when its outputs induce decision-relevant structure. In Chapter 2, this structure takes the form of a tractable subclass of transformer architectures. Transformers are powerful because they capture interaction effects across recommended items, but this same expressiveness can make downstream optimization computationally intractable. By identifying the class of simple transformers, the chapter shows that one can preserve key behavioral effects, including variety seeking, complementarity, and substitution, while still enabling near-optimal recommendation in sublinear time. The chapter further demonstrates, through both offline experiments and large-scale field evidence, that these behavioral effects matter in practice and can generate measurable gains in user engagement. More broadly, the chapter illustrates that expressive AI models need not be black boxes for operations; when their structural properties are understood, they can be integrated into real-time optimization with theoretical rigor.

In Chapters 3 and 4, AI enters the decision process in a different way: not as the objective itself, but as a source of guidance for sequential decisions under uncertainty. Chapter 3 studies online resource allocation with predictions in the form of shadow prices. A central challenge in such settings is that prediction quality is unknown in advance, and algorithms that rely too heavily on inaccurate guidance may perform poorly. The chapter addresses this challenge by designing allocation policies that

adapt to prediction quality without requiring it as input. The resulting guarantees sharply characterize how one can balance robustness and consistency: the algorithm remains protected against poor predictions while improving automatically as the predictions become more accurate. Chapter 4 develops the same philosophy in inventory control under nonstationary demand. It first provides a complete regret characterization of the nonstationary newsvendor problem without predictions, and then shows how generic demand forecasts can be incorporated without assuming prior knowledge of their quality. The resulting policies adapt simultaneously to environmental change and to prediction error, remaining robust when forecasts are poor while benefiting when they are informative.

Taken together, these chapters suggest three broader principles for AI-driven operations. First, predictive or representational power alone is not enough; what matters is whether that power can be converted into actionable structure for optimization. Second, because model quality is typically unknown and can change over time, algorithms must be adaptive to the reliability of AI guidance rather than assuming it is correct. Third, progress in this area requires end-to-end guarantees that connect the AI model, the induced optimization problem, and the resulting sequential performance. These principles provide a common foundation across seemingly different settings, from personalization to revenue management to inventory control.

Beyond the specific technical results, the broader implication of this thesis is that AI and operations management should not be treated as separate layers of a system, with one producing predictions and the other consuming them passively. Instead, they should be designed jointly. The central opportunity is not merely to make better predictions, but to understand how AI models can reshape the structure of operational decisions themselves. This perspective becomes increasingly important as operational environments become more dynamic, as decision latency becomes more stringent, and as firms rely on AI models that are more expressive but also harder to optimize over directly.

Several directions for future research follow naturally from this work. One direction is to study richer model classes beyond the structured settings considered here, and to identify broader families of neural architectures that admit efficient optimization with guarantees. A second direction is to understand settings in which the quality of AI guidance evolves endogenously over time, for example because predictions are updated online or because decisions themselves affect the future data-generating process. A third direction is to extend these ideas to settings with strategic behavior,

partial observability, or multiple interacting decision-makers, where the interface between AI and optimization becomes even more subtle. From an applied standpoint, there is also substantial opportunity to bring these methods to a wider range of operational domains, including digital platforms, retail media, supply chains, and service systems, where firms increasingly require decisions that are both behaviorally informed and operationally scalable.

In summary, this thesis argues that the next stage of research in operations management is not simply to apply AI to existing decision problems, but to develop the optimization principles that make AI actionable. By studying transformer-based personalization, prediction-robust online allocation, and nonstationary inventory control, the thesis takes several steps toward that goal. I hope these results contribute to a broader research agenda at the interface of AI, optimization, and operations management, where modern predictive models are not treated as opaque tools, but as structured inputs to decision-making algorithms that are both practically effective and theoretically grounded.

BIBLIOGRAPHY

[1] Shipra Agrawal, Zizhuo Wang, and Yinyu Ye. A dynamic near-optimal algorithm for online linear programming. *Operations Research*, 62(4):876–890, 2014.

[2] Nir Ailon and Bernard Chazelle. The fast johnson–lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on computing*, 39(1):302–322, 2009.

[3] Josh Alman and Zhao Song. Fast attention requires bounded entries. *Advances in Neural Information Processing Systems*, 36, 2024.

[4] Lin An, Andrew A Li, Benjamin Moseley, and Gabriel Visotsky. Best of many in both worlds: Online resource allocation with predictions under unknown arrival model. *arXiv preprint arXiv:2402.13530*, 2024.

[5] Lin An, Andrew A Li, Benjamin Moseley, and R Ravi. The nonstationary newsvendor with (and without) predictions. *Manufacturing & Service Operations Management*, 27(3):881–896, 2025.

[6] Lin An, Andrew A Li, Vaisnavi Nemala, and Gabriel Visotsky. Real-time personalization with simple transformers. *ACM SIGMETRICS Performance Evaluation Review*, 53(2):116–118, 2025.

[7] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

[8] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal lsh for angular distance. *Advances in neural information processing systems*, 28, 2015.

[9] Antonios Antoniadis, Themis Gouleakis, Pieter Kleer, and Pavel Kolev. Secretary and online matching problems with machine learned advice. *Advances in Neural Information Processing Systems*, 33:7933–7944, 2020.

[10] Alessandro Arlotto and Itai Gurvich. Uniformly bounded regret in the multisecretary problem. *Stochastic Systems*, 9(3):231–260, 2019.

[11] Kenneth Joseph Arrow et al. Studies in the mathematical theory of inventory and production. *JSTOR*, 1958.

[12] Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Y Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923, 1998.

[13] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32 (1):48–77, 2002.

[14] Yossi Aviv and Amit Pazgal. A partially observed markov decision process for dynamic pricing. *Management science*, 51(9):1400–1416, 2005.

[15] Katy S Azoury. Bayes solution to dynamic inventory models under unknown demand distribution. *Management science*, 31(9):1150–1160, 1985.

[16] Dheeraj Baby and Yu-Xiang Wang. Online forecasting of total-variation-bounded sequences. *NIPS*, 32, 2019.

[17] Yong Bai, Yu-Jie Zhang, Peng Zhao, Masashi Sugiyama, and Zhi-Hua Zhou. Adapting to online label shift with provable guarantees. *NIPS*, 35:29960–29974, 2022.

[18] Michael O Ball and Maurice Queyranne. Toward robust revenue management: Competitive analysis of online booking. *Operations Research*, 57(4):950–963, 2009.

[19] Santiago Balseiro, Haihao Lu, and Vahab Mirrokni. Dual mirror descent for online allocation problems. In *International Conference on Machine Learning*, pages 613–628. PMLR, 2020.

[20] Santiago Balseiro, Christian Kroer, and Rachitesh Kumar. Single-leg revenue management with advice. *arXiv preprint arXiv:2202.10939*, 2022.

[21] Santiago R Balseiro and Yonatan Gur. Learning in repeated auctions with budgets: Regret minimization and equilibrium. *Management Science*, 65(9): 3952–3968, 2019.

[22] Santiago R Balseiro, Haihao Lu, and Vahab Mirrokni. The best of many worlds: Dual mirror descent for online allocation problems. *Operations Research*, 71(1):101–119, 2023.

[23] Etienne Bamas, Andreas Maggiori, and Ola Svensson. The primal-dual method for learning augmented algorithms. *Advances in Neural Information Processing Systems*, 33:20083–20094, 2020.

[24] Gah-Yi Ban and Cynthia Rudin. The big data newsvendor: Practical insights from machine learning. *Operations Research*, 67(1):90–108, 2019.

[25] Salvador Barberà, Peter Hammond, and Christian Seidl. *Handbook of utility theory: volume 2 extensions*. Springer Science & Business Media, 2004.

[26] Manel Baucells and Rakesh K Sarin. Satiation in discounted utility. *Operations research*, 55(1):170–181, 2007.

[27] Heinz H Bauschke, Jonathan M Borwein, and Patrick L Combettes. Essential smoothness, essential strict convexity, and legendre functions in banach spaces. *Communications in Contemporary Mathematics*, 3(04):615–647, 2001.

[28] Cristina Bazgan, Hadrien Hugot, and Daniel Vanderpooten. Implementing an efficient fptas for the 0–1 multi-objective knapsack problem. *European Journal of Operational Research*, 198(1):47–56, 2009.

[29] Cristina Bazgan, Hadrien Hugot, and Daniel Vanderpooten. Solving efficiently the 0–1 multi-objective knapsack problem. *Computers & Operations Research*, 36(1):260–279, 2009.

[30] Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

[31] Walid Bendada, Théo Bontempelli, Mathieu Morlon, Benjamin Chapus, Thibault Cador, Thomas Bouabça, and Guillaume Salha-Galvan. Track mix generation on music streaming services using transformers. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 112–115, 2023.

[32] Steve Berry, Ahmed Khwaja, Vineet Kumar, Andres Musalem, Kenneth C Wilbur, Greg Allenby, Bharat Anand, Pradeep Chintagunta, W Michael Hanemann, Przemek Jeziorski, et al. Structural models of complementary choices. *Marketing Letters*, 25:245–256, 2014.

[33] Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[34] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[35] Omar Besbes and Alp Muharremoglu. On implications of demand censoring in the newsvendor problem. *Management Science*, 59(6):1407–1424, 2013.

[36] Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. *Advances in neural information processing systems*, 27, 2014.

[37] Omar Besbes, Yonatan Gur, and Assaf Zeevi. Non-stationary stochastic optimization. *Operations research*, 63(5):1227–1244, 2015.

[38] Alberto Bietti, Vivien Cabannes, Diane Bouchacourt, Herve Jegou, and Leon Bottou. Birth of a transformer: A memory viewpoint. *Advances in Neural Information Processing Systems*, 36, 2024.

[39] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[40] Niv Buchbinder, Kamal Jain, and Joseph Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *European Symposium on Algorithms*, pages 253–264. Springer, 2007.

[41] Apostolos N Burnetas and Craig E Smith. Adaptive ordering and pricing for perishable products. *Operations Research*, 48(3):436–443, 2000.

[42] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 335–336, New York, NY, 1998. ACM.

[43] Yair Carmon and Oliver Hinder. Making sgd parameter-free. In *Conference on Learning Theory*, pages 2360–2389. PMLR, 2022.

[44] Marjan Celikik, Ana Peleteiro Ramallo, and Jacek Wasilewski. Reusable self-attention recommender systems in fashion industry applications. In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 448–451, 2022.

[45] Kamalika Chaudhuri, Yoav Freund, and Daniel J Hsu. A parameter-free hedging algorithm. *Advances in neural information processing systems*, 22, 2009.

[46] Boxiao Chen, Xiuli Chao, and Hyun-Soo Ahn. Coordinating pricing and inventory replenishment with nonparametric demand learning. *Operations Research*, 67(4):1035–1052, 2019.

[47] Ching-Wei Chen, Paul Lamere, Markus Schedl, and Hamed Zamani. Recsys challenge 2018: Automatic music playlist continuation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 527–528, 2018.

[48] Jianer Chen, Xiuzhen Huang, Iyad A Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 72(8):1346–1367, 2006.

[49] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 335–344, 2017.

[50] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. Behavior sequence transformer for e-commerce recommendation in alibaba. In *Proceedings of the 1st international workshop on deep learning practice for high-dimensional sparse data*, pages 1–4, 2019.

[51] Xi Chen, Yining Wang, and Yu-Xiang Wang. Nonstationary stochastic optimization under l p, q-variation measures. *Operations Research*, 67(6): 1752–1765, 2019.

[52] Wang Chi Cheung, David Simchi-Levi, and Ruihao Zhu. Hedging the drift: Learning to optimize under nonstationarity. *Man. Sci.*, 68(3):1696–1713, 2022.

[53] Joel E Cohen and Uriel G Rothblum. Nonnegative ranks, decompositions, and factorizations of nonnegative matrices. *Linear Algebra and its Applications*, 190:149–168, 1993.

[54] Ashok Cutkosky. Artificial constraints and hints for unbounded online learning. In *Conference on Learning Theory*, pages 874–894. PMLR, 2019.

[55] Ashok Cutkosky and Kwabena Boahen. Online learning without prior information. In *Conference on learning theory*, pages 643–677. PMLR, 2017.

[56] Ashok Cutkosky and Francesco Orabona. Black-box reductions for parameter-free online learning in banach spaces. In *Conference On Learning Theory*, pages 1493–1529. PMLR, 2018.

[57] Nikhil R Devanur and Thomas P Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In *Proceedings of the 10th ACM conference on Electronic commerce*, pages 71–78, 2009.

[58] Nikhil R Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *Proceedings of the 12th ACM conference on Electronic commerce*, pages 29–38, 2011.

[59] Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Faster matchings via learned duals. *Advances in neural information processing systems*, 34:10393–10406, 2021.

[60] Ilan Doron-Arad, Ariel Kulik, and Pasin Manurangsi. Fine grained lower bounds for multidimensional knapsack. *arXiv preprint arXiv:2407.10146*, 2024.

[61] Ilya Dumer. Covering spheres with spheres. *Discrete & Computational Geometry*, 38:665–679, 2007.

[62] Paul Dütting, Silvio Lattanzi, Renato Paes Leme, and Sergei Vassilvitskii. Secretaries with advice. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 409–429, 2021.

[63] Paul Dütting, Evangelia Gergatsouli, Rojin Rezvan, Yifeng Teng, and Alexandros Tsigonias-Dimitriadis. Prophet secretary against the online optimal. *arXiv preprint arXiv:2305.11144*, 2023.

[64] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American economic review*, 97(1):242–259, 2007.

[65] Francis Y Edgeworth. The mathematical theory of banking. *Journal of the Royal Statistical Society*, 51(1):113–127, 1888.

[66] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

[67] Vivek F Farias, Andrew A Li, and Deeksha Sinha. Optimizing offer sets in sub-linear time. In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 639–640, 2020.

[68] Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S Mirrokni, and Cliff Stein. Online stochastic packing applied to display ad allocation. In *European Symposium on Algorithms*, pages 182–194. Springer, 2010.

[69] Mingsheng Fu, Hong Qu, Dagmawi Moges, and Li Lu. Attention based collaborative filtering. *Neurocomputing*, 311:88–98, 2018.

[70] Guillermo Gallego, Garud Iyengar, Robert Phillips, and Abhay Dubey. Managing flexible products on a network. *Available at SSRN 3567371*, 2004.

[71] Gregory A Godfrey and Warren B Powell. An adaptive, distribution-free algorithm for the newsvendor problem with censored demands, with applications to inventory and distribution. *Management Science*, 47(8):1101–1112, 2001.

[72] Negin Golrezaei, Patrick Jaillet, and Zijie Zhou. Online resource allocation with convex-set machine-learned advice. *arXiv preprint arXiv:2306.12282*, 2023.

[73] Vineet Goyal and Ramamoorthi Ravi. An fptas for minimizing a class of low-rank quasi-concave functions over a convex set. *Operations Research Letters*, 41(2):191–196, 2013.

[74] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.

[75] Anupam Gupta and Marco Molinaro. How the experts algorithm can help solve lps online. *Mathematics of Operations Research*, 41(4):1404–1431, 2016.

[76] Insu Han, Rajesh Jayaram, Amin Karbasi, Vahab Mirrokni, David P Woodruff, and Amir Zandieh. Hyperattention: Long-context attention in near-linear time. *arXiv preprint arXiv:2310.05869*, 2023.

[77] Botao Hao et al. Leveraging demonstrations to improve online learning: Quality matters. In *ICML*, 2023.

[78] Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

[79] Stephen J Hoch, Eric T Bradlow, and Brian Wansink. The variety of an assortment. *Marketing Science*, 18(4):527–546, 1999.

[80] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[81] Piao Hu, Jiashuo Jiang, Guodong Lyu, and Hao Su. Constrained online two-stage stochastic optimization: Algorithm with (and without) predictions. *arXiv preprint arXiv:2401.01077*, 2024.

[82] Chengpiao Huang and Kaizheng Wang. A stability principle for learning under non-stationarity. *arXiv preprint arXiv:2310.18304*, 2023.

[83] Jakob Huber, Sebastian Müller, Moritz Fleischmann, and Heiner Stuckenschmidt. A data-driven newsvendor problem: From data to decision. *European Journal of Operational Research*, 278(3):904–915, 2019.

[84] Woonghee Tim Huh and Paat Rusmevichientong. A nonparametric asymptotic analysis of inventory planning with censored demand. *Mathematics of Operations Research*, 34(1):103–123, 2009.

[85] Donald L Iglehart. The dynamic inventory problem with unknown demand distribution. *Management Science*, 10(3):429–440, 1964.

[86] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

[87] Klaus Jansen, Felix Land, and Kati Land. Bounding the running time of algorithms for scheduling and packing problems. *SIAM Journal on Discrete Mathematics*, 30(1):343–366, 2016.

[88] Jiashuo Jiang, Xiaocheng Li, and Jiawei Zhang. Online stochastic optimization with wasserstein based non-stationarity. *arXiv preprint arXiv:2012.06961*, 2020.

[89] Billy Jin and Will Ma. Online bipartite matching with advice: Tight robustness-consistency tradeoffs for the two-stage model. *Advances in Neural Information Processing Systems*, 35:14555–14567, 2022.

[90] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE, 2018.

[91] Samuel Karlin. Dynamic inventory policy with varying stochastic demands. *Management Science*, 6(3):231–258, 1960.

[92] Zohar S Karnin and Oren Anava. Multi-armed bandits: Competing with optimal sequences. *NIPS*, 29, 2016.

[93] Richard M Karp. *Reducibility among combinatorial problems*. Springer, 2010.

[94] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[95] N Bora Keskin and Assaf Zeevi. Chasing demand: Learning and earning in a changing environment. *Mathematics of Operations Research*, 42(2):277–307, 2017.

[96] N Bora Keskin, Xu Min, and Jing-Sheng Jeannette Song. The nonstationary newsvendor: Data-driven nonparametric learning. *Available at SSRN 3866171*, 2023.

[97] Thomas Kesselheim, Andreas Tönnis, Klaus Radke, and Berthold Vöcking. Primal beats dual on online packing lps in the random-order model. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 303–312, 2014.

[98] Enikő Kevi and Kim Tháng Nguyen. Primal-dual algorithms with predictions for online bounded allocation and ad-auctions problems. In *International Conference on Algorithmic Learning Theory*, pages 891–908. PMLR, 2023.

[99] Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

[100] Anton J Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on optimization*, 12(2):479–502, 2002.

[101] Peter Knees, Yashar Deldjoo, Farshad Bakhshandegan Moghaddam, Jens Adamczak, Gerard-Paul Leyson, and Philipp Monreal. Recsys challenge 2019: Session-based hotel recommendations. In *Proceedings of the Thirteenth ACM Conference on Recommender Systems*, RecSys '19, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6243-6/19/09. doi: 10.1145/3298689.3346974. URL `https://doi.org/10.1145/3298689.3346974`.

[102] Joohwan Ko and Andrew A Li. Modeling choice via self-attention. *arXiv preprint arXiv:2311.07607*, 2023.

[103] Ariel Kulik and Hadas Shachnai. There is no eptas for two-dimensional knapsack. *Information Processing Letters*, 110(16):707–710, 2010.

[104] Sumit Kunnumkal and Huseyin Topaloglu. Using stochastic approximation methods to compute optimal base-stock levels in inventory control problems. *Operations Research*, 56(3):646–664, 2008.

[105] Thom Lake, Sinead A Williamson, Alexander T Hawk, Christopher C Johnson, and Benjamin P Wing. Large-scale collaborative filtering with product embeddings. *arXiv preprint arXiv:1901.04321*, 2019.

[106] Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1859–1877. SIAM, 2020.

[107] Thomas Lavastida, Benjamin Moseley, R Ravi, and Chenyang Xu. Using predicted weights for ad delivery. In *SIAM Conference on Applied and Computational Discrete Algorithms (ACDA21)*, pages 21–31. SIAM, 2021.

[108] Daniel Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13, 2000.

[109] Sanghak Lee, Jaehwan Kim, and Greg M Allenby. A direct utility model for asymmetric complements. *Marketing Science*, 32(3):454–470, 2013.

[110] Retsef Levi, Robin O Roundy, and David B Shmoys. Provably near-optimal sampling-based policies for stochastic inventory control models. *Mathematics of Operations Research*, 32(4):821–839, 2007.

[111] Retsef Levi, Georgia Perakis, and Joline Uichanco. The data-driven newsvendor problem: New bounds and insights. *Operations Research*, 63(6):1294–1306, 2015.

[112] Chengxi Li, Yejing Wang, Qidong Liu, Xiangyu Zhao, Wanyu Wang, Yiqi Wang, Lixin Zou, Wenqi Fan, and Qing Li. Strec: Sparse transformer for sequential recommendations. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 101–111, 2023.

[113] Xiaocheng Li and Yinyu Ye. Online linear programming: Dual convergence, new algorithms, and regret bounds. *Operations Research*, 70(5):2948–2966, 2022.

[114] Xiaocheng Li, Chunlin Sun, and Yinyu Ye. Simple and fast algorithm for binary integer and online linear programming. *Advances in Neural Information Processing Systems*, 33:9412–9421, 2020.

[115] Xiaocheng Li, Chunlin Sun, and Yinyu Ye. The symmetry between arms and knapsacks: A primal-dual approach for bandits with knapsacks. In *International Conference on Machine Learning*, pages 6483–6492. PMLR, 2021.

[116] Shang Liu, Jiashuo Jiang, and Xiaocheng Li. Non-stationary bandits with knapsacks. *Advances in Neural Information Processing Systems*, 35:16522–16532, 2022.

[117] Liwan H Liyanage and J George Shanthikumar. A practical inventory control policy using operational statistics. *Operations Research Letters*, 33(4):341–348, 2005.

[118] William S Lovejoy. Myopic policies for some inventory models with uncertain demand distributions. *Management Science*, 36(6):724–738, 1990.

[119] Haihao Lu. "relative continuity" for non-lipschitz nonsmooth convex optimization using stochastic (or deterministic) mirror descent. *INFORMS Journal on Optimization*, 1(4):288–303, 2019.

[120] Haihao Lu, Robert M Freund, and Yurii Nesterov. Relatively smooth convex optimization by first-order methods, and applications. *SIAM Journal on Optimization*, 28(1):333–354, 2018.

[121] R Duncan Luce. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2012.

[122] Haipeng Luo, Chen-Yu Wei, Alekh Agarwal, and John Langford. Efficient contextual bandits in non-stationary worlds. In *Conference On Learning Theory*, pages 1739–1776. PMLR, 2018.

[123] Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *Journal of the ACM (JACM)*, 68(4):1–25, 2021.

[124] Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Online optimization with uncertain information. *ACM Transactions on Algorithms (TALG)*, 8 (1):1–29, 2012.

[125] Spyros Makridakis and Michele Hibon. The m3-competition: results, conclusions and implications. *International journal of forecasting*, 16(4):451–476, 2000.

[126] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4):1346–1364, 2022.

[127] Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018.

[128] Reza Yousefi Maragheh, Alexandra Chronopoulou, and James Mario Davis. A customer choice model with halo effect. *arXiv preprint arXiv:1805.01603*, 2018.

[129] Leigh McAlister. A dynamic attribute satiation model of variety-seeking behavior. *Journal of consumer research*, 9(2):141–150, 1982.

[130] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5):22–es, 2007.

[131] M Jeffrey Mei, Cole Zuber, and Yasaman Khazaeni. A lightweight transformer for next-item product recommendation. In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 546–549, 2022.

[132] Zakaria Mhammedi and Wouter M Koolen. Lipschitz and comparator-norm adaptivity in online learning. In *Conference on Learning Theory*, pages 2858–2887. PMLR, 2020.

[133] Vahab S Mirrokni, Shayan Oveis Gharan, and Morteza Zadimoghaddam. Simultaneous approximations for adversarial and stochastic online budgeted allocation. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1690–1701. SIAM, 2012.

[134] Shashi Mittal and Andreas S Schulz. An fptas for optimizing a class of low-rank functions over a polytope. *Mathematical Programming*, 141:103–120, 2013.

[135] Dmitrii Moor, Yi Yuan, Rishabh Mehrotra, Zhenwen Dai, and Mounia Lalmas. Exploiting sequential music preferences via optimisation-based sequencing. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 4759–4765, 2023.

[136] Marius Muja and David G Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE transactions on pattern analysis and machine intelligence*, 36(11):2227–2240, 2014.

[137] Andres Munoz and Sergei Vassilvitskii. Revenue optimization with approximate bid predictions. *Advances in Neural Information Processing Systems*, 30, 2017.

[138] Arkadij Semenovič Nemirovskij and David Borisovich Yudin. Problem complexity and method efficiency in optimization. 1983.

[139] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

[140] Trung Thanh Nguyen and Khaled Elbassioni. A ptas for a class of binary non-linear programs with low-rank functions. *Operations Research Letters*, 49(5):633–638, 2021.

[141] Francesco Orabona. Dimension-free exponentiated gradient. *Advances in Neural Information Processing Systems*, 26, 2013.

[142] Francesco Orabona and Ashok Cutkosky. Icml tutorial on parameter-free stochastic optimization. ICML, 2020. URL https://parameterfree.com/icml-tutorial/.

[143] Afshin Oroojlooyjadid, Lawrence V Snyder, and Martin Takáč. Applying deep learning to the newsvendor problem. *IISE Transactions*, 52(4):444–463, 2020.

[144] Jorge Pérez, Javier Marinković, and Pablo Barceló. On the turing completeness of modern neural network architectures. *arXiv preprint arXiv:1901.03429*, 2019.

[145] Warren Powell, Andrzej Ruszczyński, and Huseyin Topaloglu. Learning algorithms for separable approximations of discrete stochastic optimization problems. *Mathematics of Operations Research*, 29(4):814–836, 2004.

[146] Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ml predictions. *Advances in Neural Information Processing Systems*, 31, 2018.

[147] David J Rader Jr and Gerhard J Woeginger. The quadratic 0–1 knapsack problem with series–parallel support. *Operations Research Letters*, 30(3): 159–166, 2002.

[148] Omid Rafieian. Optimizing user engagement through adaptive ad sequencing. *Marketing Science*, 42(5):910–933, 2023.

[149] Dhruv Rohatgi. Near-optimal bounds for online caching with machine learned advice. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1834–1845. SIAM, 2020.

[150] Francisco JR Ruiz, Susan Athey, and David M Blei. Shopper. *The Annals of Applied Statistics*, 14(1):1–27, 2020.

[151] Clayton Sanford, Daniel Hsu, and Matus Telgarsky. One-layer transformers fail to solve the induction heads task. *arXiv preprint arXiv:2408.14332*, 2024.

[152] Clayton Sanford, Daniel J Hsu, and Matus Telgarsky. Representational strengths and limitations of transformers. *Advances in Neural Information Processing Systems*, 36, 2024.

[153] Herbert Scarf. Bayes solutions of the statistical inventory problem. *The annals of mathematical statistics*, 30(2):490–508, 1959.

[154] Herbert Scarf, K Arrow, S Karlin, and P Suppes. The optimality of (s, s) policies in the dynamic inventory problem. *Optimal pricing, inflation, and the cost of price adjustment*, pages 49–56, 1960.

[155] A Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[156] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[157] Alexander Shapiro. Monte carlo sampling methods. *Handbooks in operations research and management science*, 10:353–425, 2003.

[158] Matthew Streeter and H Brendan McMahan. No-regret algorithms for unconstrained online convex optimization. *arXiv preprint arXiv:1211.2260*, 2012.

[159] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. *Advances in neural information processing systems*, 28, 2015.

[160] Kalyan T Talluri and Garrett J Van Ryzin. *The theory and practice of revenue management*, volume 68. Springer Science & Business Media, 2006.

[161] Richard Taylor. Approximation of the quadratic knapsack problem. *Operations Research Letters*, 44(4):495–497, 2016.

[162] Sean J Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.

[163] James T Treharne and Charles R Sox. Adaptive inventory control for nonstationary demand and partial information. *Management Science*, 48(5):607–624, 2002.

[164] Pravin M Vaidya. A new algorithm for minimizing convex functions over convex sets. *Mathematical programming*, 73(3):291–341, 1996.

[165] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[166] Stephen A Vavasis. On the complexity of nonnegative matrix factorization. *SIAM journal on optimization*, 20(3):1364–1377, 2010.

[167] Erik Vee, Sergei Vassilvitskii, and Jayavel Shanmugasundaram. Optimal online assignment with forecasts. In *Proceedings of the 11th ACM conference on Electronic commerce*, pages 109–118, 2010.

[168] Jean-Louis Verger-Gaugry. Covering a ball with smaller equal balls in $\mathbb{R}^n$. *Discrete & Computational Geometry*, 33:143–155, 2005.

[169] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[170] Hanzhao Wang, Xiaocheng Li, and Kalyan Talluri. Transformer choice net: A transformer neural network for choice prediction. *arXiv preprint arXiv:2310.08716*, 2023.

[171] Shoujin Wang, Liang Hu, Longbing Cao, Xiaoshui Huang, Defu Lian, and Wei Liu. Attention-based transactional context embedding for next-item recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[172] Yu-Xiong Wang and Yu-Jin Zhang. Nonnegative matrix factorization: A comprehensive review. *IEEE Transactions on knowledge and data engineering*, 25(6):1336–1353, 2012.

[173] Colin Wei, Yining Chen, and Tengyu Ma. Statistically meaningful approximation: a case study on approximating turing machines with transformers. *Advances in Neural Information Processing Systems*, 35:12071–12083, 2022.

[174] Timo Wilm, Philipp Normann, Sophie Baumeister, and Paul-Vincent Kobow. Scaling session-based transformer recommendations using optimized negative sampling and loss functions. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1023–1026, 2023.

[175] Peter R Winters. Forecasting sales by exponentially weighted moving averages. *The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries*, 6(3):324–342, 1960.

[176] Yihong Wu. Lecture 14: Covering and packing numbers. `http://www.stat.yale.edu/~yw562/teaching/598/lec14.pdf`, 2017. STAT 598: High Dimensional Statistics, Yale University.

[177] Boming Yang, Dairui Liu, Toyotaro Suzumura, Ruihai Dong, and Irene Li. Going beyond local: Global graph-enhanced personalized news recommendations. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 24–34, 2023.

[178] Tianbao Yang, Lijun Zhang, Rong Jin, and Jinfeng Yi. Tracking slowly moving clairvoyant: Optimal dynamic regret of online learning with true and noisy gradient. In *International Conference on Machine Learning*, pages 449–457. PMLR, 2016.

[179] Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint arXiv:1912.10077*, 2019.

[180] Lijun Zhang, Shiyin Lu, and Zhi-Hua Zhou. Adaptive online learning in dynamic environments. *NIPS*, 31, 2018.

[181] Luhao Zhang, Jincheng Yang, and Rui Gao. Optimal robust policy for feature-based newsvendor. *Management Science, Forthcoming*, 2023.

[182] Zhi Zheng, Ying Sun, Xin Song, Hengshu Zhu, and Hui Xiong. Generative learning plan recommendation for employees: A performance-aware reinforcement learning approach. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 443–454, 2023.

Appendix A

APPENDIX FOR CHAPTER 2

A.1 Proofs in Section 2.2

First, we provide some previous results on ϵ-ANN algorithms. [8] gives an ϵ-ANN algorithm by using Locality-Sensitive Hashing:

Proposition 13 (Corollary 1 in [8]). *For any given $\epsilon > 0$, ϵ-ANN on the unit sphere $\mathbb{S}^{d-1} \subset \mathbb{R}^d$ can be solved with expected amortized runtime $O(dn^{\gamma(\epsilon)})$, where $\gamma(\epsilon) = \frac{1}{1+c\epsilon} + o(1)$, where $c > 0$ is a universal constant.*

[2] gives an ϵ-ANN algorithm by applying fast Johnson-Lindenstrauss transform (FJLT):

Proposition 14 (Theorem 1 in [2]). *For any given $\epsilon > 0$, ϵ-ANN on the unit sphere $\mathbb{S}^{d-1} \subset \mathbb{R}^d$ can be solved with expected amortized runtime $O\left(d\log(d) + \epsilon^{-3}\log^2(n)\right)$.*

[12] gives an ϵ-ANN algorithm by using a tree-based data structure:

Proposition 15 (Theorem 3.1 in [12]). *For any given $\epsilon > 0$, the ϵ-ANN on the unit sphere $\mathbb{S}^{d-1} \subset \mathbb{R}^d$ can be solved with expected amortized runtime $O(\log n + 1/\epsilon^d)$.*

Proof of Lemma 1. We prove that this can be done by applying a θ-Approximate Nearest Neighbor (θ-ANN) algorithm from [2]. Given a set of n "data points" $P \subset \mathbb{R}^{d_v}$, the goal of the θ-Approximate Near Neighbor algorithm is to build a data structure that, given a query $q \in \mathbb{R}^{d_v}$, returns a data point $p \in P$ such that $\|q - p\|_2 \leq (1+\theta)\|q - p'\|_2$ for all $p' \in P$.

Lemma 4 (Theorem 1 in [2]). *Algorithm 1 in [2] solves the θ-ANN problem in query time $O(d\log(d) + \epsilon^{-3}\log^2(n))$.*

Moreover, the Algorithm 1 in [2] can be easily modified to return k data points $p_1, \ldots, p_k \in P$ such that $\|q - p_i\|_2 \leq (1+\theta)\|q - p'\|_2$ for all $i = 1, \ldots, k$ and all $p' \in P \setminus \{p_1, \ldots, p_k\}$. This can be done by running their Algorithm 1 for k times, where each time we ignore the data points that have already been returned in previous runs. More specifically, in their Algorithm 1 we change the 'else if

$D_T(x, p') > (1 + 1/2)k''$ to 'else if $D_T(x, p') > (1 + 1/2)k'$ or p' has been returned in previous rounds', which gives our desired performance.

Because the guarantee of Lemma 1 is a multiplicative guarantee on the ℓ_2 distance, below we show how to convert it into an additive guarantee on the inner products. We first reduce $v_1, \ldots, v_n$ and u to the unit sphere. Let $v_{\max} = \max_i \|v_i\|_2$. Let

$$v_i' = \left(\sqrt{v_{\max} - \|v_i\|_2^2}, v_i^\top\right)^\top / v_{\max} \in \mathbb{R}^{d+1},$$

for each $i = 1, \ldots, n$, where we append a scalar to v_i and rescale the vector. Let $u' = (0, u^\top)^\top / \|u\|_2$. Then $v_i', u' \in \mathbb{S}^d$. Moreover,

$$\begin{aligned} \|u' - v_i'\|_2^2 &= \|u'\|_2^2 + \|v_i'\|_2^2 - 2v_i'^\top u' \\ &= 2 - 2v_i^\top u / (v_{\max}||u||_2). \end{aligned}$$

Therefore $v_i^\top u = \frac{v_{\max}||u||_2}{2}(2 - \|u' - v_i'\|_2^2)$. Hence if $|\|u' - v_i'\|_2^2 - \|u' - v_j'\|_2^2| \leq \frac{2\epsilon}{v_{\max}||u||_2}$, then $|v_i^\top u - v_j^\top u| \leq \delta$. Hence, we can run the Algorithm 1 in [2] with inputs $v_1', \ldots, v_n', u'$, and $\theta = \frac{8\sqrt{2}\epsilon}{(Vu)_{\max}}$, which outputs k indices $i_1, \ldots, i_k$ such that $\|u' - v_{i_j}'\|_2 \leq (1 + \theta)\|u' - v_{i_j^*}'\|_2$ for each $j = 1, \ldots, k$. Then

$$\begin{aligned} |\|u' - v_{i_j}'\|_2^2 - \|u' - v_{i_j^*}'\|_2^2| &= \|u' - v_{i_j^*}'\|_2^2 - \|u' - v_{i_j}'\|_2^2 \\ &= (\|u' - v_{i_j^*}'\|_2 + \|u' - v_{i_j}'\|_2)(\|u' - v_{i_j^*}'\|_2 - \|u' - v_{i_j}'\|_2) \\ &\leq 4(\|u' - v_{i_j^*}'\|_2 - \|u' - v_{i_j}'\|_2) \\ &\leq 4\theta\|u' - v_{i_j^*}'\|_2 \\ &\leq 4\sqrt{2}\epsilon \\ &= \frac{2\epsilon}{v_{\max}||u||_2}, \end{aligned}$$

where the first equality follows from the definition of i_j^*, the second equality and the second inequality follows since $v_{i_j}, v_{i_j^*}, u' \in \mathbb{S}^{d_v}$. Therefore we have $v_{i_j}^\top u \geq v_{i_j^*}^\top u - \epsilon$ for each $j = 1, \ldots, k$ as desired. ■

Proof of Proposition 2. The algorithm ALG operates as follows:

1. Partition the items according to their reward functions f_i.
2. For each partition, apply the given ϵ-Approximate k-Nearest Neighbor algorithm to identify k items that are collectively among the most attractive to the user within that partition.

Algorithm 8: ϵ-Approximate k-Nearest Neighbor in Lemma 1

PREPROCESS

Input: $v_1, \ldots, v_n \in \mathbb{R}^d$ and $\epsilon > 0$;

$v_{\max} \leftarrow \max_i \|v_i\|_2$;

$v_i' \leftarrow \left(\sqrt{v_{\max} - \|v_i\|_2^2}, v_i\right) / v_{\max}$ for each $i = 1, \ldots, n$;

$\theta \leftarrow 8\sqrt{2}\epsilon/c$, where c is an upper bound on $v_{\max}||u||_2$ for all possible u;

Run PREPROCESS of the θ-ANN algorithm in [2] with inputs $v_1', \ldots, v_n' \in \mathbb{R}^d$ and $\theta > 0$.

QUERY

Input: $u \in \mathbb{R}^d$;

$u' \leftarrow (0, u)/\|u\|_2$;

$j \leftarrow 1$;

while $j \leq k$ **do**

  Run PREPROCESS of the θ-ANN algorithm in [2] with input u', with the modification of changing 'else if $D_T(x, p') > (1 + 1/2)k'$' to 'else if $D_T(x, p') > (1 + 1/2)k'$ or $p' \notin \{v_{i_{j'}}\}_{j'<j}$' ;

  $i_j \leftarrow$ the index of the output of the θ-ANN algorithm in [2];

Return $i_1, \ldots, i_k$.

3. Evaluate the rewards of all such candidate items across partitions and select the top-k items with the highest overall reward.

We analyze the performance of ALG. Let $\{S_1, \ldots, S_\tau\}$ be a partition of the index set $[n]$ such that $f_i = f_j$ for every $\tau' \in [\tau]$ and $i, j \in S_{\tau'}$. We first construct an index set $J_{\tau'} \subset S_{\tau'}$ for each $\tau' \in [\tau]$, and then combine them to obtain the set of all candidate items I.

For each index set $S_{\tau'}$, we only choose k indices out of it to include in $J_{\tau'}$, namely the k indices that are approximately the k highest indices in $\{v_i^\top u\}_{i \in S_{\tau'}}$.[1] Specifically, for each $\ell' \in [\ell]$, we run the given ϵ-Approximate k-Nearest Neighbor oracle with given set of points $\bigcup_{i \in S_{\tau'}} \{V_i\} \subset \mathbb{R}^d$, and query u, numbers k and ϵ as inputs. We let $J_{\tau'}$ be the collection of all output indices for each $\ell' \in [\ell]$. Then $|J_{\tau'}| \leq k$ for each τ'. Let $I = \bigcup_{\tau' \in [\tau]} J_{\tau'}$, then $|I| \leq \tau k$.

Let $S^* = \{i_1^*, \ldots, i_k^*\}$ be the optimal solution to Problem (Pure Embedding) (for simplicity we assume $|S^*| = k$, and other cases can be handled similarly). We show that $\text{ALG}_{\text{Pure Embedding}} \geq (1 - g(\epsilon))\text{OPT}_{\text{Pure Embedding}} - kh(\epsilon)$. For each $m = 1, \ldots, k$, let i_m be the index such that i_m and i_m^* are in the same $S_{\tau'}$ and

[1] For simplicity we assume $|S_{\tau'}| \geq k$. Otherwise we simply choose all indices in $S_{\tau'}$.

$v_{i_m}^\top u \geq v_{i_m^*}^\top u - \epsilon$. Then $f_{i_m} = f_{i_m^*}$. Let $S = \{i_1, \ldots, i_k\}$. Because ALG returns the k-highest value in $\bigcup_{i \in I}\{f_i(v_i^\top u)\}$, we have

$$\begin{aligned}
\text{ALG}_{\text{Pure Embedding}} &\geq \sum_{i \in S} f_i(v_i^\top u) \\
&\geq \sum_{i \in S^*} f_i(v_i^\top u - \epsilon) \\
&\geq (1 - g(\epsilon)) \sum_{i \in S^*} f_i(v_i^\top u) - kh(\epsilon) \\
&= (1 - g(\epsilon))\text{OPT}_{\text{Pure Embedding}} - kh(\epsilon).
\end{aligned}$$

Finally we analyze the expect amortized runtime of ALG. The expect amortized runtime of constructing each $J_{\tau'}$ is k-ANN$(|S_{\tau'}|, d, k, \epsilon)$. Because $|I| \leq \tau k$, the runtime of finding the k-highest value in $\bigcup_{i \in I}\{f_i(v_i^\top u)\}$ is $\log_2(\tau k)$. Therefore the expect amortized runtime of ALG is

$$\sum_{\tau'=1}^{\tau} k\text{-ANN}(|S_{\tau'}|, d, k, \epsilon) + \log_2(\tau k) \leq \tau \cdot k\text{-ANN}\left(\frac{n}{\tau}, d, k, \epsilon\right) + \log_2(\tau k).$$

The inequality follows since $\sum_{\tau'=1}^{\tau} |S_{\tau'}| = n$ and k-ANN(n, d, k, ϵ) is concave in n.

■

A.2 Proofs in Section 2.2

Proof of Proposition 3 (Model 1). We construct a simple transformer with the following parameters:

- The input dimension is set to

  $$N = nk + nkd + 1 + nk,$$

  and is indexed as follows. Define index sets

  $$\mathcal{I} = \{(i,t) : i \in [n],\, t \in [k]\}, \quad \mathcal{M} = \{(j,m,a) : j \in [n],\, m \in [k],\, a \in [d]\},$$

  $$\mathcal{D} = \{\odot\} \quad \text{(one dummy row)}, \qquad \mathcal{B} = \{(i,t)^{\text{base}} : i \in [n],\, t \in [k]\},$$

  and order the N rows of Q and K as $[\mathcal{I}; \mathcal{M}; \mathcal{D}; \mathcal{B}]$.

  Before proceeding, we give intuitions on how these index sets are used:

  - $\mathcal{I} = \{(i,t)\}$: These rows are the only rows with nonzero queries Q. Each of them represents the effects of the past items to the current item.

- $\mathcal{M} = \{(j, m, a)\}$: These rows have non-zero keys K and scalar values V. They have zero queries Q. They encode the item i_m at position m and its similarity embedding $x_{i_m,a}$.
- $\mathcal{D} = \{\odot\}$: This is a dummy row that dominates the softmax denominator, so that the softmax vector behaves like division by a constant.
- $\mathcal{B} = \{(i, t)^{\text{base}}\}$: These rows encode the base utility $\hat{u}_i$ of each item i.

- The embedding dimension is set to $d_{kq} = k + d + 2$. This is split into a *position* block of length k, a *component* block of length d, a single *dummy* column (denoted Δ), and a single *base* column (denoted Θ).

- Let $M > 0$ be a sufficiently large constant and $b_0 \in \mathbb{R}$ a fixed constant. Later we will take M large enough so that the simple transformer approximates $g(S, i_t)$ to arbitrary precision.

- For each position $t \in [k]$, define the row vector $r_t \in \mathbb{R}^{1\times k}$ by

$$(r_t)_m = \begin{cases} \log \lambda_{t-m}, & m < t, \\ -M, & m \geq t, \end{cases} \qquad m \in [k],$$

  i.e.

$$r_t = \big[\, \log \lambda_{t-1},\ \log \lambda_{t-2},\ \cdots\ \log \lambda_1,\ \underbrace{-M,\ -M,\ \cdots,\ -M}_{k-t+1} \,\big].$$

  For each $t \in [k]$, let $R_t \in \mathbb{R}^{n\times k}$ be the matrix with all rows equal to r_t:

$$R_t = \begin{bmatrix} r_t \\ r_t \\ \vdots \\ r_t \end{bmatrix} \quad (n \text{ rows}).$$

  Then the *position block* of the query matrix is the vertical stacking of these k blocks:

$$Q_{\text{pos}} = \begin{bmatrix} R_1 \\ R_2 \\ \vdots \\ R_k \end{bmatrix} \in \mathbb{R}^{(nk)\times k}.$$

Next, define the $n \times d$ matrix G collecting the (component-wise) logarithms of the similarity embeddings $x_i = (x_{i,1}, \ldots, x_{i,d}) \in S^{d-1}$:

$$G = \begin{bmatrix} (\log x_1)^\top \\ (\log x_2)^\top \\ \vdots \\ (\log x_n)^\top \end{bmatrix} = \begin{bmatrix} \log x_{1,1} & \log x_{1,2} & \cdots & \log x_{1,d} \\ \log x_{2,1} & \log x_{2,2} & \cdots & \log x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ \log x_{n,1} & \log x_{n,2} & \cdots & \log x_{n,d} \end{bmatrix}.$$

The *component block* of the query matrix is k identical copies of G stacked vertically:

$$Q_{\text{cmp}} = \begin{bmatrix} G \\ G \\ \vdots \\ G \end{bmatrix} \in \mathbb{R}^{(nk) \times d}.$$

Let $\mathbf{1}_p \in \mathbb{R}^{p \times 1}$ be the all-ones column, and $0_p \in \mathbb{R}^{p \times 1}$ be the all-zeross column. Define the *dummy* and *base* query columns as

$$Q_\Delta = \begin{bmatrix} M^3 \, \mathbf{1}_{nk} \\ 0_{nkd} \\ 0 \\ 0_{nk} \end{bmatrix}, \qquad Q_\Theta = \begin{bmatrix} b_0 \, \mathbf{1}_{nk} \\ 0_{nkd} \\ 0 \\ 0_{nk} \end{bmatrix} \in \mathbb{R}^{N \times 1}.$$

Putting the query blocks together, we define Q to be

$$Q = \begin{bmatrix} Q_{\text{pos}} & Q_{\text{cmp}} & Q_\Delta & Q_\Theta \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{N \times (k+d+2)}.$$

- The *position block* of the key matrix is defined as

$$K_{\text{pos}} = \begin{bmatrix} \mathbf{1}_{nd} & 0 & \cdots & 0 \\ 0 & \mathbf{1}_{nd} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{1}_{nd} \end{bmatrix} \in \mathbb{R}^{(nkd) \times k},$$

i.e., a block-diagonal matrix with k diagonal blocks, each block the column $\mathbf{1}_{nd}$.

Let $I_d \in \mathbb{R}^{d\times d}$ be the identity matrix. The *component block* of the key matrix is defined as

$$K_{\text{cmp}} \;=\; \begin{bmatrix} I_d \\ I_d \\ \vdots \\ I_d \end{bmatrix} \in \mathbb{R}^{(nkd)\times d},$$

Thus, in the block associated with position m and item j, the d consecutive rows equal the identity I_d.

The *dummy* and *base* key columns are

$$K_\Delta \;=\; \begin{bmatrix} 0_{nk} \\ 0_{nkd} \\ 1 \\ 0_{nk} \end{bmatrix}, \qquad K_\Theta \;=\; \begin{bmatrix} 0_{nk} \\ 0_{nkd} \\ 0 \\ \mathbf{1}_{nk} \end{bmatrix} \in \mathbb{R}^{N\times 1}.$$

Putting the key blocks together, we define K to be

$$K \;=\; \begin{bmatrix} 0 & 0 & 0 & 0 \\ K_{\text{pos}} & K_{\text{cmp}} & 0 & 0 \\ 0 & 0 & K_\Delta & 0 \\ 0 & 0 & 0 & K_\Theta \end{bmatrix} \in \mathbb{R}^{N\times(k+d+2)}.$$

- Let $v \in \mathbb{R}^{nd\times 1}$:

$$v \;=\; \begin{bmatrix} x_1^\top \\ x_2^\top \\ \vdots \\ x_n^\top \end{bmatrix} \;=\; \begin{bmatrix} x_{1,1} \\ \vdots \\ x_{1,d} \\ x_{2,1} \\ \vdots \\ x_{n,d} \end{bmatrix}.$$

We define V (one scalar per row) by placing zeros on $\mathcal{I}$ and $\mathcal{D}$, the item–component entries on $\mathcal{M}$, and the base utilities on $\mathcal{B}$:

$$V \;=\; \begin{bmatrix} 0_{nk} \\ \beta\, e^{M^3}\, v \\ 0 \\ e^{M^3}\left(\frac{1}{\beta}\log\hat{u}_1, \ldots, \frac{1}{\beta}\log\hat{u}_n\right)^\top \\ \vdots \\ e^{M^3}\left(\frac{1}{\beta}\log\hat{u}_1, \ldots, \frac{1}{\beta}\log\hat{u}_n\right)^\top \end{bmatrix} \in \mathbb{R}^{N\times 1},$$

where $e^{M^3}\left(\frac{1}{\beta}\log\hat{u}_1,\ldots,\frac{1}{\beta}\log\hat{u}_n\right)^\top$ is repeated k times. Set $u = 1$ so that $V_i^\top u = V_i$ for each row i.

- We set $f_i(x) = \exp(\beta x)$ for all $i \in [nk + nkd + 1 + nk]$.
- For a length-k sequence $S = (i_1,\ldots,i_k)$, let $S' \subset [nk + nkd + 1 + nk] = [\mathcal{I}; \mathcal{M}; \mathcal{D}; \mathcal{B}]$ such that S' contains $\{(i_t,t) : t \in [k]\} \subset [\mathcal{I}]$ from the first set of rows, $\{(i_m,m,a) : m \in [k],\ a \in [d]\} \subset [\mathcal{M}]$ from the second set of rows, the dummy row $\mathcal{D}$, and $\{(i_t,t)^{\text{base}} : t \in [k]\} \subset \mathcal{B}$ from the set of base rows.

We show that the simple transformer above approximates $g(S,i_t)$ to arbitrary precision. The intuition of our construction is given below:

- Q encodes lags (λ) and similarity embeddings ($\log x_i$).
- K encodes positions (t) in the sequence.
- V encodes similarity embeddings ($\log x_i$) and base utilities ($\hat{u}_i$).
- The dummy row makes softmax act like a constant divider.

Fix a length-k sequence $S = (i_1,\ldots,i_k)$. We first compute $X_{S'}Q(X_{S'}K)^\top$ for each $(i_t,t) \in S'$:

$$\begin{aligned}
\text{(context)}\quad & (X_{S'}Q(X_{S'}K)^\top)_{(i_t,t),(i_m,m,a)} &&= r_t(m) + \log x_{i_t,a}\\
\text{(dummy)}\quad & (X_{S'}Q(X_{S'}K)^\top)_{(i_t,t),\odot} &&= M^3,\\
\text{(base)}\quad & (X_{S'}Q(X_{S'}K)^\top)_{(i_t,t),(i_t,t)^{\text{base}}} &&= 0.
\end{aligned}$$

Hence the denominator of the softmax operation on row $X_{S'}Q(X_{S'}K)^\top$ is

$$Z_t \;=\; e^{M^3} + 1 + \sum_{m<t}\sum_{a=1}^{d}\lambda_{t-m}x_{i_t,a} + \sum_{m\geq t}\sum_{a=1}^{d}e^{-M}x_{i_t,a}.$$

Define the following (finite) constants

$$\Lambda_t = \sum_{m<t}\lambda_{t-m}, \qquad S_t^{(\leq)} = \sum_{m<t}\sum_{a}\lambda_{t-m}x_{i_t,a}\,, \qquad S_t^{(\geq)} = \sum_{m\geq t}\sum_{a}e^{-M}x_{i_t,a}\,.$$

Set

$$\delta_t = e^{-M^3} + e^{-M^3}S_t^{(\leq)} + e^{-M^3}S_t^{(\geq)}.$$

Then

$$Z_t = e^{M^3}(1+\delta_t).$$

Therefore the attention weights on row (i_t, t) are

$$\begin{aligned}\mathrm{softmax}(X_{S'}Q(X_{S'}K)^\top)_{(i_t,t),(i_m,m,a)} &= \frac{\exp(r_t(m)+\log x_{i_t,a})}{Z_t} \\ &= \frac{\lambda_{t-m}\, x_{i_t,a}}{e^{M^3}(1+\delta_t)} \quad (m<t),\end{aligned}$$

and

$$\mathrm{softmax}(X_{S'}Q(X_{S'}K)^\top)_{(i_t,t),(i_t,t)^{\mathrm{base}}} = \frac{\exp(0)}{Z_t} = \frac{e^{-M^3}}{1+\delta_t}.$$

Recall that

$$V_{(i_m,m,a)} = e^{M^3}\, x_{i_m,a}, \qquad V_{\odot} = 0, \qquad V_{(i_t,t)^{\mathrm{base}}} = e^{M^3}\cdot\frac{1}{\beta}\log\hat{u}_{i_t}.$$

So we have

$$\begin{aligned}\mathrm{SA}_{Q,K,V}(X_{S'})_{(i_t,t)} &= \sum_{m<t}\sum_{a=1}^{d}\mathrm{softmax}(X_{S'}Q(X_{S'}K)^\top)_{(i_t,t),(i_m,m,a)}\, V_{(i_m,m,a)} \\ &\quad + \mathrm{softmax}(X_{S'}Q(X_{S'}K)^\top)_{(i_t,t),(i_t,t)^{\mathrm{base}}}\, V_{(i_t,t)^{\mathrm{base}}} \\ &= \frac{1}{1+\delta_t}\left(\sum_{m<t}\sum_{a=1}^{d}\frac{\lambda_{t-m}x_{i_t,a}}{e^{M^3}}\, e^{M^3}x_{i_m,a} + e^{-M^3}\, e^{M^3}\cdot\frac{1}{\beta}\log\hat{u}_{i_t}\right) \\ &= \frac{1}{1+\delta_t}\left(\sum_{m<t}\lambda_{t-m}\sum_{a=1}^{d}x_{i_t,a}x_{i_m,a} \;+\; \frac{1}{\beta}\log\hat{u}_{i_t}\right) \\ &= \frac{1}{1+\delta_t}\left(\sum_{\ell=1}^{t-1}\lambda_\ell\, x_{i_t}^\top x_{i_{t-\ell}} \;+\; \frac{1}{\beta}\log\hat{u}_{i_t}\right).\end{aligned}$$

Fix any $\epsilon > 0$. Since $S_t^{(\le)} \le \Lambda_t \le \Lambda_{k-1}$ and $S_t^{(\ge)} \le d\, e^{-M}$, we can set M large enough so that

$$e^{-M^3}\left(1+\Lambda_{k-1}\right) + d\, e^{-(M^3+M)} \le \epsilon,$$

which gives $\delta_t \le \epsilon$ for every t. Finally, we have

$$\begin{aligned}\mathcal{T}_{Q,K,V,f_1,\ldots,f_n,u}(X_{S'})_{(i_t,t)} &= f_{(i_t,t)}\left(\mathrm{SA}_{Q,K,V}(X_{S'})_{(i_t,t)}\right) \\ &= \exp\left(\frac{\beta}{1+\delta_t}\sum_{\ell=1}^{t-1}\lambda_\ell\, x_{i_t}^\top x_{i_{t-\ell}} + \frac{1}{1+\delta_t}\log\hat{u}_{i_t}\right) \\ &= \hat{u}_{i_t}^{\frac{1}{1+\delta_t}}\ \exp\left(\frac{\beta}{1+\delta_t}\sum_{\ell=1}^{t-1}\lambda_\ell\, x_{i_t}^\top x_{i_{t-\ell}}\right).\end{aligned}$$

Hence, as $M \to \infty$ (so $\delta_t \to 0$ uniformly in t), we have

$$(1-\epsilon)g(S,i_t) \le \mathcal{T}_{Q,K,V,f_1,\ldots,f_n,u}(X_{S'})_{(i_t,t)} \le \mathcal{T}_{Q,K,V,f_1,\ldots,f_n,u}(X_{S'})_{(i_t,t)}.$$

Therefore the simple transformer approximate $g(S,i_t)$ to arbitrary precision.

■

Proof of Proposition 3 (Model 2). We construct a simple transformer with three self-attention heads. The first head and the second head represent the complementarity effects and the substitution effects, respectively. They have input dimensions $n+1$, output dimension 1, and embedding dimension $n+1$. The third head represents the base utility of each item. It has input dimensions n, output dimension d, and embedding dimension n. Note that this multi-head construction can be equivalently represented as a single-head simple transformer by arranging the Q, K, V, and u matrices for all three heads into block form. However, for clarity, we present the proof by describing each head separately.

Let $A, B \in \mathbb{R}_+^{n\times n}$ be two matrices with positive entries such that $H_{ij} = \exp(A_{ij}) - \exp(B_{ij})$ for every $i, j \in [n]$. Let M be a sufficiently large constant. Later we will set M to be large enough so that the simple transformer approximates $g(S,i)$ to arbitrary precision.

Head 1. The first self-attention head has the following parameters:

- $Q^{(1)}, K^{(1)} \in \mathbb{R}^{(n+1)\times(n+1)}$ are set such that

$$Q^{(1)}(K^{(1)})^\top = \left[\begin{array}{c|c} A & \begin{matrix} M \\ \vdots \\ M \end{matrix} \\ \hline 0\cdots\cdots 0 & M \end{array}\right].$$

- $V^{(1)} \in \mathbb{R}^{n+1}$ is set to $V_i^{(1)} = 1$ for each $i \in [n]$ and $V_{n+1}^{(1)} = 0$. Set $u^{(1)} = 1$ so that $(V_i^{(1)})^\top u^{(1)} = V_i^{(1)}$.
- Set $f_i^{(1)}(x) = \exp(M)x$ for every $i \in [n^2+1]$.
- For a subset $S \subset [n]$, we set $S^{(1)} \subset [n+1]$ where $S^{(1)} = S \cup \{n+1\}$.

We have

$$\begin{aligned}
&\operatorname{softmax}\big(X_{S^{(1)}}Q^{(1)}(X_{S^{(1)}}K^{(1)})^\top\big)_{ij} \\
&= \begin{cases}
\dfrac{\exp(A_{ij})}{\sum_{j\in S}\exp(A_{ij}) + \exp(M) + (n-|S|)}, & i \in S,\ j \in S, \\
\dfrac{\exp(M)}{\sum_{j\in S}\exp(A_{ij}) + \exp(M) + (n-|S|)}, & i \in S,\ j = n+1, \\
\dfrac{1}{\sum_{j\in S}\exp(A_{ij}) + \exp(M) + (n-|S|)}, & i \in S,\ j \in [n]\setminus S, \\
\dfrac{1}{n+\exp(M)}, & i = n+1,\ j \in [n], \\
\dfrac{\exp(M)}{n+\exp(M)}, & i = n+1,\ j = n+1, \\
\dfrac{1}{n+1}, & i \in [n]\setminus S,\ j \in [n{+}1].
\end{cases}
\end{aligned}$$

Therefore, for every $i \in S$, we have

$$\mathrm{SA}_{Q^{(1)},K^{(1)},V^{(1)}}(X_{S^{(1)}})_i^\top u^{(1)} = \frac{\sum_{j\in S}\exp(A_{ij})}{\sum_{j\in S}\exp(A_{ij}) + \exp(M) + (n-|S|)}.$$

For any given $\epsilon > 0$, we can take M to be sufficiently large such that

$$\frac{\sum_{j\in S}\exp(A_{ij}) - \frac{\epsilon}{2}}{\exp(M)} \le \mathrm{SA}_{Q^{(1)},K^{(1)},V^{(1)}}(X_{S^{(1)}})_i^\top u^{(1)} \le \frac{\sum_{j\in S}\exp(A_{ij})}{\exp(M)}.$$

Finally, we get

$$\sum_{j\in S}\exp(A_{ij}) - \frac{\epsilon}{3} \le \mathcal{T}_{Q^{(1)},K^{(1)},V^{(1)},f_1^{(1)},\ldots,f_n^{(1)},u^{(1)}}(X_{S^{(1)}})_i \le \sum_{j\in S}\exp(A_{ij}).$$

Head 2. The second self-attention head has exactly the same parameters, expect we replace A with B and we flip the sign of f_i:

- $Q^{(2)}, K^{(2)} \in \mathbb{R}^{(n+1)\times(n+1)}$ are set such that

$$Q^{(2)}(K^{(2)})^\top = \left[\begin{array}{ccc|c} & & & M \\ & B & & \vdots \\ & & & M \\ \hline 0 & \cdots & 0 & M \end{array}\right].$$

- $V^{(2)} \in \mathbb{R}^{n+1}$ is set to $V_i^{(2)} = 1$ for each $i \in [n]$ and $V_{n+1}^{(2)} = 0$. Set $u^{(2)} = 1$ so that $(V_i^{(1)})^\top u^{(2)} = V_i^{(2)}$.

- Set $f_i^{(2)}(x) = -\exp(M)x$ for every $i \in [n^2+1]$.
- For a subset $S \subset [n]$, we set $S^{(2)} \subset [n+1]$ where $S^{(2)} = S \cup \{n+1\}$.

Then, similar to head 1, we get

$$-\sum_{j\in S}\exp(B_{ij}) \le \mathcal{T}_{Q^{(2)},K^{(2)},V^{(2)},f_1^{(2)},\ldots,f_n^{(2)},u^{(2)}}(X_{S^{(2)}})_i \le -\sum_{j\in S}\exp(B_{ij}) + \frac{\epsilon}{3}.$$

Head 3. The third self-attention head has the following parameters:

- $Q^{(3)}, K^{(3)} \in \mathbb{R}^{n\times n}$ are set such that

$$Q^{(3)}(K^{(3)})^\top = \begin{bmatrix} 0 & -M & \cdots & -M \\ -M & 0 & \cdots & -M \\ \vdots & \vdots & \ddots & \vdots \\ -M & -M & \cdots & 0 \end{bmatrix}.$$

- $V^{(3)} \in \mathbb{R}^{n\times d}$ is set to $V_i^{(3)} = \hat{v}_i$ for each $i \in [n]$.
- $u^{(3)} \in \mathbb{R}^d$ is set to $u^{(3)} = \hat{u}$.
- Set $f_i^{(3)}(x) = x$ for every $i \in [n]$.
- For a subset $S \subset [n]$, we set $S^{(3)} = S$.

For every $i \in S$, we have

$$\mathrm{SA}_{Q^{(3)},K^{(3)},V^{(3)}}(X_{S^{(3)}})_i^\top u^{(3)} = \frac{(\hat{v}_i)^\top\hat{u} + (n-|S|)\exp(-M)}{1+(n-|S|)\exp(-M)}.$$

For any given $\epsilon > 0$, we can take M to be sufficiently large such that

$$(\hat{v}_i)^\top\hat{u} - \frac{\epsilon}{3} \le \mathcal{T}_{Q^{(3)},K^{(3)},V^{(3)},f_1^{(3)},\ldots,f_n^{(3)},u^{(3)}}(X_{S^{(3)}})_i \le (\hat{v}_i)^\top\hat{u}.$$

Complete the Proof. Because $\exp(A_{ij}) - \exp(B_{ij}) = H_{ij}$ Combining the above three self-attention heads, we have

$$\hat{v}_i^\top\hat{u} + \sum_{j\in S}H_{ij} - \epsilon \le \sum_{\ell=1}^{3}\mathcal{T}_{Q^{(\ell)},K^{(\ell)},V^{(\ell)},f_1^{(\ell)},\ldots,f_n^{(\ell)},u^{(\ell)}}(X_{S^{(\ell)}})_i \le \hat{v}_i^\top\hat{u} + \sum_{j\in S}H_{ij}.$$

Therefore the three attention heads approximate $g(S,i)$ to arbitrary precision.

■

A.3 Relationship of Model 2 and Choice Models

Model 2 is closely related to *choice models* that generate *conversion probabilities*. Choice models are a fundamental input to many now-canonical optimization problems in the field of operations management, including assortment, inventory, and price optimization. At a high level, a choice model maps an offer set $S \subset [n]$ to conversion probabilities $\{p(i \mid S)\}_{i \in S}$ that sum to one. Interpreting $g(S, i)$ from Model 2 as an unnormalized log-weight, a choice model can naturally set purchase probabilities proportional to $\exp(g(S, i))$. This precisely recovers a particular type of choice model, called the *Halo Multinomial Logit* (Halo MNL) choice model [128].

The Halo MNL choice model is a generalization of the simplest and most widely used *multinomial logit* (MNL) choice model, which assumes that customers choose among a set of items according to a fixed utility associated with each option. While the MNL choice model provides a tractable and interpretable framework, it has well-known limitations, such as the independence of irrelevant alternatives property, which can fail to capture context-dependent or irrational choice behaviors observed in practice. The Halo MNL choice model addresses this by allowing the utility of each item to depend on the presence or absence of other items in the offered set. In particular, certain items can impose a positive or negative "halo effect" on the utility of other items, enabling the model to capture behaviors that violate classical rationality assumptions.

The halo effect is completely parametrized by a matrix $H \in \mathbb{R}^{n \times n}$, where each off-diagonal entry H_{ij} quantifies the halo effect that the presence of item j imposes on the utility of item i: positive values capture complementarity (item j makes i more attractive), while negative values capture substitution (item j detracts from i). Crucially, these interactions are precisely represented by Model 2: the interaction-adjusted scores serve as the weights of the conversion probabilities. Normalizing these weights over the offered set (e.g., via a softmax) yields the Halo MNL choice model with parameter H. Moreover, when H is the zero matrix, there are no interaction terms and the normalization reduces to the standard MNL choice model. Therefore the MNL choice model is a special case nested within our framework.

A.4 Graph Interpretation of Self-attention Layers.

Let

$$W = \mathrm{softmax}\big((XQ)(XK)^\top\big) \in \mathbb{R}^{n \times n}.$$

Because each row of W sums up to one, W can be interpreted as the weighted adjacency matrix of a directed graph on $[n]$, with edge weight $i \to j$ equal to W_{ij}. Then a single self-attention layer performs one round of information passing:

$$\mathrm{SA}_{Q,K,V}(X)_i = \sum_{j=1}^{n} W_{ij}\,(XV)_j,$$

i.e., vertex i aggregates value vectors from its (out-)neighbors according to the edge weights in W.

This perspective also clarifies what stacking self-attention layers does and does *not* do. For $\ell = 1, 2$, define

$$W^{(\ell)} = \mathrm{softmax}\big((XQ^{(\ell)})(XK^{(\ell)})^\top\big), H^{(1)} = W^{(1)}(XV^{(1)}), H^{(2)} = W^{(2)}\big(H^{(1)}V^{(2)}\big),$$

where we ignore point-wise activations functions for intuition. Expanding $H^{(2)}$ shows that information propagate along *two-hop* paths:

$$H_i^{(2)} = \sum_{j=1}^{n} W_{ij}^{(2)}\, H_j^{(1)} = \sum_{j=1}^{n}\sum_{k=1}^{n} W_{ij}^{(2)}\, W_{jk}^{(1)}\, \big(XV^{(1)}V^{(2)}\big)_k,$$

so the contribution of vertex k to vertex i through vertex j factorizes as $W_{ij}^{(2)}\, W_{jk}^{(1)}$. This is a composition of two pairwise interactions, not an arbitrary *triplet* interaction. Indeed, representing a general triplet interaction requires $\Theta(n^3)$ free parameters, whereas two self-attention layers only provide $2n^2$ free parameters. Therefore, stacked self-attention layers naturally model multi-hop effects rather than unrestricted set interactions.

A.5 Proofs in Section 2.2.

Proof of Proposition 4 (a). Fix any instance of the $(k-1)$-Clique problem: let G be the (undirected, unweighted) graph with $n-1$ vertices, and let $A \in \{0,1\}^{(n-1)\times(n-1)}$ be its adjacency matrix (where we follow the convention that diagonal entries are set equal to 1). We will create an explicit instance of Problem (Main) in the following way (using the formulation in P):

- $K = I_{n\times n}$, i.e. the $n \times n$ identity matrix, so that $W = \mathrm{softmax}(QK^\top) = \mathrm{softmax}(Q)$.
- $Q \in \mathbb{R}^{n\times n}$ is set according to

$$Q_{ij} = \begin{cases} 0 & \text{for } i = n \text{ or } j = n \\ -NA_{ij} & \text{for } i, j \neq n, \end{cases}$$

where $N > 1$ is a constant large enough that $(k-1)\exp(-N) \leq 1/2$. In block notation, this is

$$Q = \left[\begin{array}{c|c} -N \cdot A & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \\ \hline 0 \cdots\cdots\cdots 0 & 0 \end{array}\right].$$

- $d_v = 1$, and we set $u = 1$, so that $Vu = V$.
- $V \in \mathbb{R}^{n \times 1}$ is set according to $V_i = 1$ for $i = 1, \ldots, n-1$, and $V_n = M$, where we take any constant $M \geq 2$.
- For $i = 1, \ldots, n-1$, we set $f_i(\cdot)$ to be:

$$f_i(x) = \begin{cases} 0 & \text{if } 0 \leq x \leq \frac{M+1}{2} \\ \frac{6x-3(M+1)}{M-1} & \text{if } \frac{M+1}{2} < x < \frac{2M+1}{3} \\ 1 & \text{if } x \geq \frac{2M+1}{3}. \end{cases}$$

  Because $M \geq 2$, we have $(2M+1)/3 > (M+1)/2$, so $f_i(\cdot)$ is continuous piece-wise linear for every $i = 1, \ldots, n-1$. We set $f_n(x) = 0$ for every x.

Note that the quantity

$$\frac{(w_i \odot Vu)^\top x}{w_i^\top x}$$

remains unchanged if the vector w_i is multiplied by a non-zero constant. Thus, rescaling the rows of W does not change P. So for simplicity of exposition, we replace W with W', defined to be

$$W'_{ij} = \begin{cases} 1 & \text{for } i = n \text{ or } j = n \\ \exp(-NA_{ij}) & \text{for } i, j \neq n. \end{cases}$$

As a sanity check, W is simply W' with each row rescaled to sum to one.

Now notice that because $f_i(x) \leq 1$ for every $i = 1, \ldots, n-1$, and $f_n(x) = 0$, the optimal value of this instance of P is at most $k-1$. It will suffice to show that G has a clique of size $k-1$ if and only if the optimal value is $k-1$. We prove both directions separately.

If G has a clique of size $k-1$, then the optimal value is $k-1$: Suppose G has a clique of size $k-1$, and let $C \subset [n-1]$ be the vertex set of one such clique. Consider the solution x^*, where $x_i^* = 1$ if and only if $i \in C \cup \{n\}$. We will show that the objective value at x^* is $k-1$ (and thus the optimal value is $k-1$):

Because $W'_{ij} = \exp(-N)$ whenever $i, j \in C$, we have

$$\begin{aligned}\sum_{i=1}^{n} x_i^* f_i\left(\frac{(w'_i \odot Vu)^\top x^*}{w_i'^\top x^*}\right) &= \sum_{i\in C} f_i\left(\frac{(w'_i \odot Vu)^\top x^*}{w_i'^\top x^*}\right) \\ &= \sum_{i\in C} f_i\left(\frac{M + (k-1)\exp(-N)}{1 + (k-1)\exp(-N)}\right).\end{aligned}$$

Since N was chosen to be large enough that $(k-1)\exp(-N) \le 1/2$, it follows that

$$\frac{M + (k-1)\exp(-N)}{1 + (k-1)\exp(-N)} \ge \frac{M + \frac{1}{2}}{1 + \frac{1}{2}} = \frac{2M+1}{3}.$$

Therefore,

$$\sum_{i=1}^{n} x_i^* f_i\left(\frac{(w'_i \odot Vu)^\top x^*}{w_i'^\top x^*}\right) = \sum_{i\in C} f_i\left(\frac{M + (k-1)\exp(-N)}{1 + (k-1)\exp(-N)}\right) = |C| = k-1.$$

If the optimal value is $k-1$, the G has a clique of size $k-1$: Suppose the optimal value is $k-1$. Let x^* be an optimal solution, and let $C \subset [n-1]$ be the index set such that $x_i^* = 1$ for $i \in C$. Notice that if a solution x has $x_n = 0$, then since $(w'_i \odot Vu)_j = W'_{ij}$ for all $j \in [n-1]$, we have

$$\sum_{i=1}^{n} x_i f_i\left(\frac{(w'_i \odot Vu)^\top x}{w_i'^\top x}\right) = \sum_{i=1}^{n} x_i f_i(1) = 0.$$

Therefore, it must be the case that $x_n^* = 1$. Also, because

$$k-1 = \sum_{i=1}^{n} x_i^* f_i\left(\frac{(w'_i \odot Vu)^\top x^*}{w_i'^\top x^*}\right) \le \sum_{i=1}^{n-1} x_i^* = |C|,$$

we must have $|C| = k-1$ and

$$f_i\left(\frac{(w'_i \odot Vu)^\top x^*}{w_i'^\top x^*}\right) = 1$$

for every $i \in C$.

Let $d(i)$ be the degree of vertex i in the induced subgraph of G with vertex set C. Then

$$\begin{aligned}\sum_{i=1}^{n} x_i^* f_i\left(\frac{(w_i' \odot Vu)^\top x^*}{w_i'^\top x^*}\right) &= \sum_{i\in C} f_i\left(\frac{(w_i' \odot Vu)^\top x^*}{w_i'^\top x^*}\right) \\ &= \sum_{i\in C} f_i\left(\frac{M + (k-2-d(i)) + \exp(-N)(d(i)+1)}{1 + (k-2-d(i)) + \exp(-N)(d(i)+1)}\right) \\ &\leq \sum_{i\in C} f_i\left(\frac{M + (k-2-d(i))}{1 + (k-2-d(i))}\right),\end{aligned}$$

where the last inequality follows since $M \geq 2$. Because $|C| = k-1$, we have $d(i) \leq k-2$. Suppose for the sake of contradiction that $d(i) \leq k-3$ for some $i \in I$. Then

$$f_i\left(\frac{M + (k-2-d(i))}{1 + (k-2-d(i))}\right) \leq f_i\left(\frac{M+1}{2}\right) = 0,$$

which contradicts that

$$f_i\left(\frac{(w_i' \odot Vu)^\top x^*}{w_i'^\top x^*}\right) = 1$$

for every $i \in C$. Therefore we must have $d(i) = k-2$ for every $i \in C$. Hence C corresponds to the vertex set of a clique of size $k-1$. ■

Proof of Proposition 4 (b). Fix any constant $M \geq 1$. We construct an instance of Problem (Main) (using the fornulation in P) by applying the Johnson-Lindenstrauss Lemma:

Lemma 5 (Johnson-Lindenstrauss Lemma)**.** *For any $0 < \epsilon < 1$ and any set S of m points in $\mathbb{R}^n$, there exists a universal constant $c > 0$ and a linear function $f : \mathbb{R}^n \to \mathbb{R}^d$ with $d = c\epsilon^{-2}\log(m)$ such that*

$$(1-\epsilon)\|x_i\|_2^2 \leq \|f(x_i)\|_2^2 \leq (1+\epsilon)\|x_i\|_2^2$$

for all $x_i \in S$ and

$$(1-\epsilon)\|x_i - x_j\|_2^2 \leq \|f(x_i) - f(x_j)\|_2^2 \leq (1+\epsilon)\|x_i - x_j\|_2^2$$

for all $x_i, x_j \in S$.

Take N to be large enough such that $\exp(-N) \leq 1/2M$. Take $0 < \delta < 1$ to be small enough such that $\exp(-\delta N) \geq 1 - \exp(-N)$. Then N and δ can be chosen to be both only depend on M. We obtain the following corollary:

Corollary 2 (Corollary of Lemma 5.)**.** *There exists a number $c(M) > 0$ and a set S of $\exp(c(M) \cdot d_{kq})$ unit vectors in $\mathbb{R}^{d_{kq}}$ such that $|u_i^\top u_j| \le \delta$ for every $u_i, u_j \in S$ and $u_i \neq u_j$.*

Proof of Corollary 2. Let $c > 0$ be the universal constant in Lemma 5 and let $n = \exp(c^{-1}(\delta/4)^2 \cdot d_{kq})$. Let $c(M) = c^{-1}(\delta/4)^2$, then $n = \exp(c(M) \cdot d_{kq})$. Because $\delta > 0$ only depends on M, we have $c(M) > 0$ also only depends on M. Consider $\{e_i\}_{i=1}^n \subset \mathbb{R}^n$ where $e_i \in \mathbb{R}^n$ is the unit vector where the i-th entry equals to 1 and all other entries equal to 0. Then we have $\|e_i - e_j\|_2^2 = 2$ for every $e_i \neq e_j$. By Lemma 5, there exists a linear function $f : \mathbb{R}^n \to \mathbb{R}^{d_{kq}}$ such that

$$\left(1 - \frac{\delta}{4}\right) \le \|f(e_i)\|_2^2 \le \left(1 + \frac{\delta}{4}\right)$$

for all e_i and

$$2\left(1 - \frac{\delta}{4}\right) \le \|f(e_i) - f(e_j)\|_2^2 \le 2\left(1 + \frac{\delta}{4}\right)$$

for all $e_i \neq e_j$.

For every $e_i \neq e_j$, because

$$\|f(e_i) - f(e_j)\|_2^2 = \|f(e_i)\|_2^2 + \|f(e_j)\|_2^2 - 2f(e_i)^\top f(e_j),$$

we have

$$\begin{aligned} f(e_i)^\top f(e_j) &= (\|f(e_i)\|_2^2 + \|f(e_j)\|_2^2 - \|f(e_i) - f(e_j)\|_2^2)/2 \\ &\le \left(\left(1 + \frac{\delta}{4}\right) + \left(1 + \frac{\delta}{4}\right) - 2\left(1 - \frac{\delta}{4}\right)\right)/2 \\ &= \frac{\delta}{2}, \end{aligned}$$

and

$$\begin{aligned} f(e_i)^\top f(e_j) &= (\|f(e_i)\|_2^2 + \|f(e_j)\|_2^2 - \|f(e_i) - f(e_j)\|_2^2)/2 \\ &\ge \left(\left(1 - \frac{\delta}{4}\right) + \left(1 - \frac{\delta}{4}\right) - 2\left(1 + \frac{\delta}{4}\right)\right)/2 \\ &= -\frac{\delta}{2}. \end{aligned}$$

Let $u_i = f(e_i)/\|f(e_i)\|_2$ for all $i \in [n]$, then each u_i is a unit vector. Moreover, for every $u_i \neq u_j$,

$$u_i^\top u_j = \frac{f(e_i)^\top f(e_j)}{\|f(e_i)\|_2 \|f(e_j)\|_2} \le \frac{\frac{\delta}{2}}{1 - \frac{\delta}{4}} = \delta\left(\frac{2}{4 - \delta}\right) < \delta,$$

where the last inequality follows since $0 < \delta < 1$. Similarly,

$$u_i^\top u_j = \frac{f(e_i)^\top f(e_j)}{\|f(e_i)\|_2 \|f(e_j)\|_2} \geq \frac{-\frac{\delta}{2}}{1 - \frac{\delta}{4}} = -\delta\left(\frac{2}{4-\delta}\right) > -\delta.$$

Therefore $|u_i^\top u_j| \leq \delta$ for every $u_i \neq u_j$. Take $S = \{u_i\}_{i=1}^n$ gives the desired set of unit vectors. ■

Corollary 2 states that there exists a set of unit vectors in $\mathbb{R}^{d_{kq}}$, with size exponential in d_{kq}, where all unit vectors in the set are approximately orthonormal.

Fix any d_{kq} such that $\exp(c(M) \cdot d_{kq}) \leq n - 1$ and any $k \geq M + 1$. Let G be a graph with $n - 1$ vertices $v_1, \ldots, v_{n-1}$ and $\ell = \exp(c(M) \cdot d_{kq})$ disjoint cliques, each of size at least $k - M$ and at most $k - 1$. Let $I_1, \ldots, I_\ell \subset [n-1]$ be the index sets of vertices corresponding to these ℓ cliques. That is, $\{v_i\}_{i \in I_{\ell'}}$ forms a clique for each $\ell' \in [\ell]$. Without loss of generality we assume $\ell' \in I_{\ell'}$ for every $\ell' \in [\ell]$.

By Corollary 2, there exists ℓ unit vectors $u_1, \ldots, u_\ell \in \mathbb{R}^{d_{kq}}$ such that $|u_i^\top u_j| \leq \delta$ for $i \neq j$. Let $u_{\ell+1}, \ldots, u_{n-1}$ be unit vectors such that $u_i = u_{\ell'}$ for $i \in I_{\ell'}$. That is, for indices i, j such that v_i and v_j are in the same clique, we have $u_i = u_j$. Let $U \in \mathbb{R}^{(n-1)\times d_{kq}}$ where the i-th row of U is $u_i^\top$. Let $A = UU^\top \in \mathbb{R}^{(n-1)\times(n-1)}$, then A has rank at most d_{kq}.

Because A has rank at most d_{kq}, there exists $Q, K \in \mathbb{R}^{n \times d_{kq}}$ such that

$$QK^\top = \left[\begin{array}{c|c} & 0 \\ -N \cdot A & \vdots \\ & 0 \\ \hline 0 \cdots\cdots\cdots 0 & 0 \end{array}\right].$$

We create an explicit instance of P similar to the instance in the proof of Proposition 4 (a):

- $W = \text{softmax}(QK^\top)$. For simplicity of exposition, we replace W with W', defined to be

$$W'_{ij} = \begin{cases} 1 & \text{for } i = n \text{ or } j = n \\ \exp(-N|u_i^\top u_j|) & \text{for } i, j \neq n \text{ and } A_{ij} \neq 1 \\ \exp(-N) & \text{for } i, j \neq n \text{ and } A_{ij} = 1, \end{cases}$$

  As a sanity check, W is simply W' with each row rescaled to sum to one.

- $d_v = 1$, and we set $u = 1$, so that $Vu = V$.
- $V \in \mathbb{R}^{n\times 1}$ is set according to $V_i = 1$ for $i = 1, \ldots, n-1$, and $V_n = 2$.
- For $i = 1, \ldots, n-1$, we set $f_i(\cdot)$ to be:

$$f_i(x) = \begin{cases} 0 & \text{if } 0 \le x \le \frac{2+\exp(-N)k+\frac{1}{4}}{1+\exp(-N)k+\frac{1}{4}} \\ \frac{x - \frac{2+\exp(-N)k+\frac{1}{4}}{1+\exp(-N)k+\frac{1}{4}}}{\frac{2+\exp(-N)k}{1+\exp(-N)k} - \frac{2+\exp(-N)k+\frac{1}{4}}{1+\exp(-N)k+\frac{1}{4}}} & \text{if } \frac{2+\exp(-N)k+\frac{1}{4}}{1+\exp(-N)k+\frac{1}{4}} < x < \frac{2+\exp(-N)k}{1+\exp(-N)k} \\ 1 & \text{if } x \ge \frac{2+\exp(-N)k}{1+\exp(-N)k}. \end{cases}$$

  Because $\frac{2+\exp(-N)k}{1+\exp(-N)k} > \frac{2+\exp(-N)k+\frac{1}{4}}{1+\exp(-N)k+\frac{1}{4}}$, we have $f_i(\cdot)$ is continuous piece-wise linear for every $i = 1, \ldots, n-1$. We set $f_n(x) = 0$ for every x.

Now notice that because $f_i(x) \le 1$ for every $i = 1, \ldots, n-1$, and $f_n(x) = 0$, the optimal value of this instance of Problem P is at most $k-1$. It will suffice to show that the largest clique G has size k' if and only if the optimal value is k'. We prove both directions separately.

If G has a clique of size k', then the optimal value is at least k': Suppose G has a clique of size k'. Without loss of generality, let $I_1 \subset [n-1]$ correspond the vertex set of a clique of size k' in G. Consider the solution x^*, where $x^*_i = 1$ for $i \in I_1 \cup \{n\}$. Because $|I_1 \cup \{n\}| \le k$, the solution x^* is feasible. We will show that the objective value of P at x^* is k' (and thus the optimal value is at least k').

Because $W'_{ij} = \exp(-N)$ whenever $i, j \in I_1$, we have

$$\begin{aligned} \sum_{i=1}^{n} x^*_i f_i\left(\frac{(w'_i \odot Vu)^\top x^*}{w_i'^\top x^*}\right) &= \sum_{i\in I_1} f_i\left(\frac{(w'_i \odot Vu)^\top x^*}{w_i'^\top x^*}\right) \\ &= \sum_{i\in I_1} f_i\left(\frac{2 + k'\exp(-N)}{1 + k'\exp(-N)}\right). \end{aligned}$$

Since $k' \le k-1$. It follows that

$$\frac{2 + k'\exp(-N)}{1 + k'\exp(-N)} > \frac{2 + k\exp(-N)}{1 + k\exp(-N)}.$$

Therefore

$$\sum_{i=1}^{n} x^*_i f_i\left(\frac{(w'_i \odot Vu)^\top x^*}{w_i'^\top x^*}\right) = \sum_{i\in I_1} f_i\left(\frac{2 + k'\exp(-N)}{1 + k'\exp(-N)}\right) = |I_1| = k'.$$

If the largest clique in G has size k', then the optimal value is at most k': By our assumption we have $k' \geq k - M$. Suppose for the sake of contradiction that the optimal value is $k^* > k'$. Let x^* be an optimal solution to P. Notice if $x_n^* = 0$, then since $(w_i' \odot Vu)_j = W_{ij}'$ for all $j \in [n-1]$, we have

$$\sum_{i=1}^{n} x_i^* f_i\left(\frac{(w_i' \odot Vu)^\top x^*}{w_i'^\top x^*}\right) = \sum_{i=1}^{n} x_i^* f_i(1) = 0.$$

Therefore we must have $x_n^* = 1$.

Let $I \subset [n-1]$ be the index set where $x_i^* = 1$ for $i \in I$. Because $f_i(x) \leq 1$ for every $i = 1, \ldots, n-1$, we have $|I| \geq k^*$. Let $d(i)$ be the degree of vertex i in the induced subgraph of G with vertex set $\{v_i\}_{i \in I}$. Because G consists of disjoint cliques with size at most k', we have $d(i) \leq k' - 1$ for every $i \in C$.

Fix any $i \in I$ such that

$$f_i\left(\frac{(w_i' \odot Vu)^\top x^*}{w_i'^\top x^*}\right) > 0,$$

or, equivalently,

$$\frac{(w_i' \odot Vu)^\top x^*}{w_i'^\top x^*} > \frac{2 + \exp(-N) + \frac{1}{4}}{1 + \exp(-N) + \frac{1}{4}}.$$

Because $\exp(-N|u_i^\top u_j|) \geq \exp(-\delta N)$ for every $i, j \neq n$ and $A_{ij} \neq 1$, we have

$$\begin{aligned}
\frac{(w_i' \odot Vu)^\top x^*}{w_i'^\top x^*} &= \frac{2 + \sum_{j \in I, A_{ij} \neq 1} \exp(-N|u_i^\top u_j|) + (d(i)+1)\exp(-N)}{1 + \sum_{j \in I, A_{ij} \neq 1} \exp(-N|u_i^\top u_j|) + (d(i)+1)\exp(-N)} \\
&\leq \frac{2 + (|I| - d(i) - 1)\exp(-\delta N) + (d(i)+1)\exp(-N)}{1 + (|I| - d(i) - 1)\exp(-\delta N) + (d(i)+1)\exp(-N)} \\
&\leq \frac{2 + (k^* - d(i) - 1)\exp(-\delta N) + (d(i)+1)\exp(-N)}{1 + (k^* - d(i) - 1)\exp(-\delta N) + (d(i)+1)\exp(-N)} \\
&= \frac{2 + \exp(-\delta N)k^* - (\exp(-\delta N) - \exp(-N))(d(i)+1)}{1 + \exp(-\delta N)k^* - (\exp(-\delta N) - \exp(-N))(d(i)+1)} \\
&\leq \frac{2 + \exp(-\delta N)k^* - (\exp(-\delta N) - \exp(-N))k'}{1 + \exp(-\delta N)k^* - (\exp(-\delta N) - \exp(-N))k'},
\end{aligned}$$

where the second inequality follows since $|I| \geq k^*$, and the last inequality follows since $\exp(-\delta N) - \exp(-N) > 0$ and $d(i) \leq k' - 1$. Therefore

$$\frac{2 + \exp(-\delta N)k^* - (\exp(-\delta N) - \exp(-N))k'}{1 + \exp(-\delta N)k^* - (\exp(-\delta N) - \exp(-N))k'} > \frac{2 + \exp(-N)k + \frac{1}{4}}{1 + \exp(-N)k + \frac{1}{4}}.$$

So we have

$$\exp(-\delta N)k^* - (\exp(-\delta N) - \exp(-N))k' < \exp(-N)k + \frac{1}{4}.$$

Rearrange the above inequality gives

$$\exp(-\delta N)(k^* - k') < \exp(-N)(k - k') + \frac{1}{4}.$$

Because $k^* \geq k' + 1$ and $k' \geq k - M$, we get

$$\exp(-\delta N) < \exp(-N)M + \frac{1}{4}.$$

However, since $\exp(-N) \leq 1/2M$ and $\exp(-\delta N) \geq 1 - 1/2M$, we have

$$\exp(-\delta N) \geq 1 - \frac{1}{2M} \geq \frac{5}{6} > \exp(-N)M + \frac{1}{4}.$$

A contradiction. Therefore $k^* \leq k'$. ■

Proof of Proposition 5. Our proof is based on a reduction from Problem (Main) to the well-known *Multi-dimensional Knapsack Problem* (MDKP), defined as follows:

Definition 5 (Multi-dimensional Knapsack Problem)**.** The Multi-dimensional Knapsack Problem (MDKP) with c items and d dimensions is defined as:

$$\text{(MDKP)} \qquad \begin{aligned} \max \quad & f_{\mathsf{MDKP}}(x) = p^\top x \\ \text{s.t.} \quad & y_i^\top x \leq t_i \quad \forall i \in [d], \\ & x \in \{0,1\}^c. \end{aligned}$$

Here, $p \in \mathbb{N}^c$, and for all $i \in [d]$, we have $y_i \in \mathbb{Z}^c_{\geq 0}$ and $t_i \in \mathbb{Z}_{\geq 0}$.[2]

We present the reduction in the following proposition:

Proposition 16. *Consider Problem* (Main)*, written in the equivalent form* P *as given in Observation 2, reproduced below, which has parameters n, k, and r_+, where r_+ is the non-negative rank of W:*

$$\text{(P)} \qquad \begin{aligned} \max \quad & f_{\mathsf{P}(I)}(x) = \sum_{i=1}^{n} x_i f_i \left(\frac{(w_i \odot Vu)^\top x}{w_i^\top x} \right) \\ \text{s.t.} \quad & x \in \{0,1\}^n, \quad 1 \leq e^\top x \leq k. \end{aligned}$$

Now consider an instance of MDKP *with parameters c and d. Suppose there exists an algorithm* ALG *for solving* P *with parameters $n = k = c + d + 1$ and*

[2]We assume $p_{\min} > 0$ without loss of generality. If $p_j = 0$ for some $j \in [c]$, there always exists an optimal solution with $x_j = 0$. Hence, we can safely ignore the j-th entry of p, x, and each y_i.

$\min\{c, d\} \le r_+ \le c + d + 1$*, such that for any sufficiently small* $\epsilon > 0$*, the algorithm satisfies*

$$\mathrm{ALG}_{\mathsf{P}} \ge (1 - \epsilon)\,\mathrm{OPT}_{\mathsf{P}}$$

with runtime T*. Then we can construct an algorithm* ALG' *for solving* MDKP *with parameters* c *and* d*, such that for the same* ϵ*, it satisfies*

$$\mathrm{ALG}'_{\mathsf{MDKP}} \ge (1 - \epsilon)\,\mathrm{OPT}_{\mathsf{MDKP}}$$

with runtime $O(T)$*.*

Proof of Proposition 16. Fix an MDKP instance. We create an instance of P as follows:

- $n = k = d + c + 1$. Here, out of the n total variables, the first d variables will correspond to the d constraints of MDKP, the c variables after that will correspond to the c variables of MDKP, and the last variable will be a dummy variable that must be selected in any optimal solution of P.

- For each $i = 1, \ldots, d$, set

$$w_{ij} = \begin{cases} 0 & \text{for } j = 1, \ldots, d \\ y_{i,j-d} & \text{for } j = d+1, \ldots, d+c \\ 1 & \text{for } j = d + c + 1. \end{cases}$$

  That is, in block notation,

$$w_i^\top = \left[\; 0 \cdots 0 \;\middle|\; y_i^\top \;\middle|\; 1 \;\right].$$

  Then we have

$$w_i^\top x = \sum_{j=1}^{c} y_{ij} x_{d+j} + x_{d+c+1}.$$

  Moreover, since $y_i \in \mathbb{Z}^c_{\ge 0}$ for each $i \in [d]$, the non-negative rank of the sub-matrix of W consisting of its first d rows is at most $\min\{c, d\}$.

- For each $i = d + 1, \ldots, d + c + 1$, set $w_i \in \mathbb{R}^{d+c+1}_{\ge 0}$'s to be any non-zero vectors such that W has the desired non-negative rank r_+. This is possible since $\min\{c, d\} \le r_+ \le c + d + 1$.

- $d_v = 1$, and we set $u = 1$, so that $Vu = V$.

- $V \in \mathbb{R}^{(d+c+1)\times 1}$ is set to be $V_i = 0$ for $i = 1, \ldots, d$, and $V_i = 1$ for $i = d+1, \ldots, d+c$, and $V_{d+c+1} = 2$.
- For each $i = 1, \ldots, d$, set

$$f_i(x) = \begin{cases} -\left(\sum_{j=1}^{c} p_j + 1\right) & \text{for } x \le \frac{t_i+3}{t_i+2} \\ \dfrac{x - \frac{t_i+2}{t_i+1}}{\frac{t_i+2}{t_i+1} - \frac{t_i+3}{t_i+2}} \cdot \left(\sum\limits_{j=1}^{c} p_j + 1\right) & \text{for } \frac{t_i+3}{t_i+2} < x < \frac{t_i+2}{t_i+1} \\ 0 & \text{for } x \ge \frac{t_i+2}{t_i+1}. \end{cases}$$

  Then $f_i(x)$ is continuous piecewise-linear.
- For each $i = d+1, \ldots, d+c$, set $f_i(x) = p_{i-d}$. Set $f_{d+c+1}(x) = 0$.

Because $p_{\min} > 0$ and $\vec{0} \in \{0,1\}^c$ is a feasible solution to MDKP, we have $\mathrm{OPT}_{\mathsf{MDKP}} = 0$ if and only if $\vec{0} \in \{0,1\}^c$ is the only feasible solution to MDKP. Moreover, if $\vec{0} \in \{0,1\}^c$ is not the only feasible solution to MDKP, then $\mathrm{OPT}_{\mathsf{MDKP}} \ge p_{\min}$.

Our proof relies on the following lemma.

Lemma 6. *Let $x \in \{0,1\}^n$ be a feasible solution to* P *such that $f_{\mathsf{P}}(x) > 0$. Then we can construct a feasible solution $z \in \{0,1\}^c$ to* MDKP *such that $f_{\mathsf{P}}(x) = f_{\mathsf{MDKP}}(z)$.*

Conversely, let $z \in \{0,1\}^c$ be a feasible solution to MDKP *such that $f_{\mathsf{MDKP}}(z) > 0$. Then we can construct a feasible solution $x \in \{0,1\}^n$ to* P *such that $f_{\mathsf{P}}(x) = f_{\mathsf{MDKP}}(z)$.*

Lemma 6 gives a correspondence between solutions to P and solutions to MDKP. In particular, as a corollary, Lemma 6 implies the relationship between the optimal objective values of P and MDKP.

Corollary 3 (Corollary of Lemma 6.)**.** $\mathrm{OPT}_{\mathsf{P}} \le 0$ *if and only if* $\mathrm{OPT}_{\mathsf{MDKP}} = 0$. *Moreover, suppose* $\mathrm{OPT}_{\mathsf{MDKP}} > 0$. *Then* $\mathrm{OPT}_{\mathsf{P}} = \mathrm{OPT}_{\mathsf{MDKP}}$.

Proof of Corollary 3. Suppose $\mathrm{OPT}_{\mathsf{P}} > 0$. Let x^* be an optimal solution to P. By Lemma 6 there exists a feasible solution z to MDKP such that $f_{\mathsf{MDKP}}(z) > 0$ and

$$\mathrm{OPT}_{\mathsf{P}} = f_{\mathsf{P}}(x^*) = f_{\mathsf{MDKP}}(z) \le \mathrm{OPT}_{\mathsf{MDKP}}.$$

Conversely, suppose $\mathrm{OPT}_{\mathsf{MDKP}} > 0$. Let z^* be an optimal solution to MDKP. By Lemma 6 there exists a feasible solution x to MDKP such that

$$\mathrm{OPT}_{\mathsf{MDKP}} = f_{\mathsf{MDKP}}(z^*) = f_{\mathsf{P}}(x) \le \mathrm{OPT}_{\mathsf{P}}.$$

We also get

$$f_{\mathsf{P}}(x) \geq f_{\mathsf{MDKP}}(z^*) > 0.$$

■

Before proving Lemma 6, we first show that it is sufficient to prove Lemma 6. Assume we have an algorithm ALG for solving P such that, for any sufficiently small $\epsilon > 0$, we have ALG satisfies

$$\mathrm{ALG}_{\mathsf{P}} \geq (1 - \epsilon)\,\mathrm{OPT}_{\mathsf{P}}.$$

Then our proposed algorithm ALG′ for solving MDKP works as follows:

- If $\mathrm{ALG}_{\mathsf{P}} \leq 0$, ALG′ outputs $\vec{0} \in \{0,1\}^c$.
- If $\mathrm{ALG}_{\mathsf{P}} > 0$, let x be the solution to P given by $\mathrm{ALG}_{\mathsf{P}}$. Then ALG′ outputs the solution $z \in \{0,1\}^c$ to MDKP given by Lemma 6. That is, $z \in \{0,1\}^c$ that satisfies

$$f_{\mathsf{P}}(x) = f_{\mathsf{MDKP}}(z).$$

Performance Guarantee of ALG′: Suppose $\mathrm{OPT}_{\mathsf{MDKP}} = 0$. Then by Corollary 3,

$$\mathrm{ALG}_{\mathsf{P}} \leq \mathrm{OPT}_{\mathsf{P}} \leq 0.$$

Therefore ALG′ correctly outputs $\vec{0} \in \{0,1\}^c$.

On the other hand, suppose $\mathrm{OPT}_{\mathsf{MDKP}} > 0$. Then by Corollary 3,

$$\mathrm{ALG}'_{\mathsf{MDKP}} = f_{\mathsf{MDKP}}(z) = f_{\mathsf{P}}(x) \geq (1-\epsilon)\mathrm{OPT}_{\mathsf{P}} = (1-\epsilon)\mathrm{OPT}_{\mathsf{MDKP}}.$$

This proves the desired performance guarantee of ALG′. To finish the proof, we prove Lemma 6 and analyze the runtime of ALG′.

Proof of Lemma 6. Recall for each $i = 1, \ldots, d$, we set

$$w_{ij} = \begin{cases} 0 & \text{for } j = 1, \ldots, d \\ y_{i,j-d} & \text{for } j = d+1, \ldots, d+c \\ 1 & \text{for } j = d+c+1. \end{cases}$$

Also, $(Vu)_i = 0$ for $i = 1, \ldots, d$, and $(Vu)_i = 1$ for $i = d+1, \ldots, d+c$, and $(Vu)_{d+c+1} = 2$. Therefore for $i = 1, \ldots, d$ we have

$$\frac{(w_i \odot Vu)^\top x}{w_i^\top x} = \frac{\sum_{j=1}^c y_{ij} x_{d+j} + 2x_{d+c+1}}{\sum_{j=1}^c y_{ij} x_{d+j} + x_{d+c+1}}.$$

Recall for each $i = d+1, \ldots, d+c$ we set $f_i(x) = p_{i-d}$, and we set $f_{d+c+1}(x) = 0$. Therefore we have

$$\begin{aligned} f_{\mathsf{P}}(x) &= \sum_{i=1}^{c+d+1} x_i f_i\left(\frac{(w_i \odot Vu)^\top x}{w_i^\top x}\right) \\ &= \sum_{i=1}^{d} x_i f_i\left(\frac{\sum_{j=1}^{c} y_{ij} x_{d+j} + 2x_{d+c+1}}{\sum_{j=1}^{c} y_{ij} x_{d+j} + x_{d+c+1}}\right) + \sum_{i=1}^{c} x_{d+i} p_i. \end{aligned}$$

First, let $x \in \{0,1\}^n$ be a feasible solution to $\mathsf{P}(I)$ such that $f_{\mathsf{P}}(x) > 0$. Let $z \in \{0,1\}^c$ where $z_j = x_{d+j}$ for $j \in [c]$. We claim that z is a feasible solution to MDKP and

$$f_{\mathsf{P}}(x) = f_{\mathsf{MDKP}}(z).$$

Because $\sum_{i=1}^{c} x_{d+i} p_i < \sum_{j=1}^{c} p_j + 1$, we must have $x_i = 1$ for every $i \in [d]$. Moreover, if $x_{d+c+1} = 0$, then

$$f_i\left(\frac{\sum_{j=1}^{c} y_{ij} x_{d+j} + 2x_{d+c+1}}{\sum_{j=1}^{c} y_{ij} x_{d+j} + x_{d+c+1}}\right) = f_i(1) < 0$$

for every $i \in [d]$. Hence we must have $x_{d+c+1} = 1$. Because $y_i \in \mathbb{Z}_{\geq 0}^c$ and $t_i \in \mathbb{Z}_{\geq 0}$ for all $i \in [d]$, if $\sum_{j=1}^{c} y_{ij} x_{d+j} > t_i$, we must have $\sum_{j=1}^{c} y_{ij} x_{d+j} \geq t_i + 1$. Then

$$f_i\left(\frac{\sum_{j=1}^{c} y_{ij} x_{d+j} + 2x_{d+c+1}}{\sum_{j=1}^{c} y_{ij} x_{d+j} + x_{d+c+1}}\right) \leq f_i\left(\frac{t_i + 3}{t_i + 2}\right) = -\left(\sum_{j=1}^{c} p_j + 1\right).$$

Therefore we must have $\sum_{j=1}^{c} y_{ij} x_{d+j} \leq t_i$ for all $i \in [d]$. Hence $\sum_{j=1}^{c} y_{ij} z_j \leq t_i$ for all $i \in [d]$, which shows z is a feasible solution to MDKP. Finally, because

$$\frac{\sum_{j=1}^{c} y_{ij} x_{d+j} + 2x_{d+c+1}}{\sum_{j=1}^{c} y_{ij} x_{d+j} + x_{d+c+1}} \geq \frac{t_i + 2}{t_i + 1}$$

for all $i \in [d]$, we have

$$f_i\left(\frac{\sum_{j=1}^{c} y_{ij} x_{d+j} + 2x_{d+c+1}}{\sum_{j=1}^{c} y_{ij} x_{d+j} + x_{d+c+1}}\right) = 0$$

for all $i \in [d]$. Therefore

$$\begin{aligned} f_{\mathsf{P}}(x) &= \sum_{i=1}^{d} x_i f_i\left(\frac{\sum_{j=1}^{c} y_{ij} x_{d+j} + 2x_{d+c+1}}{\sum_{j=1}^{c} y_{ij} x_{d+j} + x_{d+c+1}}\right) + \sum_{i=1}^{c} x_{d+i} p_i \\ &= \sum_{i=1}^{c} z_i p_i \\ &= f_{\mathsf{MDKP}}(z). \end{aligned}$$

Conversely, let $z \in \{0,1\}^c$ be a feasible solution to MDKP such that $f_{\mathsf{MDKP}}(z) > 0$. Let x be a solution to P where $x_i = 1$ for $i = 1, \ldots, d$ and $i = d + c + 1$, and $x_{d+i} = z_i$ for $i = 1, \ldots, c$. We claim that x is a feasible solution to MDKP and

$$f_{\mathsf{P}}(x) = f_{\mathsf{MDKP}}(z).$$

Because z is a feasible solution to MDKP, we have $\sum_{j=1}^{c} y_{ij} z_j \leq t_i$ for all $i \in [d]$. Therefore $\sum_{j=1}^{c} y_{ij} x_{d+j} \leq t_i$ for all $i \in [d]$. Then

$$\frac{\sum_{j=1}^{c} y_{ij} x_{d+j} + 2x_{d+c+1}}{\sum_{j=1}^{c} y_{ij} x_{d+j} + x_{d+c+1}} \geq \frac{t_i + 2}{t_i + 1}$$

for all $i \in [d]$. Therefore we have

$$f_i\left(\frac{\sum_{j=1}^{c} y_{ij} x_{d+j} + 2x_{d+c+1}}{\sum_{j=1}^{c} y_{ij} x_{d+j} + x_{d+c+1}}\right) = 0$$

for all $i \in [d]$. Hence

$$\begin{aligned} f_{\mathsf{P}}(x) &= \sum_{i=1}^{d} x_i f_i\left(\frac{\sum_{j=1}^{c} y_{ij} x_{d+j} + 2x_{d+c+1}}{\sum_{j=1}^{c} y_{ij} x_{d+j} + x_{d+c+1}}\right) + \sum_{i=1}^{c} x_{d+i} p_i \\ &= \sum_{i=1}^{c} z_i p_i \\ &= f_{\mathsf{MDKP}}(z). \end{aligned}$$

■

Given an MDKP instance, our construction of the corresponding P instance takes $O(1)$ runtime. Also, the procedure of the construction of a feasible solution $z \in \{0,1\}^c$ to MDKP given a feasible solution to P described in Lemma 6 takes $O(1)$ runtime: we simply set $z \in \{0,1\}^c$ where $z_j = x_{d+j}$ for $j \in [c]$. Therefore, if ALG has runtime T, our ALG′ has runtime $O(T)$.

■

By Proposition 16, any $(1-\epsilon)$-approximation scheme for P can be directly translated into a $(1-\epsilon)$-approximation scheme for MDKP with comparable runtime. Therefore, many hardness results for MDKP carry over directly to P. We cite several such results below:

Proposition 17 (Theorem 6 in [103]). *If $d = 2$,* MDKP *admits no $(1-\epsilon)$-approximation scheme with runtime $f(1/\epsilon)\, c^{O(1)}$ for any function f, assuming the k-Clique problem is not Fixed-Parameter Tractable.*

Proposition 18 (Theorem 5.1 in [87]). *If $d = 2$,* MDKP *admits no $(1-\epsilon)$-approximation scheme with runtime $f(1/\epsilon)\, c^{O(1)}$ for any function f, with runtime $f(1/\epsilon)\, c^{o(1/\epsilon)}$ for any function f, assuming Exponential Time Hypothesis holds.*

Proposition 19 (Corollary of [60]). *For general d,* MDKP *admits no $(1-\epsilon)$-approximation scheme with runtime*

$$k^{o\left(\frac{d}{\epsilon \log^2(d/\epsilon)}\right)} \text{ or } k^{o(\sqrt{d})},$$

assuming Gap Exponential Time Hypothesis holds.

These results, along with Proposition 16, gives our desired lower bound statement in Proposition 5.

■

A.6 Proofs in Section 2.3.

Proof of Proposition 6. Let $i_1 \in I_1, \ldots, i_\ell \in I_\ell$. Let $\pi : \{(m, m') \mid m, m' \in [\ell], m \leq m'\} \to [\ell(\ell+1)/2]$ be any bijection. Let $A, B \in \mathbb{R}_{\geq 0}^{n \times \ell(\ell+1)/2}$ where

$$A_{i,\pi(m,m')} = \begin{cases} \sqrt{\exp(q_i^\top k_{m'}) / \sum_{s=1}^n \exp(q_i^\top k_s)} & \text{for } i = m \\ 0 & \text{otherwise,} \end{cases}$$

and

$$B_{j,\pi(m,m')} = \begin{cases} \sqrt{\exp(q_m^\top k_j) / \sum_{s=1}^n \exp(q_m^\top k_s)} & \text{for } j = m' \\ 0 & \text{otherwise.} \end{cases}$$

Let $W' = AB^\top$. Then

$$\begin{aligned} W'_{ij} &= A_i^\top B_j \\ &= \sum_{\pi(m,m')} A_{i,\pi(m,m')} B_{j,\pi(m,m')} \\ &= A_{i,\pi(i,j)} B_{j,\pi(i,j)} \\ &= \exp(q_i^\top k_j) / \sum_{s=1}^n \exp(q_i^\top k_s). \end{aligned}$$

Finally, because for every $i, i' \in I_{\ell'}$ we have $\|q_i - q_{i'}\|_2 \le \delta$ and $\|k_i - k_{i'}\|_2 \le \delta$, we have exactly the same setup as in the proof of Proposition 7. Therefore the exact same proof gives $1-\gamma \le W_{ij}/W'_{ij} \le 1+\gamma$ for all i, j, where $\gamma = 17\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}$.

■

Proof of Corollary 1. In the proof of Proposition 7, we constructed a partition $\mathcal{I} = \{I_1, \ldots, I_\ell\}$ with $\ell = (4\max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}/\delta)^{2d_{kq}}$ that satisfies for every $i, i' \in I_{\ell'}$ we have $\|q_i - q_{i'}\|_2 \le \delta$ and $\|k_i - k_{i'}\|_2 \le \delta$. The result then follows from Proposition 6. ■

A.7 Pseudo-code of Main Algorithm

Below we give the pseudo-code of our main algorithm.

Algorithm 9: Main Algorithm

Input: Number of items n, maximum number of recommended items k, key matrix $K \in \mathbb{R}^{n\times d_{kq}}$, query matrix $Q \in \mathbb{R}^{n\times d_{kq}}$, value matrix $V \in \mathbb{R}^{n\times d_v}$, reward functions $\{f_i\}_{i=1}^n$, user vector $u \in \mathbb{R}^{d_v}$, parameter $\epsilon > 0$, ϵ-Approximate k-Nearest Neighbor oracle (ANN), attention matrix $W = \mathrm{softmax}(QK^\top) \in \mathbb{R}_+^{n\times n}$, low-rank approximation $W' \in \mathbb{R}_+^{n\times n}$ with factorization $W' = AB^\top$ and element wise guarantee $1 - \gamma \le W_{ij}/W'_{ij} \le 1 + \gamma$

Output: Solution x to Problem (Main).

```
// Phase One:  Preprocess (Algorithm 10)
```
$(\delta, \mathcal{I}, \mathcal{S}, \text{preprocessed ANN}) \leftarrow$ **Run** Algorithm 10 with inputs $(n, k, K, Q, V, \{f_i\}, \epsilon, \text{ANN})$

```
// Phase One:  Query (Algorithm 11)
```
$I \leftarrow$ **Run** Algorithm 11 with inputs $(u, k, \delta, \mathcal{I}, \mathcal{S}, \text{preprocessed ANN})$

```
// Phase Two (Algorithm 16)
```
$x \leftarrow$ **Run** Algorithm 16 with inputs $(W, W' = AB^\top, \gamma, V, u, k, I, \epsilon)$

return x

A.8 Proofs in Section 2.4.

Proof of Observation 2. Fix $S = \{s_1, \ldots, s_m\}$ and let $x \in \{0,1\}^n$ where $x_{s_1} = \cdots = x_{s_m} = 1$. We prove that the objective values of Problem (Main) and P are the same. By our construction of X_S, we get $XQ \in \mathbb{R}^{n\times d_{kq}}$ where $(XQ)_{s_j} = q_{s_j}$, and similarly for $XK \in \mathbb{R}^{n\times d_{kq}}$ and $XV \in \mathbb{R}^{n\times d_v}$. Then for $i \in S$ we get

$\mathrm{softmax}((XQ)(XK)^\top)_{i,s_\ell} = \exp(q_i^\top k_{s_\ell})/\sum_{s_{\ell'}\in S}\exp(q_i^\top k_{s_{\ell'}})$. Therefore, for $i \in S$ we have

$$\begin{aligned}
\frac{(w_i \odot Vu)^\top x}{w_i^\top x} &= \frac{\sum_{s_\ell\in S} w_{i,s_\ell} Vu_{s_\ell}}{\sum_{s_\ell\in S} w_{i,s_\ell}} \\
&= \frac{\sum_{s_\ell\in S}(\exp(q_i^\top k_{s_\ell})/\sum_{j\in[n]}\exp(q_i^\top k_j))(V_{s_\ell}^\top u)}{\sum_{s_\ell\in S}(\exp(q_i^\top k_{s_\ell})/\sum_{j\in[n]}\exp(q_i^\top k_j))} \\
&= \frac{\sum_{s_\ell\in S}(\exp(q_i^\top k_{s_\ell})/\sum_{s_{\ell'}\in S}\exp(q_i^\top k_{s_{\ell'}}))(V_{s_\ell}^\top u)}{\sum_{s_\ell\in S}(\exp(q_i^\top k_{s_\ell})/\sum_{s_{\ell'}\in S}\exp(q_i^\top k_{s_{\ell'}}))} \\
&= \frac{\sum_{s_\ell\in S}\mathrm{softmax}((XQ)(XK)^\top)_{i,s_\ell}(V_{s_\ell}^\top u)}{\sum_{s_\ell\in S}\mathrm{softmax}((XQ)(XK)^\top)_{i,s_\ell}} \\
&= ((\mathrm{softmax}((XQ)(XK)^\top)XV)_i)^\top u \\
&= \mathrm{SA}_{Q,K,V}(X_S)_i^\top u,
\end{aligned}$$

where the fifth equality follows since $\sum_j \mathrm{softmax}(A)_{i,j} = 1$ for every matrix A. Because this holds for every $i \in S$, the objective values of Problem (Main) and P are the same. ■

Proof of Proposition 7. Fix any $\epsilon > 0$. Let $\delta > 0$ be a parameter such that

(A.1)
$$\delta \le \frac{1}{140\max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}} \quad \text{and} \quad \epsilon \ge 34\delta\max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}(Vu)_{\max}+\delta.$$

First we partition the rows of Q and K. Because we can cover a ball with radius $\|Q\|_{2,\infty}$ in $\mathbb{R}^{d_{kq}}$ using $\lceil 4\|Q\|_{2,\infty}/\delta\rceil^{d_{kq}}$ number of balls with radius $\delta/2$ (see e.g. [61, 156, 168]), we can create a partition of the index set $[n]$ such that the size of the partition is $\lceil 4\|Q\|_{2,\infty}/\delta\rceil^{d_{kq}}$, and $\|q_i - q_{i'}\|_2 \le \delta$ for every i, i' in the same partition. [3] Similarly, we can create a partition of the index set $[n]$ such that the size of the partition is $\lceil 4\|K\|_{2,\infty}/\delta\rceil^{d_{kq}}$, and $\|k_j - k_{j'}\|_2 \le \delta$ for every j, j' in the same partition. Let $\mathcal{I} = \{I_1, \ldots, I_\ell\}$ be the product partition of the above two partitions. Then $\ell = \lceil 4\max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}/\delta\rceil^{2d_{kq}}$, and for every $\ell' \in [\ell]$ and $i, i' \in I_{\ell'}$, we have $\|q_i - q_{i'}\|_2 \le \delta$ and $\|k_i - k_{i'}\|_2 \le \delta$. Therefore, for any $i, i' \in I_{\ell'}$ and $j, j' \in I_{\ell''}$, we

[3] We can create such a partition via the packing-covering duality in the following way (see e.g. Theorem 14.1, Theorem 14.2, and Example 14.1 of [176]). First we create a maximal packing of the ball $B(\vec{0}, \|Q\|_{2,\infty}) \subset \mathbb{R}^{d_{kq}}$ using balls with radius $\delta/4$ greedily, where we greedily choose points $x_1, x_2, \ldots$ such that the balls $B(x_i, \delta/4)$ are disjoint. We stop when no more such points can be added in $B(\vec{0}, \|Q\|_{2,\infty}) \subset \mathbb{R}^{d_{kq}}$. Then, by the packing-covering duality, the balls $B(x_i, \delta/2)$ cover $B(\vec{0}, \|Q\|_{2,\infty})$. This is because any uncovered point can be added to the packing we created before, which contradicts the maximality of the packing.

have

$$\begin{aligned}|q_i^\top k_j - q_{i'}^\top k_{j'}| &= |q_i^\top (k_j - k_{j'}) + k_{j'}^\top (q_i - q_{i'})| \\ &\le |q_i^\top (k_j - k_{j'})| + |k_{j'}^\top (q_i - q_{i'})| \\ &\le \|q_i\|_2 \|k_j - k_{j'}\|_2 + \|k_{j'}\|_2 \|q_i - q_{i'}\|_2 \\ &\le 2\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}.\end{aligned}$$

Then, because $2\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\} \le 1$, we have

$$\begin{aligned}\left|\frac{\exp(q_i^\top k_j)}{\exp(q_{i'}^\top k_{j'})} - 1\right| &\le \left|\exp(|q_i^\top k_j - q_{i'}^\top k_{j'}|) - 1\right| \\ &\le |\exp(2\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}) - 1| \\ &\le 4\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\},\end{aligned}$$

where the last inequality follows by $\exp(x) \le 1 + 2x$ for $0 \le x \le 1$. Therefore, because $4\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\} \le 1/35$, we have

$$\begin{aligned}\left|\frac{w_{ij}}{w_{i'j'}} - 1\right| &= \left|\frac{\exp(q_i^\top k_j)/\sum_{m=1}^n \exp(q_i^\top k_m)}{\exp(q_{i'}^\top k_{j'})/\sum_{m'=1}^n \exp(q_{i'}^\top k_{m'})} - 1\right| \\ &\le \left|\frac{(1 + 4\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\})^2}{(1 - 4\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\})^2} - 1\right| \\ &\le \frac{4 \cdot 35^2}{34^2} \cdot 4\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\} \\ &\le 17\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\},\end{aligned}$$

where the first inequality follows since

$$\begin{aligned}1 - 4\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\} &\le \frac{\exp(q_i^\top k_j)}{\exp(q_{i'}^\top k_{j'})} \\ &\le 1 + 4\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}.\end{aligned}$$

, and the second inequality follows since $(1 + x)^2/(1 - x)^2 \le 1 + (4a^2/(a - 1)^2)x$ for every $a > 1$ and $0 \le x \le 1/a$.

Now we construct our desired index set I. Let $\{S_1, \ldots, S_\tau\}$ be a partition of the index set $[n]$ such that $f_i = f_j$ for every $\tau' \in [\tau]$ and $i, j \in S_{\tau'}$. We first construct an index set $J_{\tau'} \subset S_{\tau'}$ for each $\tau' \in [\tau]$, and then combine them to obtain I.

Fix any $\tau' \in \tau$. For each index set $I_{\ell'}$, we only choose k indices out of it to include in $J_{\tau'}$, namely the k indices that are approximately the k highest indices in $\{(Vu)_i\}_{i \in S_{\tau'} \cap I_{\ell'}}$.[4] Specifically, for each $\ell' \in [\ell]$, we run the given δ-Approximate

[4]For simplicity we assume $|S_{\tau'} \cap I_{\ell'}| \ge k$. Otherwise we simply choose all indices in $S_{\tau'} \cap I_{\ell'}$.

k-Nearest Neighbor oracle with given set of points $\bigcup_{i\in S_{\tau'}\cap I_{\ell'}}\{V_i\} \subset \mathbb{R}^{d_v}$, and query u, numbers k and δ as inputs. We let $J_{\tau'}$ be the collection of all output indices for each $\ell' \in [\ell]$. Then because $\ell = \lceil 4\max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}/\delta\rceil^{2d_{kq}}$, we have

$$|J_{\tau'}| = k\lceil 4\max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}/\delta\rceil^{2d_{kq}},$$

and the expect amortized runtime of constructing each $J_{\tau'}$ is

$$\lceil 4\max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}/\delta\rceil^{2d_{kq}} \cdot k\text{-ANN}(|S_{\tau'}|, d_v, k, \delta).$$

Let $I = \cup_{\tau'\in[\tau]} J_{\tau'}$. Then we have

$$|I| = \tau k\lceil 4\max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}/\delta\rceil^{2d_{kq}},$$

and the expect amortized runtime of constructing I is

$$\begin{aligned}&\sum_{\tau'=1}^{\tau} \lceil 4\max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}/\delta\rceil^{2d_{kq}} \cdot k\text{-ANN}(|S_{\tau'}|, d_v, k, \delta)\\ \le\ &\lceil 4\max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}/\delta\rceil^{2d_{kq}}\tau \cdot k\text{-ANN}\left(\frac{n}{\tau}, d_v, k, \delta\right).\end{aligned}$$

The inequality follows since $\sum_{\tau'=1}^{\tau} |S_{\tau'}| = n$ and k-ANN(n, d, k, ϵ) is concave in n.

Finally, we show that $\mathrm{OPT}_{\mathsf{P}(I)} \ge (1 - g(\epsilon))\mathrm{OPT}_{\mathsf{P}} - kh(\epsilon)$ with our choice of δ. Let $i_1^*, \ldots, i_k^*$ be the non-zero coordinates of an optimal solution x^* to the original problem (for simplicity we assume x^* has k non-zero entries, and other cases can be handled similarly). For each $m = 1, \ldots, k$, let i_m be the index such that i_m and i_m^* are in the same $S_{\tau'} \cap I_{\ell'}$ and $(Vu)_{i_m} \ge (Vu)_{i_m^*} - \delta$. Then $f_{i_m} = f_{i_m^*}$. Let $x \in \{0,1\}^n$ such that $x_{i_m} = 1$, then x is feasible to $\mathsf{P}(I)$. We prove that the objective value of $\mathsf{P}(I)$ at x is at least $(1 - g(\epsilon))\mathrm{OPT}_{\mathsf{P}} - kh(\epsilon)$. Indeed, because $\epsilon \ge 34\delta\max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}(Vu)_{\max} + \delta$, then for each $m = 1, \ldots, k$ we have

$$
\begin{aligned}
& f_{i_m}\left(\frac{(w_{i_m} \odot Vu)^\top x}{w_{i_m}^\top x}\right) \\
= \quad & f_{i_m}\left(\frac{\sum_{j=1}^k (w_{i_m})_{i_j}(Vu)_{i_j}}{\sum_{j=1}^k (w_{i_m})_{i_j}}\right) \\
\geq \quad & f_{i_m}\left(\frac{\sum_{j=1}^k (w_{i_m})_{i_j}(Vu)_{i_j^*}}{\sum_{j=1}^k (w_{i_m})_{i_j}} - \delta\right) \\
\geq \quad & f_{i_m^*}\left(\frac{(1-\epsilon)\sum_{j=1}^k (w_{i_m^*})_{i_j^*}(Vu)_{i_j^*}}{(1+\epsilon)\sum_{j=1}^k (w_{i_m^*})_{i_j^*}} - \delta\right) \\
\geq \quad & f_{i_m^*}\left(\frac{\sum_{j=1}^k (w_{i_m^*})_{i_j^*}(Vu)_{i_j^*}}{\sum_{j=1}^k (w_{i_m^*})_{i_j^*}} - 34\delta \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}(Vu)_{\max} - \delta\right) \\
\geq \quad & f_{i_m^*}\left(\frac{\sum_{j=1}^k (w_{i_m^*})_{i_j^*}(Vu)_{i_j^*}}{\sum_{j=1}^k (w_{i_m^*})_{i_j^*}} - \epsilon\right) \\
\geq \quad & (1-g(\epsilon)) f_{i_m^*}\left(\frac{\sum_{j=1}^k (w_{i_m^*})_{i_j^*}(Vu)_{i_j^*}}{\sum_{j=1}^k (w_{i_m^*})_{i_j^*}}\right) - h(\epsilon),
\end{aligned}
$$

where the first inequality follows since $(Vu)_{i_j} \geq (Vu)_{i_j^*} + \delta$, the second inequality follows since $1-\epsilon \leq (w_{i_m^*})_{i_j^*}/(w_{i_m})_{i_j} \leq 1+\epsilon$ for every i_m, i_m^* that are in the same partition in $\mathcal{I}$ and i_j, i_j^* that are in the same partition in $\mathcal{I}$, and the third inequality follows since $(1-\epsilon)/(1+\epsilon) \geq 1-2\epsilon$ and $\sum_{j=1}^k (w_{i_m^*})_{i_j^*}(Vu)_{i_j^*} / \sum_{j=1}^k (w_{i_m^*})_{i_j^*} \leq (Vu)_{\max}$. Then

$$
\begin{aligned}
\mathrm{OPT}_{\mathsf{P}(I)} \quad & \geq \quad \sum_{m=1}^k x_{i_m} f_{i_m}\left(\frac{(w_{i_m} \odot Vu)^\top x}{w_{i_m}^\top x}\right) \\
& \geq \quad \sum_{m=1}^k x_{i_m^*}^* \left(g(\epsilon) f_{i_m^*}\left(\frac{\sum_{j=1}^k (w_{i_m^*})_{i_j^*}(Vu)_{i_j^*}}{\sum_{j=1}^k (w_{i_m^*})_{i_j^*}}\right) - h(\epsilon)\right) \\
& \geq \quad (1-g(\epsilon))\mathrm{OPT}_{\mathsf{P}} - k h(\epsilon)
\end{aligned}
$$

as desired. By our choice of δ, we have

$$
|I| = k\left\lceil \frac{140(\max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\})^2}{(Vu)_{\max} \cdot \epsilon} \right\rceil^{2d_{kq}},
$$

and the expected amortized runtime of finding I is

$$
\left\lceil \frac{140\big(\max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}\big)^2 (Vu)_{\max}}{\epsilon} \right\rceil^{2d_{kq}} \tau \\
\cdot \; k\text{-ANN}\left(\frac{n}{\tau},\, d_v,\, k,\, \frac{\epsilon}{35\, \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}\, (Vu)_{\max}}\right).
$$

■

Below we give the pseudo-code of our phase one algorithm in Proposition 7.

Algorithm 10: Phase One: Preprocess

`// Preprocess`

Input: Number of items n, maximum number of recommended items k, key matrix $K \in \mathbb{R}^{n \times d_{kq}}$, query matrix $Q \in \mathbb{R}^{n \times d_{kq}}$, value matrix $V \in \mathbb{R}^{n \times d_v}$, reward functions $\{f_i\}_{i=1}^n$, parameter $\epsilon > 0$, and ϵ-Approximate k-Nearest Neighbor oracle

Output: Parameter $\delta > 0$, partitions $\mathcal{I} = \{I_1, \ldots, I_\ell\}$ and $\mathcal{S} = \{S_1, \ldots, S_\tau\}$, and preprocessed ϵ-Approximate k-Nearest Neighbor oracle with set of points $\{V_i : i \in I_{\ell'} \cap S_{\tau'}\}$ for each $\tau' \in [n]$ and $\ell' \in [\ell]$

Set $\delta > 0$ according to Eq. (A.1)

Let $R \leftarrow \max\{\|Q\|_{2,\infty}, \|K\|_{2,\infty}\}$

Form a maximal $(\delta/2)$-separated set $C \subset B(0, R)$

Set balls $\{B_i\}_{i=1}^m \leftarrow \{B(c, \delta/2) : c \in C\}$ with $m = |C|$

Create partition $\mathcal{I} = \{I_1, \ldots, I_\ell\}$ of $[n]$: each I_j contains indices whose (K_i, Q_i) fall in the same pair of balls

Let τ be the number of distinct functions among $f_1, \ldots, f_n$

Create partition $\mathcal{S} = \{S_1, \ldots, S_\tau\}$ of $[n]$ where $f_i = f_j$ for all $i, j \in S_{\tau'}$

for $\tau' = 1$ **to** τ **do**
  for $\ell' = 1$ **to** ℓ **do**
    if $I_{\ell'} \cap S_{\tau'} \neq \emptyset$ **then**
      Preprocess the ϵ-Approximate k-Nearest Neighbor oracle with set of points $\{V_i : i \in I_{\ell'} \cap S_{\tau'}\}$
    end
  end
end

return *Parameter $\delta > 0$, partitions $\mathcal{I} = \{I_1, \ldots, I_\ell\}$ and $\mathcal{S} = \{S_1, \ldots, S_\tau\}$, and preprocessed ϵ-Approximate k-Nearest Neighbor oracle with set of points $\{V_i : i \in I_{\ell'} \cap S_{\tau'}\}$ for each $\tau' \in [n]$ and $\ell' \in [\ell]$*

Algorithm 11: Phase One: Query

`// Query`

Input: User vector $u \in \mathbb{R}^{d_v}$, maximum number of recommended items k, outputs of Algorithm 10: parameter $\delta > 0$, partitions $\mathcal{I} = \{I_1, \ldots, I_\ell\}$ and $\mathcal{S} = \{S_1, \ldots, S_\tau\}$, and preprocessed ϵ-Approximate k-Nearest Neighbor oracle with set of points $\{V_i : i \in I_{\ell'} \cap S_{\tau'}\}$ for each $\tau' \in [n]$ and $\ell' \in [\ell]$

Output: Index set I of candidate items

Initialize $I \leftarrow \emptyset$
for $\tau' = 1$ **to** τ **do**
  for $\ell' = 1$ **to** ℓ **do**
    if $I_{\ell'} \cap S_{\tau'} \neq \emptyset$ **then**
      $J \leftarrow$ Query the ϵ-Approximate k-Nearest Neighbor oracle with set of points $\{V_i : i \in I_{\ell'} \cap S_{\tau'}\}$, query u, and numbers k and δ as inputs
      $I \leftarrow I \cup J$
    end
  end
end
return I

A.9 Proof of Proposition 8.

The proof is completed according to the following steps:

1. **Low Non-negative Rank Approximation:** In Lemma 7, we prove that in order to approximately solve $\mathsf{P}(I)$, it is sufficient to approximately solve $\mathsf{P}'(I)$, where $\mathsf{P}'(I)$ is obtained by replacing W with W'.
2. **Enumeration of Partial Solutions:** We took guesses on some index sets $X_1, \ldots, X_{r_+}$, which corresponds to the non-negative factorization of W'. Let $X = (X_1, \ldots, X_{r_+})$ and let $\mathsf{P}(X)$ denote the problem where $\mathsf{P}'(I)$ has additional constraints that $x_i = 1$ for all $i \in \cup_j X_j$. We showed that the total number of guesses X is bounded above, so it is sufficient to solve $\mathsf{P}(X)$ for each guess X.
3. **Linearization of Fractional Objective Terms:** In order to solve $\mathsf{P}(X)$, we linearize the fractional terms in the objective function by defining a set of auxiliary problems $\mathsf{P}(X, t)$ for each $t \in \mathbb{R}_+^m$. These problems are parameterized by the denominator terms in the objective function of $\mathsf{P}(X)$. In Lemma 9, we prove it suffices to find a t^* for which $\mathsf{P}(X, t^*)$ has the highest optimal value

among all $\mathsf{P}(X,t)$'s, as the corresponding optimal x^* is an optimal solution to $\mathsf{P}(X)$.

4. **Dimensionality Reduction and Discretization of Auxiliary Problems:** In order to approximately solve $\mathsf{P}(X,t)$ for all $t \in \mathbb{R}^m_+$, we discretize t-space and show in Lemma 10 that it suffices to solve $\mathsf{P}(X,t)$ for a small number of t's.
5. **Complete Linearization of Auxiliary Problems:** Fix a given t, the objective functions of $\mathsf{P}(X,t)$ inside f has rank r_+. We discretized the value space of those objective functions. In Lemma 11, we showed that in order to solve $\mathsf{P}(X,t)$, it is sufficient to give an oracle that, for each discretization of the value space, identify whether there exists a feasible solution to $\mathsf{P}(X,t)$ with objective values that are approximately inside the discretization.
6. **Approximation of Linearized Auxiliary Problems via LP Rounding:** Finally, we gave such an oracle by a rounding procedure. Lemma 12 and Lemma 13 proved that the oracle is correct by using the properties of our guess X.

Step 1: Low Non-negative Rank Approximation

First we bound the loss incurred by replacing W with W':

Lemma 7. *Let Problem* $\mathsf{P}'(I)$ *be defined as*

$$(\mathsf{P}'(I)) \qquad \max \quad f_{\mathsf{P}'(I)} = \sum_{i=1}^{m} x_i f_i \left(\frac{(w_i' \odot Vu)^\top x}{w_i'^\top x} \right)$$
$$\text{s.t.} \quad x \in \{0,1\}^m,\ 1 \le e^\top x \le k.$$

Let x be a feasible solution to $\mathsf{P}'(I)$ (and hence also a feasible solution to $\mathsf{P}(I)$), and suppose x satisfies

$$f_{\mathsf{P}'(I)}(x) \ge (1-\alpha)\mathrm{OPT}_{\mathsf{P}'(I)} - \beta.$$

Then we have

$$\begin{aligned} f_{\mathsf{P}(I)}(x) \ge\ & (1-\alpha)(1-g(2\gamma(Vu)_{\max}))^2 \mathrm{OPT}_{\mathsf{P}(I)} \\ & - kh(2\gamma(Vu)_{\max})(1+(1-\alpha)(1-g(2\gamma(Vu)_{\max})) \\ & - \beta(1-g(2\gamma(Vu)_{\max})). \end{aligned}$$

Proof of Lemma 7. Let x be a feasible solution to $\mathsf{P}'(I)$. Because $1-\gamma \le W_{ij}/W'_{ij} \le 1+\gamma$, we have

$$\frac{(w_i' \odot Vu)^\top x}{w_i'^\top x} \ge \frac{(1-\gamma)}{(1+\gamma)} \frac{(w_i \odot Vu)^\top x}{w_i{}^\top x} \ge (1-2\gamma) \frac{(w_i \odot Vu)^\top x}{w_i{}^\top x}.$$

Therefore, for any feasible solution x, we have

$$\begin{aligned} f_{\mathsf{P}'(I)}(x) &= \sum_{i=1}^{m} x_i f_i \left(\frac{(w_i' \odot Vu)^\top x}{w_i'^\top x} \right) \\ &\geq \sum_{i=1}^{m} x_i f_i \left((1-2\gamma) \frac{(w_i \odot Vu)^\top x}{w_i^\top x} \right) \\ &\geq \sum_{i=1}^{m} x_i f_i \left(\frac{(w_i \odot Vu)^\top x}{w_i^\top x} - 2\gamma (Vu)_{\max} \right) \\ &\geq (1 - g(2\gamma(Vu)_{\max}) f_{\mathsf{P}(I)}(x) - kh(2\gamma(Vu)_{\max}), \end{aligned} \tag{A.2}$$

where the third inequality follows since $(w_i \odot Vu)^\top x^* / w_i^\top x^* \leq (Vu)_{\max}$.

Similarly, we also have

$$\frac{(w_i \odot Vu)^\top x}{w_i{}^\top x} \geq (1-2\gamma) \frac{(w_i' \odot Vu)^\top x}{w_i'^\top x},$$

which gives

$$f_{\mathsf{P}(I)}(x) \geq (1 - g(2\gamma(Vu)_{\max})) f_{\mathsf{P}'(I)}(x) - kh(2\gamma(Vu)_{\max}). \tag{A.3}$$

Now Let $x^*_{\mathsf{P}(I)}$ be an optimal solution to $\mathsf{P}(I)$. Then by Eq. (A.2), we have

$$\begin{aligned} \mathrm{OPT}_{\mathsf{P}'(I)} &\geq f_{\mathsf{P}'(I)}(x^*_{\mathsf{P}'(I)}) \\ &\geq (1 - g(2\gamma(Vu)_{\max})) f_{\mathsf{P}(I)}(x^*_{\mathsf{P}(I)}) - kh(2\gamma(Vu)_{\max}) \\ &= (1 - g(2\gamma(Vu)_{\max})) \mathrm{OPT}_{\mathsf{P}(I)} - kh(2\gamma(Vu)_{\max}). \end{aligned}$$

Finally, applying Eq. (A.3), we conclude that

$$\begin{aligned} f_{\mathsf{P}(I)}(x) &\geq (1 - g(2\gamma(Vu)_{\max})) f_{\mathsf{P}'(I)}(x) - kh(2\gamma(Vu)_{\max}) \\ &\geq (1-\alpha)(1 - g(2\gamma(Vu)_{\max})) \mathrm{OPT}_{\mathsf{P}'(I)} - kh(2\gamma(Vu)_{\max}) \\ &\quad - \beta(1 - g(2\gamma(Vu)_{\max})) \\ &\geq (1-\alpha)(1 - g(2\gamma(Vu)_{\max}))^2 \mathrm{OPT}_{\mathsf{P}(I)} \\ &\quad - kh(2\gamma(Vu)_{\max})(1 + (1-\alpha)(1 - g(2\gamma(Vu)_{\max})) \\ &\quad - \beta(1 - g(2\gamma(Vu)_{\max})). \end{aligned}$$

■

Lemma 7 shows that, in order to approximately solve $\mathsf{P}(I)$, it is enough to approximately solve $\mathsf{P}'(I)$.

Let $W' = AB^\top$ be the known non-negative factorization, where $A, B \in \mathbb{R}_{\geq 0}^{m \times r_+}$. Let $a_i^\top \in \mathbb{R}_{\geq 0}^{r_+}$ be the i-th row of A and $b_j \in \mathbb{R}_{\geq 0}^{m}$ be the j-th column of B. Then $w'_i = \sum_{j=1}^{r_+} a_{ij} b_j$. Let

$$d_j = b_j \odot (Vu). \tag{A.4}$$

Then $\mathsf{P}'(I)$ can be rewritten as:

$$(\mathsf{P}'(I)) \qquad \max \quad f_{\mathsf{P}'(I)} = \sum_{i=1}^{m} x_i f_i \left(\frac{\sum_{j=1}^{r_+} a_{ij} d_j^\top x}{w_i'^\top x} \right)$$
$$\text{s.t.} \quad x \in \{0,1\}^m, \ 1 \leq e^\top x \leq k.$$

Step 2: Enumeration of Partial Solutions

Our algorithm enumerates a set of partial solutions (where a "partial solution" fixes the values of a subset of variables), and then for each partial solution, solves the remaining problem near-optimally. In this step we bound the total number of partial solutions, and in the next steps we show that for each partial solution, the remaining problem can be solved sufficiently fast.

Let

$$\lambda = \lceil (2r_+ + 2)(Vu)_{\max} / \epsilon \rceil. \tag{A.5}$$

Each partial solution that our algorithm considers is specified by a tuple $(X_1, \ldots, X_{r_+})$, where each $X_j \subset [m]$ is an index set such that $1 \leq |X_1| = \cdots = |X_{r_+}| \leq \lambda$. For each $j \in [r_+]$, let

$$\hat{X}_j = \{ i \in [m] \setminus X_j \mid d_{ji} > \min_{i' \in X_j} \{ d_{ji'} \} \}. \tag{A.6}$$

In words, $\hat{X}_j$ consists of the indices outside of X_j whose coefficients in d_j are strictly greater than the minimum coefficient in d_j across indices in X_j.

We say a tuple $(X_1, \ldots, X_{r_+})$, with corresponding index sets $\hat{X}_1, \ldots, \hat{X}_{r_+}$ defined according to (A.6), is **valid** if $|\cup_j X_j| \leq k$ and $(\cup_j X_j) \bigcap (\cup_j \hat{X}_j) = \emptyset$. Then every feasible solution z to $\mathsf{P}'(I)$ **corresponds** to a valid tuple $(X_1, \ldots, X_{r_+})$ in the following way: Let $Z = \{ i \in [m] \mid z_i = 1 \}$. For each $j \in [r_+]$, we define $X_j \subset Z$ to be the set of indices $i \in Z$ such that d_{ji} is among the $\min\{\lambda, |Z|\}$ highest values in Z. That is, let $\pi_j : [|Z|] \to Z$ be a sorting of Z according to d_j such that $d_{j,\pi_j(1)} \geq \cdots \geq d_{j,\pi_j(|Z|)}$. Then $X_j = \{ \pi_j(1), \ldots, \pi_j(\min\{\lambda, |Z|\}) \}$. Notice that $|Z| \leq k$, so $|\cup_j X_j| \leq k$. Also, we claim that $Z \cap \hat{X}_j = \emptyset$ for each $j \in [r_+]$. Supposing otherwise that $i \in Z \cap \hat{X}_j$, then

by construction $d_{ji} > d_{j,\pi_j(\min\{\lambda,|Z|\})}$. Because $i \in Z$, we have that i is in the image of π_j, so there exists i' such that $\pi_j(i') = i$. Therefore $\pi_j(i') \geq \pi_j(\min\{\lambda, |Z|\})$, which shows $i \in X_j$. This contradicts $X_j \cap \hat{X}_j = \emptyset$. Therefore $Z \cap \hat{X}_j = \emptyset$ for each $j \in [r_+]$. Hence we have $(\cup_j X_j) \bigcap (\cup_j \hat{X}_j) = \emptyset$. Therefore $(X_1, \ldots, X_{r_+})$ is indeed a valid tuple.

The notion of correspondence to valid tuples forms a partition of the set of feasible solutions to $\mathsf{P}'(I)$, so it suffices to solve $\mathsf{P}'(I)$ separately for each subset of this partition. This is formally stated in the following result:

Lemma 8. *Suppose we are given an oracle* ALG′ *that takes* $\mathsf{P}'(I)$*, any valid tuple* $(X_1, \ldots, X_{r_+})$*, and any* $\delta > 0$ *as inputs, and outputs a solution* $x'_{(X_1,\ldots,X_{r+})}$ *of* $\mathsf{P}'(I)$ *that satisfies*

1. $x'_{(X_1,\ldots,X_{r+})}$ *corresponds to* $(X_1, \ldots, X_{r_+})$*, and*
2. *for any* $x_{(X_1,\ldots,X_{r+})}$ *that corresponds to* $(X_1, \ldots, X_{r_+})$*, we have*

$$f_{\mathsf{P}'(I)}(x'_{(X_1,\ldots,X_{r+})}) \geq (1 - g'(\delta)) f_{\mathsf{P}'(I)}(x_{(X_1,\ldots,X_{r+})}) - h'(\delta),$$

where $0 \leq g'(\delta) \leq 1$ *and* $h'(\delta) \geq 0$*,*

with runtime $T(\delta)$*. Then there exists an algorithm* ALG *that takes* $\mathsf{P}'(I)$ *and any* $\delta > 0$ *as inputs, and outputs a solution of* $\mathsf{P}'(I)$ *that satisfies*

$$\mathrm{ALG}_{\mathsf{P}'(I)} \geq (1 - g'(\delta))\mathrm{OPT}_{\mathsf{P}'(I)} - h'(\delta)$$

with runtime

$$r_+ m \log_2 m + \lambda m^\lambda + \lambda r_+^2 m^{\lambda r_+} + m^{\lambda r_+} T(\delta).$$

Proof of Lemma 8. We will construct ALG explicitly. Now because every feasible solution z to $\mathsf{P}'(I)$ corresponds to exactly one valid tuple, we can solve $\mathsf{P}'(I)$ by enumerating all valid tuples, and applying ALG′ to each valid tuple. It turns out that pre-sorting the vectors d_j allows for more-efficient enumeration. Let ALG take the following steps:

1. **Sort d_j for each $j \in [r_+]$.** This takes runtime $r_+ m \log_2 m$.
2. **Enumerate all valid tuples with $|X_1| < \lambda$, and record the unique corresponding feasible solutions.** We will show momentarily that when $|X_1| < \lambda$, there is a unique corresponding feasible solution, and as a result this step takes runtime λm^λ.

3. **Enumerate all valid tuples with $|X_1| = \lambda$, and record the solution output by $\text{ALG}'_{\mathsf{P}'(I)}$ for each such valid tuple.** We will show that it takes runtime $\lambda r_+^2 m^{\lambda r_+}$ to enumerate all such valid tuples, and then because there are at most $m^{\lambda r_+}$ such valid tuples, the total runtime of this step is $\lambda r_+^2 m^{\lambda r_+} + m^{\lambda r_+} T(\delta)$.
4. **Output a solution that is recorded with the highest objective value in $\mathsf{P}'(I)$.**

Therefore the total runtime of ALG is

$$r_+ m \log_2 m + \lambda m^\lambda + \lambda r_+^2 m^{\lambda r_+} + m^{\lambda r_+} T(\delta).$$

Finally, let $x^*_{(X_1^*,\ldots,X_{r+}^*)}$ be an optimal solution of $\mathsf{P}'(I)$ where $(X_1^*,\ldots,X_{r+}^*)$ is the valid tuple that it corresponds to. Let $x'_{(X_1^*,\ldots,X_{r+}^*)}$ be the solution that $\text{ALG}'_{\mathsf{P}'(I)}$ outputs with input $\mathsf{P}'(I)$, valid tuple $(X_1^*,\ldots,X_{r_+}^*)$, and $\delta > 0$. Then

$$\begin{aligned}
\text{ALG}_{\mathsf{P}'(I)} &\geq f_{\mathsf{P}'(I)}(x'_{(X_1^*,\ldots,X_{r+}^*)}) \\
&\geq (1 - g'(\delta)) f_{\mathsf{P}'(I)}(x^*_{(X_1^*,\ldots,X_{r+}^*)}) - h'(\delta) \\
&= (1 - g'(\delta))\text{OPT}_{\mathsf{P}'(I)} - h'(\delta).
\end{aligned}$$

It remains to analyze Steps 2 and 3 of ALG.

Step 2: Fix any valid tuple $(X_1,\ldots,X_{r_+})$ such that $|X_1| < \lambda$. Assume there exists a feasible solution z to $\mathsf{P}'(I)$ that corresponds to the valid tuple, and let $Z = \{i \in [m] \mid z_i = 1\}$. Then because $|X_1| = \min\{\lambda, |Z|\} = |Z|$ and $X_1 \subset Z$, we have $X_1 = Z$. Similarly, $X_j = Z$ for all $j \in [r_+]$. Therefore, there exists a feasible solution to $\mathsf{P}'(I)$ that corresponds to $(X_1,\ldots,X_{r_+})$ only if $X_1 = \cdots = X_{r+}$. There are at most $\sum_{i=1}^{\lambda-1} \binom{m}{i} \leq \lambda m^\lambda$ such tuples. Moreover, there is a unique feasible solution z that corresponds to $(X_1,\ldots,X_{r_+})$, namely $z_i = 1$ for every $i \in X_1$ and $z_i = 0$ for every $i \notin X_1$. Thus, the runtime of enumerating all corresponding feasible solutions is bounded by λm^λ.

Step 3: Fix any valid tuple $(X_1,\ldots,X_{r_+})$ such that $|X_1| = \lambda$. Then by construction we must have $z_i = 1$ for every $i \in \cup_j X_j$, and $z_i = 0$ for every $i \in \cup_j \hat{X}_j$. Therefore every feasible solution z that corresponds to $(X_1,\ldots,X_{r_+})$ must lie in the following set:

$$\begin{aligned}
\{z \in \{0,1\}^m \mid & z_i = 1 \ \forall i \in \cup_j X_j, \\
& z_i = 0 \ \forall i \in \cup_j \hat{X}_j, \\
& 1 \leq e^\top z \leq k\}.
\end{aligned}$$

There are $\binom{m}{\lambda}^{r_+} \le m^{\lambda r_+}$ tuples such that $|X_1| = \lambda$. We enumerate all such tuples, and check each for validity according to the following procedure:

0. From Step 1 of ALG, let $\pi_j : [m] \to [m]$ be a sorting of $[m]$ according to d_j such that $d_{j,\pi_j(1)} \ge \cdots \ge d_{j,\pi_j(m)}$.
1. Fix any given tuple $(X_1, \ldots, X_{r_+})$. For each $j \in [r_+]$, let $i(j) \in [m]$ be an index such that $d_{j,i(j)} \in X_j$ and $d_{j,i(j)} \le d_{ji}$ for all $d_{ji} \in X_j$. Without loss of generality, if $d_{j,i(j)} = d_{ji'}$ and $d_{ji'} \notin X_j$ for some index i', we let $\pi_j(i(j)) > \pi_j(i')$ for tie-breaking in the sorting. Also, if $d_{j,i(j)} = d_{ji'}$ and $d_{ji'} \in X_j$ for some index i', we let $\pi_j(i(j)) < \pi_j(i')$ for tie-breaking in the sorting. Then by our construction

$$\begin{aligned} \hat{X}_j &= \{i \in [m] \setminus X_j \mid d_{ji} > \min_{i' \in X_j}\{d_{ji'}\}\} \\ &= \{i \in [m] \setminus X_j \mid d_{ji} > d_{j,i(j)}\}\} \\ &= \{i \in [m] \setminus X_j \mid \pi_j(i) > \pi_j(i(j))\}. \end{aligned}$$

   Therefore $X_j \cup \hat{X}_j = \{i \in [m] \mid \pi_j(i) \ge \pi_j(i(j))\}$.
2. Note that $\sum_{j=1}^{r_+} |X_j| = \lambda r_+$. Therefore, in order to check whether $|\cup_j X_j| \le k$, we just need to count the number of overlaps among all X_j's. Set a counter $c = 0$ to count the overlaps. For each $j \in [r_+]$ and for each $i \in X_j$, we check if $\pi_{j'}(i) \ge \pi_{j'}(i(j'))$, and do the following:
   - If $\pi_{j'}(i) < \pi_{j'}(i(j'))$, then we have $i \notin X_{j'} \cup \hat{X}_{j'}$. We do nothing in this case.
   - If $\pi_{j'}(i) \ge \pi_{j'}(i(j'))$ and $\pi_{j'}(i) \in X_{j'}$, then i appears in both X_j and $X_{j'}$. Therefore we increase c by 1.
   - If $\pi_{j'}(i) \ge \pi_{j'}(i(j'))$ and $\pi_{j'}(i) \notin X_{j'}$, then we must have $\pi_{j'}(i) \in \hat{X}_{j'}$. Therefore $(\cup_j X_j) \bigcap (\cup_j \hat{X}_j) \ne \emptyset$, so we can terminate the process and declare that $(X_1, \ldots, X_{r_+})$ is not a valid tuple.

   We iterate all $j \in [r_+]$ and $i \in X_j$. Notice that c counts the number of overlaps (with multiplicity) of elements in X_j, we have $|\cup_j X_j| = \sum_{j=1}^{r_+} |X_j| - c = \lambda r_+ - c$. Therefore we can check if $|\cup_j X_j| \le k$. Also, if the above procedure never encounters $\pi_{j'}(i) \in \hat{X}_{j'}$, then we have $(\cup_j X_j) \bigcap (\cup_j \hat{X}_j) = \emptyset$. Therefore this procedure allows us to check the validity of $(X_1, \ldots, X_{r_+})$.

Because each $d_{j'}$ is sorted, the above procedure takes a constant runtime for each $j' \in [r_+]$, so the runtime for each fixed $j \in [r_+]$ and $i \in X_j$ is r_+. Because

$\sum_{j=1}^{r_+} |X_j| = \lambda r_+$, there are λr_+ combinations of $j \in [r_+]$ and $i \in X_j$. Therefore the runtime to check the validity is λr_+^2 for any given tuple $(X_1, \ldots, X_{r_+})$. Because there are at most $m^{\lambda r_+}$ such tuples, the runtime of enumerating all such tuples is $\lambda r_+^2 m^{\lambda r_+}$.

■

Below we give the pseudo-code of the algorithm ALG in Lemma 8.

Algorithm 12: Phase Two: Enumeration of Partial Solutions

Input: Instance of Problem $\mathsf{P}'(I)$ from Lemma 7, oracle ALG$'$ from Lemma 8, parameter $\delta > 0$

Output: Solution x to $\mathsf{P}(I)$

Set $\lambda > 0$ according to Eq. (A.5)

`// Enumerate valid tuples with` $|X_j| < \lambda$

for $\lambda' = 1$ **to** $\lambda - 1$ **do**
  for *each* $X_1 \subset [m]$ *with* $1 \le |X_1| \le k$ **do**
    Set $x \in \{0,1\}^m$ where $x_i = 1 \iff i \in X_1$; record $(x, f_{\mathsf{P}'(I)}(x))$
  end
end

`// Enumerate valid tuples with` $|X_j| = \lambda$

for $j \in [m]$ **do**
  $d_j \leftarrow b_j \odot (Vu)$
  $\pi_j \leftarrow$ permutation sorting $[m]$ by d_j descending
end
for *each tuple* $(X_1, \ldots, X_{r_+})$ *where* $|X_j| = \lambda$ *for all* j **do**
  for $j \in [r_+]$ **do**
    $i(j) \leftarrow \arg\min_{i \in X_j} d_{j,i}$
    $\hat{X}_j \leftarrow \{i \in [m] \setminus X_j : \pi_j(i) > \pi_j(i(j))\}$
  end
  $c \leftarrow 0$; *valid* $\leftarrow$ **true**
  for $j \in [r_+]$ *and* $i \in X_j$ **do**
    for $j' \in [r_+]$ **do**
      if $\pi_{j'}(i) \le \pi_{j'}(i(j'))$ **then** $c \leftarrow c + 1$
      if $\pi_{j'}(i) \le \pi_{j'}(i(j'))$ *and* $i \notin X_{j'}$ **then** *valid* $\leftarrow$ **false**; **break**
    end
    if *not valid* **then break**
  end
  if *valid and* $c \le \lambda r_+ - k$ **then**
    Let x be the output of ALG$'$ with inputs $\mathsf{P}(X)$ where valid tuple $X = (X_1, \ldots, X_{r_+})$, and parameter δ
    record $(x, f_{\mathsf{P}'(I)}(x))$
  end
end
return *recorded* x *with maximum* $f_{\mathsf{P}'(I)}(x)$

The consequence of this result is that we have reduced to the task of, for each valid tuple $X = (X_1, \ldots, X_{r_+})$ with $|X_1| = \lambda$, solving $\mathsf{P}'(I)$ with the additional constraint that the solution must correspond to the valid tuple, as stated in Problem $\mathsf{P}(X)$ below:

$$
\begin{aligned}
(\mathsf{P}(X)) \qquad \max \quad & f_{\mathsf{P}(X)}(x) = \sum_{i=1}^{m} x_i f_i \left(\frac{\sum_{j=1}^{r_+} a_{ij} d_j^\top x}{w_i'^\top x} \right) \\
\text{s.t.} \quad & x \in \{0,1\}^m, \\
& 1 \le e^\top x \le k, \\
& x_i = 1 \quad \forall i \in \cup_j X_j, \\
& x_i = 0 \quad \forall i \in \cup_j \hat{X}_j.
\end{aligned}
$$

Step 3: Linearization

In order to solve $\mathsf{P}(X)$, we define the following auxiliary problem $\mathsf{P}(X,t)$ for each $t \in \mathbb{R}_+^m$:

$$
\begin{aligned}
(\mathsf{P}(X,t)) \qquad \max \quad & f_{\mathsf{P}(X,t)}(x) = \sum_{i=1}^{m} x_i f_i \left(\frac{\sum_{j=1}^{r_+} a_{ij} d_j^\top x}{t_i} \right) \\
\text{s.t.} \quad & x \in \{0,1\}^m, \\
& e^\top x \le k, \\
& w_i'^\top x \le t_i \quad \forall i \in [m], \\
& x_i = 1 \quad \forall i \in \cup_j X_j, \\
& x_i = 0 \quad \forall i \in \cup_j \hat{X}_j.
\end{aligned}
$$

Note that we have dropped the constraint $1 \le e^\top x$ in $\mathsf{P}(X,t)$. This is inconsequential: because we assume $\mathrm{OPT}_{\mathsf{P}}$ is positive, the solution x with all entries equal to zero is not an optimal solution to P. Indeed, the only reason we have maintained the $1 \le e^\top x$ constraint until now has been to rule out notational edge cases (such as dividing by zero).

We prove an important property regarding the relationship between optimal solutions of $\mathsf{P}(X)$ and those of $\mathsf{P}(X,t)$.

Lemma 9. *Fix any valid tuple X. Let $t^* \in \arg\max_{t \in \mathbb{R}_+^m} \mathrm{OPT}_{\mathsf{P}(X,t)}$, and let x^* be an optimal solution to $\mathsf{P}(X,t^*)$. Then $W'x^* = t^*$, and x^* is an optimal solution to $\mathsf{P}(X)$.*

Proof of Lemma 9. First, we show that $W'x^* = t^*$. Suppose otherwise, and let $t' = W'x^*$. Then $t'_i \le t^*_i$ for every $i \in [m]$ and $t'_i < t^*_i$ for some i. Therefore

$$\mathrm{OPT}_{\mathsf{P}(X,t^*)} = \sum_{i=1}^{m} x^*_i f_i\left(\frac{\sum_{j=1}^{r_+} a_{ij} d_j^\top x^*}{t^*_i}\right) < \sum_{i=1}^{m} x^*_i f_i\left(\frac{\sum_{j=1}^{r_+} a_{ij} d_j^\top x^*}{t'_i}\right) \le \mathrm{OPT}_{\mathsf{P}(X,t')},$$

contradicting the definition of t^*.

Now we show that x^* is an optimal solution to the $\mathsf{P}(X)$. For the sake of contradiction, suppose that x' gives a higher objective value than x^* to $\mathsf{P}(X)$, that is,

$$\sum_{i=1}^{m} x'_i f_i\left(\frac{\sum_{j=1}^{r_+} a_{ij} d_j^\top x'}{w_i'^\top x'}\right) > \sum_{i=1}^{m} x^*_i f_i\left(\frac{\sum_{j=1}^{r_+} a_{ij} d_j^\top x^*}{w_i'^\top x^*}\right) = \mathrm{OPT}_{\mathsf{P}(X,t^*)}.$$

Let $t' = Wx'$. Then x' is a feasible solution to $P(X, t')$. Therefore, we have

$$\mathrm{OPT}_{\mathsf{P}(X,t')} \ge \sum_{i=1}^{m} x'_i f_i\left(\frac{\sum_{j=1}^{r_+} a_{ij} d_j^\top x'}{w_i'^\top x'}\right) > \mathrm{OPT}_{\mathsf{P}(X,t^*)},$$

contradicting the definition of t^*. ■

Because $t^* = W'x^*$ and $\sum_{j=1}^{m} w_{ij} \le 1+\gamma$ for each $i \in [m]$, we have $t^* \in [W'_{\min}, 1+\gamma]^m$. Thus, from here on we will only consider the problems $\mathsf{P}(X, t)$ with $t \in [W'_{\min}, 1+\gamma]^m$.

Step 4: Dimensionality Reduction and Discretization of Auxiliary Problems: Lemma 9 shows that, in order to solve $\mathsf{P}(X)$, it is enough to solve for

$$\arg\max_{t \in \mathbb{R}^m_+} \mathrm{OPT}_{\mathsf{P}(X,t)}.$$

Below we show that it suffices to solve the auxiliary problem $\mathsf{P}(X, t)$ for a smaller, discretized set of t's.

Lemma 10. *Suppose we are given an oracle* ALG′ *that takes* $\mathsf{P}(X, t)$ *with any* $t \in [W'_{\min}, 1+\gamma]^m$ *and any* $\delta > 0$ *as inputs, and outputs a solution of* $\mathsf{P}(X, t)$ *that satisfies*

$$\mathrm{ALG}'_{\mathsf{P}(X,t)} \ge (1 - g'(\delta))\mathrm{OPT}_{\mathsf{P}(X,t)} - h'(\delta)$$

with runtime $T(\delta)$*, where* $0 \le g'(\delta) \le 1$ *and* $h'(\delta) \ge 0$*. Then there exists an algorithm* ALG *that takes* $\mathsf{P}(X)$ *and any* $\delta > 0$ *as inputs, and outputs a solution of* $\mathsf{P}(X)$ *that satisfies*

$$\mathrm{ALG}_{\mathsf{P}(X)} \ge (1 - g'(\delta))\left(\left(1 - g\left(\frac{(1+\gamma)\delta}{W'_{\min}}\right)\right)\mathrm{OPT}_{\mathsf{P}(X)} + kh\left(\frac{(1+\gamma)\delta}{W'_{\min}}\right)\right) - h'(\delta)$$

with runtime

$$\left\lceil\left(\frac{4(1+\gamma)k}{\epsilon W'_{\min}}\right)^{r_+}\right\rceil T(\delta).$$

Proof of Lemma 10. Recall that we have known non-negative factorization $W' = AB^\top$, where $A, B \in \mathbb{R}^{m \times r_+}_{\geq 0}$. First, we make the following observation on the scale of A and B:

Observation 4. *There exists $A', B' \in \mathbb{R}^{m \times r_+}_{\geq 0}$ where $W' = A'B'^\top$, such that $\|a'_i\|_1 \leq 1 + \gamma$ for every $i \in [m]$ and $\|b'_j\|_1 = 1$ for every $j \in [r_+]$.*

Proof of Observation 4. We construct A' and B' explicitly. Let the rows of B' be the rows of B that are rescaled so that $\|b_j\|_1 = 1$. That is, let $b'_{jk} = b_{jk} / \sum_{k'=1}^{m} b_{jk'}$ for every $j \in [r_+]$ and $k \in [m]$. Let the columns of A' be the columns of A that are rescaled accordingly. That is, let $a'_{ij} = a_{ij} \sum_{k'=1}^{m} b_{jk'}$ for every $i \in [m]$ and $j \in [r_+]$. Then we have $a'_{ij} b'_{jk} = a_{ij} b_{jk}$. Therefore $W' = A'B'^\top$. Finally, since $1 - \gamma \leq W_{ij}/W'_{ij} \leq 1 + \gamma$ and $\sum_{j=1}^{m} w_{ij} = 1$ for every $i \in [m]$, we have $\sum_{j=1}^{m} w'_{ij} \leq 1 + \gamma$ for every $i \in [m]$. Therefore for every $i \in [m]$, we have

$$1 + \gamma \geq \sum_{j=1}^{m} w'_{ij} = \sum_{j=1}^{r_+} a'_{ij} \sum_{k=1}^{m} b'_{jk} = \sum_{j=1}^{r_+} a'_{ij} = \|a'_i\|_1.$$

■

By Observation 4, we may assume $\|a_i\|_1 \leq 1 + \gamma$ for every $i \in [m]$ and $\|b_j\|_1 = 1$ for every $j \in [r_+]$ from now on. Let

$$Y = \{B^\top x \mid x \text{ is a feasible solution to } \mathsf{P}'(I)\} \subset \mathbb{R}^{r_+}. \tag{A.7}$$

We will partition the t-space $[W'_{\min}, 1 + \gamma]^m$ by partitioning Y. Let δ' be the quantity

$$\delta' = \frac{W'_{\min} \delta}{(1 + \gamma)\sqrt{k}}, \tag{A.8}$$

where the reason for this choice will be specified momentarily. Notice that $\|y\|_2 \leq \sqrt{k}$ for every $y \in Y$. As seen in the proof of Proposition 7, we can create a cover of a ball with radius $\sqrt{k}$ in $\mathbb{R}^{r_+}$ using $\lceil (4\sqrt{k}/\delta')^{r_+} \rceil$ number of balls with radius $\delta'/2$. Therefore we can create a partition $\mathcal{Y} = \{Y_1, \ldots, Y_\ell\}$ of Y such that $\ell = \lceil (4\sqrt{k}/\delta')^{r_+} \rceil$, and $\|y - y'\|_2 \leq \delta'$ for every $\ell' \in [\ell]$ and $y, y' \in Y_{\ell'}$.

Fix any row $a_i^\top$ of A. For every $\ell' \in [\ell]$ and $i, i' \in I_{\ell'}$, we have

$$\begin{aligned}\left|\frac{a_i^\top y}{a_i^\top y'} - 1\right| &= \left|\frac{a_i^\top (y - y')}{a_i^\top y}\right| \\ &\le \frac{\|a_i\|_2 \|y - y'\|_2}{|a_i^\top y|} \\ &\le \frac{\delta'(1+\gamma)\sqrt{k}}{W'_{\min}}.\end{aligned}$$

Thus, for our particular choice of δ', we have that $|a_i^\top y / a_i^\top y' - 1| \le \delta$ for every $\ell' \in [\ell]$ and $y, y' \in Y_{\ell'}$.

For every $\ell' \in [\ell]$, let $T_{\ell'} = \{Ay \mid y \in Y_{\ell'}\} \subset \mathbb{R}^m$. Then for every $\ell' \in [\ell]$ and $t, t' \in T_{\ell'}$, we have that $|(t_i/t'_i) - 1| \le \delta$ for every $i \in [m]$. Fix any $t_1 \in T_1, \ldots, t_\ell \in T_\ell$. The algorithm ALG will use the given oracle ALG$'$ to obtain a solution for each $\mathsf{P}(X, t_{\ell'})$, and then output the solution that has the highest objective value of $\mathsf{P}(X, t_{\ell'})$. That is,

$$\mathrm{ALG}_{\mathsf{P}(X)} = \max_{\ell' \in [\ell]} \mathrm{ALG}'_{\mathsf{P}(X, t_{\ell'})}.$$

Because $\ell = \lceil (4\sqrt{k}/\delta)^{r_+} \rceil = \lceil (4(1+\gamma)k/\epsilon W'_{\min})^{r_+} \rceil$, the runtime of ALG is

$$\lceil (4(1+\gamma)k/\epsilon W'_{\min})^{r_+} \rceil T(\delta).$$

Finally, we prove the performance guarantee for ALG. Let $t^* \in \arg\max_{t \in \mathbb{R}_+^m} \mathrm{OPT}_{\mathsf{P}(X,t)}$. Assume $t^* \in T_{\ell'}$. Then $|(t_i^*/(t_{\ell'})_i) - 1| \le \delta$. Let x^* be the corresponding optimal solution to $\mathsf{P}(X, t^*)$. Then by Lemma 9, x^* is an optimal solution to $\mathsf{P}(X)$. Let x' be an optimal solution of $\mathsf{P}(X, t_{\ell'})$. Because $t_{\ell'} \in [W'_{\min}, 1+\gamma]^m$, we have $\sum_{j=1}^{r_+} a_{ij} d_j^\top x^*/(t_{\ell'})_i = w_i'^\top x^*/(t_{\ell'})_i \le (1+\gamma)/W'_{\min}$. Also, since $f_i(x - \epsilon) \ge (1 - g(\epsilon)) f_i(x) - h(\epsilon)$ for all x, we have $f_i(x + \epsilon) \le (f_i(x) + h(\epsilon))/(1 - g(\epsilon))$ for

all x. As a consequence of Lemma 9, we have $\mathrm{OPT}_{\mathsf{P}(X)} = \mathrm{OPT}_{\mathsf{P}(X,t^*)}$. Then

$$\begin{aligned}
\mathrm{OPT}_{\mathsf{P}(X)} &= \mathrm{OPT}_{\mathsf{P}(X,t^*)} \\
&= \sum_{i=1}^{m} x_i^* f_i \left(\frac{\sum_{j=1}^{r_+} a_{ij} d_j^\top x^*}{t_i^*} \right) \\
&\le \sum_{i=1}^{m} x_i^* f_i \left((1+\delta) \frac{\sum_{j=1}^{r_+} a_{ij} d_j^\top x^*}{(t_{\ell'})_i} \right) \\
&\le \frac{\sum_{i=1}^{m} x_i^* f_i \left(\frac{\sum_{j=1}^{r_+} a_{ij} d_j^\top x^*}{(t_{\ell'})_i} \right) + kh(\delta(1+\gamma)/W'_{\min})}{1 - g(\delta(1+\gamma)/W'_{\min})} \\
&\le \frac{\mathrm{OPT}_{\mathsf{P}(X,t_{\ell'})} + kh(\delta(1+\gamma)/W'_{\min})}{1 - g(\delta(1+\gamma)/W'_{\min})}.
\end{aligned}$$

Rearranging the above, we have

$$\mathrm{OPT}_{\mathsf{P}(X,t_{\ell'})} \ge (1 - g(\delta(1+\gamma)/W'_{\min}))\mathrm{OPT}_{\mathsf{P}(X)} - kh(\delta(1+\gamma)/W'_{\min}).$$

Therefore,

$$\begin{aligned}
&\mathrm{ALG}_{\mathsf{P}(X)} \\
\ge\ &\mathrm{ALG}'_{\mathsf{P}(X,t_{\ell'})} \\
\ge\ &(1 - g'(\delta))\mathrm{OPT}_{\mathsf{P}(X,t_{\ell'})} - h'(\delta) \\
\ge\ &(1 - g'(\delta))\left(\left(1 - g\left(\frac{(1+\gamma)\delta}{W'_{\min}}\right)\right) \mathrm{OPT}_{\mathsf{P}(X)} - kh\left(\frac{(1+\gamma)\delta}{W'_{\min}}\right) \right) - h'(\delta).
\end{aligned}$$

■

Below we give the pseudo-code of the algorithm ALG in Lemma 10.

Algorithm 13: Phase Two: Discretization of Auxiliary Problems

Input: Instance of Problem $\mathsf{P}(X)$, oracle ALG′ from Lemma 10, parameter $\delta > 0$

Output: Solution x to $\mathsf{P}(X)$

`// Discretize the space of auxiliary variable`

Set $\delta' > 0$ according to Eq. (A.8)

Construct maximal $(\delta'/2)$-separated set $C \subset B(0, \sqrt{k})$ with $|C| = \lceil (4\sqrt{k}/\delta')^{r_+} \rceil$

Let $\mathcal{Y} = \{Y_1, \ldots, Y_\ell\} \leftarrow \{B(c, \delta'/2) : c \in C\}$

`// Solve each discretized auxiliary problem`

Set $\delta_1 \leftarrow \delta$ and $\delta_2 \leftarrow k\delta$

for $\ell' = 1$ **to** ℓ **do**

  $T_{\ell'} \leftarrow \{Ay : y \in Y_{\ell'}\}$

  Choose arbitrary $t_{\ell'} \in T_{\ell'}$

  Let x be the output of ALG′ with inputs $\mathsf{P}(X, t_{\ell'})$ and parameters δ_1, δ_2

  Record $(x_{\ell'}, f_{\mathsf{P}(X,t_{\ell'})}(x_{\ell'}))$

end

return $\arg\max_{x_{\ell'}} f_{\mathsf{P}(X,t_{\ell'})}(x_{\ell'})$ *among recorded solutions*

Step 5: Complete Linearization of Auxiliary Problems

Lemma 10 shows that, to approximately solve $\mathsf{P}(X)$, it suffices to construct an oracle that approximately solves $\mathsf{P}(X, t)$ for any given $t \in [W'_{\min}, 1 + \gamma]^m$. Below we give such an oracle. Let $c_i = a_i/t_i \in \mathbb{R}^{r_+}_{\geq 0}$ for each $i \in [m]$. Then $\mathsf{P}(X, t)$ can be equivalently formulated as

$$
\begin{aligned}
(\mathsf{P}(X,t)) \qquad \max \quad & f_{\mathsf{P}(X,t)}(x) = \sum_{i=1}^{m} x_i f_i \left(\sum_{j=1}^{r_+} c_{ij} d_j^\top x \right) \\
\text{s.t.} \quad & x \in \{0,1\}^m, \\
& e^\top x \leq k, \\
& w_i'^\top x \leq t_i \quad \forall i \in [m], \\
& x_i = 1 \quad \forall i \in \cup_j X_j, \\
& x_i = 0 \quad \forall i \in \cup_j \hat{X}_j.
\end{aligned}
$$

To solve $\mathsf{P}(X, t)$, we partition the space of possible values $(d_1^\top x, \ldots, d_{r_+}^\top x) \in \mathbb{R}^{r_+}$, as well as the space of the objective value $f_{\mathsf{P}(X,t)}(x)$.

Lemma 11. *Fix any $t \in [W'_{\min}, 1+\gamma]^m$. Suppose we are given an oracle with runtime $T(\delta_1, \delta_2)$ that takes $\mathsf{P}(X,t)$, any $\theta = (\theta_1, \ldots, \theta_{r_+}) \in \mathbb{R}^{r_+}$, any $\zeta \geq 0$, and any $\delta_1, \delta_2 > 0$ as inputs, and either*

1. *Scenario one: correctly declares that there is no feasible x to $\mathsf{P}(X,t)$ such that $d_j^\top x \geq \theta_j$ for every $j \in [r_+]$ and $\sum_{i=1}^m x_i f_i(c_i^\top \theta) \geq \zeta$, or*
2. *Scenario two: outputs a feasible x to $\mathsf{P}(X,t)$ such that $d_j^\top x + \delta_1 \geq \theta_j$ for every $j \in [r_+]$ and $\sum_{i=1}^m x_i f_i(c_i^\top \theta) + \delta_2 \geq \zeta$.*

Then there exists an algorithm ALG *that satisfies*

$$\mathrm{ALG}_{\mathsf{P}(X,t)} \geq \left(1 - g\left(\frac{\delta_1(1+\gamma)}{W'_{\min}}\right)\right)(\mathrm{OPT}_{\mathsf{P}(X,t)} - 2\delta_2) - kh\left(\frac{\delta_1(1+\gamma)}{W'_{\min}}\right)$$

with runtime

$$\lceil((Vu)_{\max} - \min\{0, (Vu)_{\min}\})/\delta_1\rceil^{r_+} \cdot \left\lceil k \max_{i\in[m]}\{f_i((Vu)_{\max})\}/\delta_2 \right\rceil T(\delta_1, \delta_2).$$

Proof of Lemma 11. Let $(Vu)_{\min}$ be the minimum entry of Vu (possibly negative). Because $\|b_j\|_1 = 1$ and $d_j = b_j \odot (Vu)$ for every $j \in [r_+]$, we have $d_j^\top x^* \in [\min\{0, (Vu)_{\min}\}, (Vu)_{\max}]$ for every $j \in [r_+]$. Also, because each f_i is non-decreasing, $\mathrm{OPT}_{\mathsf{P}(X,t)} \in [0, k \max_{i\in[m]}\{f_i((Vu)_{\max})\}]$. Similar to the proof of Lemma 10, we will create a partition of the space of possible values $(d_1^\top x, \ldots, d_{r_+}^\top x) \in \mathbb{R}^{r_+}$, as well as the space of the objective value $f_{\mathsf{P}(X,t)}(x)$. We then show that it is sufficient to solve $\mathsf{P}(X,t)$ in each subset of the partition. Let $\Delta_\ell = \min\{0, (Vu)_{\min}\} + \delta_1(\ell - 1)$ for $\ell = 1, \ldots, \lceil((Vu)_{\max} - \min\{0, (Vu)_{\min}\})/\delta_1\rceil$. Let $\Delta'_s = \delta_2(s-1)$ for $s = 1, \ldots, \lceil k \max_{i\in[m]}\{f_i((Vu)_{\max})\}/\delta_2\rceil$. Consider all tuples $(\ell_1, \ldots, \ell_{r_+}, s)$. There are in total

$$\lceil((Vu)_{\max} - \min\{0, (Vu)_{\min}\})/\delta_1\rceil^{r_+} \cdot \left\lceil k \max_{i\in[m]}\{f_i((Vu)_{\max})\}/\delta_2 \right\rceil$$

such tuples. Moreover, there exists a tuple $(\ell_1^*, \ldots, \ell_{r_+}^*, s^*)$ such that $\Delta_{\ell_i^*} \leq x_i^* \leq \Delta_{\ell_i^*+1}$ for each $i \in [m]$ and $\Delta'_{s^*} \leq \mathrm{OPT}_{\mathsf{P}(X,t)} \leq \Delta'_{s^*+1}$.

For each tuple $(\ell_1, \ldots, \ell_{r_+}, s)$, our desired algorithm uses the given oracle to determine whether there exists a feasible x to $\mathsf{P}(X,t)$ that satisfies the conditions in scenario two with $\theta_i = \Delta_{\ell_i}$ for each $i \in [r_+]$ and $\zeta = \Delta'_s$. Then our desired algorithm returns the x with the highest objective value of $\mathsf{P}(X,t)$ among all tuples. Note that x^* satisfies the conditions in scenario two on the tuple $(\ell_1^*, \ldots, \ell_{r_+}^*, s^*)$. Therefore the given oracle would return some feasible x' to $\mathsf{P}(X,t)$ that satisfies the conditions in scenario two

with $\theta_i = \Delta_{\ell_i^*}$ for each $i \in [r_+]$ and $\zeta = \Delta'_{s^*}$. Notice that $c_{ij} = a_{ij}/t_i \le (1+\gamma)/W'_{\min}$. Then by the conditions in scenario two we have

$$\begin{aligned}
\mathrm{ALG}_{\mathsf{P}(X,t)} &\ge \sum_{i=1}^{m} x'_i f_i\left(\sum_{j=1}^{r_+} c_{ij} d_j^\top x'\right) \\
&\ge \sum_{i=1}^{m} x'_i f_i\left(\sum_{j=1}^{r_+} c_{ij}(\theta_j - \delta_1)\right) \\
&\ge \sum_{i=1}^{m} x'_i f_i\left(\sum_{j=1}^{r_+} c_{ij}\theta_j - \delta_1(1+\gamma)/W'_{\min}\right) \\
&\ge (1 - g(\delta_1(1+\gamma)/W'_{\min})) \sum_{i=1}^{m} x'_i f_i\left(\sum_{j=1}^{r_+} c_{ij}\theta_j\right) - kh(\delta_1(1+\gamma)/W'_{\min}) \\
&\ge (1 - g(\delta_1(1+\gamma)/W'_{\min}))(\Delta'_{s^*} - \delta_2) - kh(\delta_1(1+\gamma)/W'_{\min}) \\
&\ge (1 - g(\delta_1(1+\gamma)/W'_{\min}))(\mathrm{OPT}_{\mathsf{P}(X,t)} - 2\delta_2) - kh(\delta_1(1+\gamma)/W'_{\min}),
\end{aligned}$$

where the second and the fifth inequalities follow from the conditions in scenario two, and the last inequality follows since $\Delta'_{s^*} \le \mathrm{OPT}_{\mathsf{P}(X,t)} \le \Delta'_{s^*+1}$.

Because there are in total

$$\lceil ((Vu)_{\max} - \min\{0, (Vu)_{\min}\})/\delta_1 \rceil^{r_+} \cdot \left\lceil k \max_{i\in[m]}\{f_i((Vu)_{\max})\}/\delta_2 \right\rceil$$

number of tuples $(\ell_1, \ldots, \ell_{r_+}, s)$, the runtime of our algorithm is

$$\lceil ((Vu)_{\max} - \min\{0, (Vu)_{\min}\})/\delta_1 \rceil^{r_+} \cdot \left\lceil k \max_{i\in[m]}\{f_i((Vu)_{\max})\}/\delta_2 \right\rceil T(\delta_1, \delta_2).$$

■

Below we give the pseudo-code of the algorithm ALG in Lemma 11.

Algorithm 14: Phase Two: Complete Linearization of Auxiliary Problems

Input: Instance of Problem $\mathsf{P}(X, t)$, oracle from Lemma 11, parameters $\delta_1, \delta_2 > 0$

Output: Solution x to $\mathsf{P}(X, t)$

```
// Discretize value and objective spaces
```

Let $\nu_{\min} \leftarrow \min\{0, (Vu)_{\min}\}$ and $\nu_{\max} \leftarrow (Vu)_{\max}$

Let $L \leftarrow \lceil (\nu_{\max} - \mu_{\min})/\delta_1 \rceil$ and $S \leftarrow \lceil k \cdot \max_{i \in [m]} f_i(\nu_{\max})/\delta_2 \rceil$

for $\ell = 1$ **to** L **do**

  $\Delta_\ell \leftarrow \nu_{\min} + \delta_1(\ell - 1)$

end

for $s = 1$ **to** S **do**

  $\Delta'_s \leftarrow \delta_2(s - 1)$

end

```
// Enumerate and solve linearized problems
```

for *each tuple* $(\ell_1, \ldots, \ell_{r_+}, s)$ *where* $\ell_i \in [L]$ *and* $s \in [S]$ **do**

  Set $\theta_i \leftarrow \Delta_{\ell_i}$ for each $i \in [r_+]$ and $\zeta \leftarrow \Delta'_s$

  Run oracle from Lemma 11 with inputs $\mathsf{P}(X, t)$, value tuple $(\theta_1, \ldots, \theta_{r_+}, \zeta)$, and parameters δ_1, δ_2

  if *oracle returns scenario two with feasible solution* x **then**

    Record $(x, f_{\mathsf{P}(X,t)}(x))$

  end

end

return $\arg\max_x f_{\mathsf{P}(X,t)}(x)$ *among recorded solutions*

Step 6: Approximation of Linearized Auxiliary Problems via LP Rounding

Lemma 11 shows that, to solve $P(X, t)$ for any given $t \in [W'_{\min}, 1 + \gamma]^m$, it is enough to give an oracle described in Lemma 11. Similar to the idea of enumerating partial solutions by constructing valid tuples based on the values of each d_j, we further construct index sets based on the values of $f_i(c_i^\top \theta)$ and enumerate all possible index sets. Recall that via the valid tuple $(X_1, \ldots, X_{r_+})$, we have already fixed at least λ indices of any feasible solution to $\mathsf{P}(X, t)$ to be equal to 1, namely the indices in $\cup_j X_j$. Fix

$$\lambda' = \lceil (2r_+ + 2) \max_{i \in [m]} \{f_i((Vu)_{\max})\}/\epsilon \rceil. \tag{A.9}$$

Let $X' \subset [m] \setminus (\cup_j X_j) \cup (\cup_j \hat{X}_j)$ be an index set such that $0 \le |X'| \le \lambda'$. Let

$$\hat{X}' = \{i \in [m] \setminus X' \cup (\cup_j X_j) \cup (\cup_j \hat{X}_j) \mid f_i(c_i^\top \theta) > \min_{i' \in X'} \{f_{i'}(c_{i'}^\top \theta)\}\}. \tag{A.10}$$

In words, $\hat{X}'$ consists of the indices outside of $X' \cup (\cup_j X_j) \cup (\cup_j \hat{X}_j)$ whose corresponding values of $f_i(c_{i'}^\top \theta)$ are strictly greater than the minimum value of $f_i(c_i^\top \theta)$ across indices in X'. Then every feasible solution z to $\mathsf{P}(X, t)$ **corresponds** to an index set X' in the following way: let $Z = \{i \in [m] \setminus (\cup_j X_j) \cup (\cup_j \hat{X}_j) \mid z_i = 1\}$. We define $X' \subset Z$ to be the set of indices $i \in Z$ such that $f_i(c_i^\top \theta)$ is among the $\min\{\lambda', |Z|\}$ highest values in Z. That is, let $\pi' : [|Z|] \to Z$ be a sorting of Z according to $f_i(c_i^\top \theta)$ such that $f_{\pi'(1)}(c_{\pi'(1)}^\top \theta) \ge \cdots \ge f_{\pi'(|Z|)}(c_{\pi'(|Z|)}^\top \theta)$. Then $X' = \{\pi'(1), \ldots, \pi'(\min\{\lambda', |Z|\})\}$. Similar to before, if z corresponds to X', we must have $z_i = 1$ for $i \in X'$ and $z_i = 0$ for $i \in \hat{X}'$.

As in Lemma 8, the notion of correspondence to index sets forms a partition of the set of feasible solutions to $\mathsf{P}(X, t)$. Therefore, in order to give an oracle described in Lemma 11, it suffices to give an oracle described in Lemma 11 separately for each subset of this partition. This is formally stated in the following result:

Observation 5. *Suppose we are given an oracle with runtime $T(\delta_1, \delta_2)$ that takes $\mathsf{P}(X, t)$, any $\theta = (\theta_1, \ldots, \theta_{r_+}) \in \mathbb{R}^{r_+}$, any $\zeta \ge 0$, any $\delta_1, \delta_2 > 0$, and any index set $X' \subset [m] \setminus (\cup_j X_j) \cup (\cup_j \hat{X}_j)$ such that $0 \le |X'| \le \lambda'$ as inputs, and either*

1. *Scenario one: correctly declares that there is no feasible x to $\mathsf{P}(X, t)$ that corresponds to X' such that $d_j^\top x \ge \theta_j$ for every $j \in [r_+]$ and $\sum_{i=1}^m x_i f_i(c_i^\top \theta) \ge \zeta$, or*
2. *Scenario two: outputs a feasible x to $\mathsf{P}(X, t)$ that corresponds to X' such that $d_j^\top x + \delta_1 \ge \theta_j$ for every $j \in [r_+]$ and $\sum_{i=1}^m x_i f_i(c_i^\top \theta) + \delta_2 \ge \zeta$.*

Then there exists an oracle described in Lemma 11 with runtime $\lambda' m^{\lambda'} T(\delta_1, \delta_2)$.

Proof of Observation 5. Because every feasible solution x to $\mathsf{P}(X, t)$ corresponds to exactly one index set $X' \subset [m] \setminus (\cup_j X_j) \cup (\cup_j \hat{X}_j)$ such that $0 \le |X'| \le \lambda'$, we can give an oracle described in Lemma 11 by enumerating all such index sets X' and applying the oracle in Observation 5.

More specifically, for the oracle described in Lemma 11 with inputs $\theta, \zeta, \delta_1, \delta_2$:

- If the oracle in Observation 5 outputs an x in scenario two with inputs $\theta, \zeta, \delta_1, \delta_2, X'$ for some X', then the oracle described in Lemma 11 also outputs this x in scenario two.

- If the oracle in Observation 5 declares scenario one with inputs $\theta, \zeta, \delta_1, \delta_2, X'$ for all X', then the oracle described in Lemma 11 also declares scenario one.

We can enumerate all index sets $X' \subset [m] \setminus (\cup_j X_j) \cup (\cup_j \hat{X}_j)$ such that $0 \leq |X'| \leq \lambda'$ by simply enumerating all combinations of $|X'|$ indices from $[m] \setminus (\cup_j X_j) \cup (\cup_j \hat{X}_j)$. Because there are at most $\sum_{k=0}^{\lambda'} \binom{m}{\lambda'} \leq \lambda' m^{\lambda'}$ such index sets X', the runtime of the oracle described in Lemma 11 is $\lambda' m^{\lambda'} T(\delta_1, \delta_2)$. ■

By Observation 5, it suffices to give an oracle as described. Below we give such an oracle.

First, suppose $|X'| < \lambda'$. Assume there exists a feasible solution z to $\mathsf{P}(X, t)$ that corresponds to X'. Let $Z = \{i \in [m] \setminus (\cup_j X_j) \cup (\cup_j \hat{X}_j) \mid z_i = 1\}$. Then because $|X'| = \min\{\lambda', |Z|\} = |Z|$ and $X' \subset Z$, we have $X' = Z$. Therefore, there is a unique feasible solution z of $\mathsf{P}(X, t)$ that corresponds to X', namely $z_i = 1$ for every $i \in X' \cup (\cup_j X_j)$ and $z_i = 0$ for every $i \notin X' \cup (\cup_j X_j)$. Therefore, if $|X'| < \lambda'$, the oracle in Observation 5 can directly check this unique feasible solution of $\mathsf{P}(X, t)$ that corresponds to X', and outputs the correct scenario accordingly.

From now on, we assume $|X'| = \lambda'$. Fix any $\theta \in \mathbb{R}^{r_+}$, any $\zeta \in \mathbb{R}_+$, any $\delta_1, \delta_2 > 0$, and any X'. The oracle essentially needs to determine the existence of a feasible binary solution to a system of linear inequalities, which is NP-hard in general. However, the linear constraints of $\mathsf{P}(X, t)$ lie in a lower-dimensional subspace, a structure we can exploit by solving a relaxation of the system, obtained by replacing binary variables with continuous ones, and rounding its solution back to a binary solution. Because of the rounding, it is possible that the values $(d_1^\top x, \ldots, d_{r_+}^\top x)$ and $\sum_{i=1}^m x_i f_i(c_i^\top \theta)$ of the rounded solution are out of the desired ranges. However, the valid tuple X and the index set X' we fixed before ensures that the gaps between the values and the desired ranges are within small constants.

We define a polyhedron $PH \subset \mathbb{R}^m$ as follows:

$$
(PH) \qquad
\begin{aligned}
\sum_{j=1}^{r_+} a_{ij}\, b_j^\top x &\le t_i && \text{for } i = 1, \dots, m,\\
e^\top x &\le k,\\
d_j^\top x &\ge \theta_j && \text{for } j = 1, \dots, r_+,\\
\sum_{i=1}^{m} x_i\, f_i(c_i^\top \theta) &\ge \zeta,\\
x_i &= 1 && \text{for } i \in \big(\bigcup_j X_j\big) \cup X',\\
x_i &= 0 && \text{for } i \in \big(\bigcup_j \hat{X}_j\big) \cup \hat{X}',\\
x_i &\in [0,1] && \text{for } i \notin \big(\bigcup_j X_j\big) \cup \big(\bigcup_j \hat{X}_j\big) \cup X' \cup \hat{X}'.
\end{aligned}
$$

Then PH is a polyhedron in $\mathbb{R}^m$ defined by at most $3m + r_+ + 2$ inequalities. Let $\mathrm{LP}(m, n)$ be the runtime of solving a linear program with m variables and n constraints. Then checking whether PH is non-empty and return a point in PH if PH is non-empty can be done in runtime $\mathrm{LP}(m, 3m + r_+ + 2)$. For more on the runtime of solving a linear program, we refer the readers to e.g. [74, 99] (ellipsoid methods) and [139, 164] (interior point methods). In what follows we assume that $PH \neq \emptyset$, otherwise the oracle outputs scenario one.

Lemma 12. *If $PH \neq \emptyset$, then we can find a point $z \in PH$ with at most $2r_+ + 2$ fractional components in runtime* $\mathrm{LP}(m, 3m + r_+ + 2) + \mathrm{LP}(m, 2m + 2r_+ + 2)$.

Proof of Lemma 12. Let $z^* \in PH$ be an (arbitrary) point found in runtime $\mathrm{LP}(m, 3m + r_+ + 2)$. Let $PH(z^*) \subset \mathbb{R}^{m - |(\cup_j X_j) \cup (\cup_j \hat{X}_j) \cup X' \cup \hat{X}'|}$ be the polyhedron on variable y, where the index set of y is taken to be $I_y = [m] \setminus (\cup_j X_j) \cup (\cup_j \hat{X}_j) \cup X' \cup \hat{X}'$, with the following constraints:

$$
(PH(z^*)) \qquad
\begin{aligned}
\sum_{i \in I_y} b_{ji}\, y_i &\le \sum_{i \in I_y} b_{ji}\, z_i^* && \text{for } j = 1, \dots, r_+,\\
\sum_{i \in I_y} y_i &\le \sum_{i \in I_y} z_i^*,\\
\sum_{i \in I_y} d_{ji}\, y_i &\ge \theta_j - \sum_{i \in (\bigcup_j X_j) \cup X'} d_{ji} && \text{for } j = 1, \dots, r_+,\\
\sum_{i \in I_y} y_i\, f_i(c_i^\top \theta) &\ge \zeta - \sum_{i \in (\bigcup_j X_j) \cup X'} f_i(c_i^\top \theta),\\
y_i &\in [0,1] && \text{for } i \in I_y.
\end{aligned}
$$

Note that $PH(z^*) \neq \emptyset$ since the projection of z^* on $\mathbb{R}^{|I_y|}$ is in $PH(z^*)$. Because $PH(z^*)$ has $2r_+ + 2$ linear inequalities other than the inequalities $y_i \in [0,1]$ for $i \in I_y$, we can compute a vertex y^* of $PH(z^*)$ with at most $2r_+ + 2$ fractional components with runtime $\mathrm{LP}(m, 2m + 2r_+ + 2)$ (see a standard textbook on linear programming, e.g., [155]).

Let $z \in [0,1]^m$ where

$$z_i = \begin{cases} 1 & \text{if } i \in (\cup_j X_j) \cup X' \\ 0 & \text{if } i \in (\cup_j \hat{X}_j) \cup \hat{X}' \\ y_i^* & \text{if } i \in I_y \end{cases}.$$

Then z has at most $2r_+ + 2$ fractional components. We show that $z \in PH$. Because $z_i = z_i^* = 1$ for $i \in (\cup_j X_j) \cup X'$ and $z_i = z_i^* = 0$ for $i \in (\cup_j \hat{X}_j) \cup \hat{X}'$, the last three sets of constraints of PH is satisfied. By the first set of constraints of $PH(z^*)$ we have $\sum_{i \in I_y} b_{ji} z_i \leq \sum_{i \in I_y} b_{ji} z_i^*$. Because $a_{ij} \geq 0$ for every i, j, the first set of constraints of PH is satisfied. Similarly the second constraint of PH is also satisfied. By the third set of constraints of $PH(z^*)$ we have

$$d_j^\top z = \sum_{i \in (\cup_j X_j) \cup X'} d_{ji} + \sum_{i \in I_y} d_{ji} z_i \geq \sum_{i \in (\cup_j X_j) \cup X'} d_{ji} + \left(\theta_j - \sum_{i \in (\cup_j X_j) \cup X'} d_{ji} \right) = \theta_j,$$

so the third set of constraints of PH is satisfied. Similarly the fourth constraint of PH is also satisfied. Therefore $z \in PH$ is the desired point. ■

Let z be the point obtained in Lemma 12. Then z satisfies all the constraints of $\mathsf{P}(X,t)$ except the integrality constraints. We round z down to obtain a feasible solution: let $\bar{z} \in \{0,1\}^m$ where $\bar{z}_i = \lfloor z_i \rfloor$ for each i. Notice that since $w'_{ij} > 0$ for every i, j, we have $e^\top \bar{z} \leq e^\top z \leq k$ and ${w'_i}^\top \bar{z} \leq {w'_i}^\top z \leq t_i$ for every $i \in [m]$. Therefore $\bar{z}$ is feasible to $\mathsf{P}(X,t)$. Moreover, since $\bar{z} = 1$ for $i \in X'$ and $\bar{z} = 0$ for $i \in \hat{X}'$, we have that $\bar{z}$ corresponds to X'.

In the final step, we show that by setting λ, λ' appropriately, $\bar{z}$ satisfies the conditions in scenario two of Observation 5, hence completing the oracle in Observation 5.

Lemma 13. *Set*

$$\lambda = \lceil (2r_+ + 2)(Vu)_{\max} / \delta_1 \rceil$$

and

$$\lambda' = \lceil (2r_+ + 2) k \max_{i \in [m]} \{ f_i((Vu)_{\max}) \} / \delta_2 \rceil.$$

Then $d_j^\top \bar{z} + \delta_1 \geq \theta_j$ *for every* $j \in [r_+]$, *and* $\sum_{i=1}^m \bar{z}_i f_i(c_i^\top \theta) + \delta_2 \geq \zeta$.

Proof of Lemma 13. Because $z \in PH$, we have $d_j^\top z \geq \theta_j$ for every $j \in [r_+]$ and $\sum_{i=1}^m z_i f_i(c_i^\top \theta) \geq \zeta$. Fix $j \in [r_+]$ and let $\ell \in X_j$ be an index where $d_{j\ell} = \min_{\ell' \in X_j}\{d_{j\ell'}\}$. Then since $|X_j| = \lambda$, we have

$$d_j^\top z \geq \sum_{\ell' \in X_j} d_{j\ell'} z_{\ell'} \geq \lambda d_{j\ell}.$$

On the other hand, $\bar{z}$ is obtained by rounding z down. Notice that $z_{\ell'} \in \{0, 1\}$ for all $\ell' \in X_j \cup \hat{X}_j$, that is, for all ℓ' such that $d_{j\ell'} > d_{j\ell}$. Therefore for all ℓ' such that $d_{j\ell'} > d_{j\ell}$, we have $z_{\ell'} = \bar{z}_{\ell'}$. By Lemma 12, z has at most $2r_+ + 2$ fractional components. Therefore, because $\theta_j \leq d_j^\top z \leq (Vu)_{\max}$, we have

$$\begin{aligned} d_j^\top \bar{z} &\geq d_j^\top z - (2r_+ + 2) d_{j\ell} \\ &\geq d_j^\top z - (2r_+ + 2) d_j^\top z/\lambda \\ &\geq \theta_j - (2r_+ + 2)(Vu)_{\max}/\lambda \\ &\geq \theta_j - \delta_1. \end{aligned}$$

Similarly, let $p \in X'$ be an index where $f_p(c_p^\top \theta) = \min_{p' \in X'}\{f_{p'}(c_{p'}^\top \theta)\}$. Then since $|X'| = \lambda'$, we have

$$\sum_{i=1}^m z_i f_i(c_i^\top \theta) \geq \sum_{p' \in X'} z_{p'} f_{p'}(c_{p'}^\top \theta) \geq \lambda' f_p(c_p^\top \theta).$$

On the other hand, $\bar{z}$ is obtained by rounding z down. Notice that $z_{p'} \in \{0, 1\}$ for all $p' \in X' \cup \hat{X}'$, that is, for all p' such that $f_{p'}(c_{p'}^\top \theta) > f_p(c_p^\top \theta)$. Therefore for all p' such that $f_{p'}(c_{p'}^\top \theta) > f_p(c_p^\top \theta)$, we have $z_{\ell'} = \bar{z}_{\ell'}$. By Lemma 12 z has at most $2r_+ + 2$ fractional components. Therefore, because $\zeta \leq \sum_{i=1}^m z_i f_i(c_i^\top \theta) \leq k \max_{i \in [m]}\{f_i((Vu)_{\max})\}$,

$$\begin{aligned} \sum_{i=1}^m \bar{z}_i f_i(c_i^\top \theta) &\geq \sum_{i=1}^m z_i f_i(c_i^\top \theta) - (2r_+ + 2) f_p(c_p^\top \theta) \\ &\geq \sum_{i=1}^m z_i f_i(c_i^\top \theta) - (2r_+ + 2) \sum_{i=1}^m z_i f_i(c_i^\top \theta)/\lambda' \\ &\geq \zeta - (2r_+ + 2) k \max_{i \in [m]}\{f_i((Vu)_{\max})\}/\lambda' \\ &\geq \zeta - \delta_2. \end{aligned}$$

■

Below we give the pseudo-code of the oracle described in Lemma 11.

Algorithm 15: Phase Two: Approximation of Linearized Auxiliary Problems

Input: Instance of Problem $\mathsf{P}(X,t)$, thresholds $\theta \in \mathbb{R}^{r_+}$, $\zeta \in \mathbb{R}_+$, parameters $\delta_1, \delta_2 > 0$

Output: One of two scenarios:

1. **Scenario 1:** Certify no feasible x to $\mathsf{P}(X,t)$ satisfies $d_j^\top x \geq \theta_j \ \forall j \in [r_+]$ and $\sum_{i=1}^m x_i f_i(c_i^\top \theta) \geq \zeta$
2. **Scenario 2:** Return feasible x to $\mathsf{P}(X,t)$ with $d_j^\top x + \delta_1 \geq \theta_j \ \forall j \in [r_+]$ and $\sum_{i=1}^m x_i f_i(c_i^\top \theta) + \delta_2 \geq \zeta$

Set $\lambda' > 0$ according to Eq. (A.9)

Let $\mathcal{M} \leftarrow [m] \setminus (\cup_j X_j) \cup (\cup_j \hat{X}_j)$

`// Check small solutions`

for *each* $X' \subset \mathcal{M}$ *with* $0 \leq |X'| \leq \lambda' - 1$ **do**

Set $\hat{X}'$ according to Eq. (A.10)

Define $z \in \{0,1\}^m$: $z_i = 1$ if $i \in X' \cup (\cup_j X_j)$, else $z_i = 0$

if *z feasible and satisfies thresholds with slack* (δ_1, δ_2) **then**

return *Scenario 2 with solution z*

end

end

`// Solve via LP rounding for larger solutions`

for *each* $X' \subset \mathcal{M}$ *with* $|X'| = \lambda'$ **do**

Construct polyhedron PH according to Eq. (PH)

if $PH \neq \emptyset$ **then**

Choose arbitrary $z^* \in PH$

Construct $PH(z^*)$ according to Eq. $(PH(z^*))$

Compute vertex y^* of $PH(z^*)$ with $\leq 2r_+ + 2$ fractional components

Set $I_y \leftarrow \mathcal{M} \setminus (X' \cup \hat{X}')$

Define $z \in [0,1]^m$: $z_i = \begin{cases} 1 & \text{if } i \in (\cup_j X_j) \cup X' \\ 0 & \text{if } i \in (\cup_j \hat{X}_j) \cup \hat{X}' \\ y_i^* & \text{if } i \in I_y \end{cases}$

return *Scenario 2 with solution z*

end

end

return *Scenario 1*

Completing the Proof

Finally, we analyze our algorithm's overall performance and runtime. Let $\delta_1 = \epsilon$ and $\delta_2 = k\epsilon$, and in order to apply Lemma 13, we set $\lambda = (2r_+ + 2)(Vu)_{\max}/\epsilon$ and $\lambda' = (2r_+ + 2)\max_{i\in[m]}\{f_i((Vu)_{\max})\}/\epsilon$. We will treat the performance guarantee and runtime analysis separately.

Performance Guarantee: Let

$$c_{\epsilon,\gamma,W'_{\min}} = \frac{(1+\gamma)\epsilon}{W'_{\min}}.$$

The algorithm ALG (for solving $\mathsf{P}(X,t)$) in Lemma 11 satisfies

$$\begin{aligned}\mathrm{ALG}_{\mathsf{P}(X,t)} &\geq \left(1 - g\left(\frac{\delta_1(1+\gamma)}{W'_{\min}}\right)\right)(\mathrm{OPT}_{\mathsf{P}(X,t)} - 2\delta_2) - kh\left(\frac{\delta_1(1+\gamma)}{W'_{\min}}\right)\\ &= (1 - g(c_{\epsilon,\gamma,W'_{\min}}))\mathrm{OPT}_{\mathsf{P}(X,t)} - k(2\epsilon(1 - g(c_{\epsilon,\gamma,W'_{\min}})) + h(c_{\epsilon,\gamma,W'_{\min}})).\end{aligned}$$

This gives the ALG′ (for solving $\mathsf{P}(X,t)$) in Lemma 10 with

$$g'(\epsilon) = g(c_{\epsilon,\gamma,W'_{\min}})$$

and

$$h'(\epsilon) = k(2\epsilon(1 - g(c_{\epsilon,\gamma,W'_{\min}})) + h(c_{\epsilon,\gamma,W'_{\min}})).$$

Therefore, the algorithm ALG (for solving $\mathsf{P}(X)$) in Lemma 10 satisfies

$$\begin{aligned}\mathrm{ALG}_{\mathsf{P}(X)} &\geq (1 - g'(\epsilon))((1 - g(c_{\epsilon,\gamma,W'_{\min}}))\mathrm{OPT}_{\mathsf{P}(X)} + kh(c_{\epsilon,\gamma,W'_{\min}})) - h'(\epsilon)\\ &= (1 - g(c_{\epsilon,\gamma,W'_{\min}}))((1 - g(c_{\epsilon,\gamma,W'_{\min}}))\mathrm{OPT}_{\mathsf{P}(X)} + kh(c_{\epsilon,\gamma,W'_{\min}}))\\ &\quad - k(2\epsilon(1 - g(c_{\epsilon,\gamma,W'_{\min}})) + h(c_{\epsilon,\gamma,W'_{\min}}))\\ &= (1 - g(c_{\epsilon,\gamma,W'_{\min}}))^2\mathrm{OPT}_{\mathsf{P}(X)} - k(2\epsilon(1 - g(c_{\epsilon,\gamma,W'_{\min}}))\\ &\quad + g(c_{\epsilon,\gamma,W'_{\min}})h(c_{\epsilon,\gamma,W'_{\min}})).\end{aligned}$$

Therefore, the algorithm ALG (for solving $\mathsf{P}'(I)$) in Lemma 8 satisfies

$$\begin{aligned}\mathrm{ALG}_{\mathsf{P}'(I)} \geq\ & (1 - g(c_{\epsilon,\gamma,W'_{\min}}))^2\,\mathrm{OPT}_{\mathsf{P}'(I)}\\ & - k\Big(2\epsilon(1 - g(c_{\epsilon,\gamma,W'_{\min}})) + g(c_{\epsilon,\gamma,W'_{\min}})h(c_{\epsilon,\gamma,W'_{\min}})\Big).\end{aligned}$$

Finally, we apply Lemma 7 by plugging in $1 - \alpha = (1 - g(c_{\epsilon,\gamma,W'_{\min}}))^2$ and $\beta = -k(2\epsilon(1 - g(c_{\epsilon,\gamma,W'_{\min}})) + g(c_{\epsilon,\gamma,W'_{\min}})h(c_{\epsilon,\gamma,W'_{\min}}))$. This gives

$$\begin{aligned}\mathrm{ALG}_{\mathsf{P}(I)} \geq\ & (1 - g(2\gamma(Vu)_{\max}))^2(1 - g(c_{\epsilon,\gamma,W'_{\min}}))^2\mathrm{OPT}_{\mathsf{P}(I)}\\ & - k(1 - g(2\gamma(Vu)_{\max}))(2\epsilon(1 - g(c_{\epsilon,\gamma,W'_{\min}})) + g(c_{\epsilon,\gamma,W'_{\min}})h(c_{\epsilon,\gamma,W'_{\min}})\\ & + (1 - g(c_{\epsilon,\gamma,W'_{\min}}))^2 h(2\gamma(Vu)_{\max}) + h(2\gamma(Vu)_{\max})).\end{aligned}$$

Runtime Analysis: By Lemma 12, we give an oracle described in Observation 5 with runtime

$$T_{\mathrm{LP}} := \mathrm{LP}(m, 3m + r_+ + 2) + \mathrm{LP}(m, 2m + 2r_+ + 2).$$

Therefore, by Observation 5, we give an oracle described in Lemma 11 with runtime

$$\lambda' m^{\lambda'} T_{\mathrm{LP}}.$$

Therefore, the algorithm ALG′ (for solving $\mathsf{P}(X, t)$) in Lemma 10 has runtime

$$\begin{aligned}
&\lceil ((Vu)_{\max} - \min\{0, (Vu)_{\min}\})/\delta_1 \rceil^{r_+} \cdot \left\lceil k \max_{i\in[m]}\{f_i((Vu)_{\max})\}/\delta_2 \right\rceil \cdot \lambda' m^{\lambda'} T_{\mathrm{LP}} \\
= &\lceil ((Vu)_{\max} - \min\{0, (Vu)_{\min}\})/\epsilon \rceil^{r_+} \cdot \left\lceil \max_{i\in[m]}\{f_i((Vu)_{\max})\}/\epsilon \right\rceil \cdot \lambda' m^{\lambda'} T_{\mathrm{LP}}.
\end{aligned}$$

Therefore, the algorithm ALG′ (for solving $\mathsf{P}(X)$) in Lemma 8 has runtime

$$\begin{aligned}
&\left\lceil \left(\frac{4(1+\gamma)k}{\epsilon W'_{\min}} \right)^{r_+} \right\rceil \cdot \lceil ((Vu)_{\max} - \min\{0, (Vu)_{\min}\})/\epsilon \rceil^{r_+} \\
&\cdot \left\lceil \max_{i\in[m]}\{f_i((Vu)_{\max})\}/\epsilon \right\rceil \cdot \lambda' m^{\lambda'} T_{\mathrm{LP}}.
\end{aligned}$$

Finally, by Lemma 8, our algorithm's runtime is

$$\begin{aligned}
&r_+ m \log_2 m + \lambda m^{\lambda} + \lambda r_+^2 m^{\lambda r_+} + \left\lceil \left(\frac{4(1+\gamma)k}{\epsilon W'_{\min}} \right)^{r_+} \right\rceil \cdot \left\lceil \frac{(Vu)_{\max} - \min\{0, (Vu)_{\min}\}}{\epsilon} \right\rceil^{r_+} \\
&\cdot \left\lceil \frac{\max_{i\in[m]}\{f_i((Vu)_{\max})\}}{\epsilon} \right\rceil \cdot \lambda' m^{\lambda r_+ + \lambda'} T_{\mathrm{LP}}.
\end{aligned}$$

Below we give the pseudo-code of our phase two algorithm in Proposition 8.

Algorithm 16: Phase Two

Input: Attention matrix $W = \mathrm{softmax}(QK^\top) \in \mathbb{R}_+^{n\times n}$, low-rank approximation $W' \in \mathbb{R}_+^{n\times n}$ with factorization $W' = AB^\top$ and element wise guarantee $1-\gamma \le W_{ij}/W'_{ij} \le 1+\gamma$, value matrix $V \in \mathbb{R}^{n\times d_v}$, user vector $u \in \mathbb{R}^{d_v}$, maximum number of recommended items k, candidate index set I, parameter $\epsilon > 0$.

Output: Solution x to Problem (Main).

Construct the instance of Problem ($\mathsf{P}(I)$) from the inputs

`// Low Non-negative Rank Approximation`

Form the instance of Problem ($\mathsf{P}'(I)$) by replacing W with W' in ($\mathsf{P}(I)$)

`// Enumeration of Partial Solutions`

$x \leftarrow$ **Run** Algorithm 12 on instance ($\mathsf{P}'(I)$) with parameter ϵ, which internally invokes:

`// Discretization of Auxiliary Problems`

▹ **Algorithm 13** with *Input:* Instance ($\mathsf{P}(X)$), parameter ϵ; *Calls:* Algorithm 14

`// Complete Linearization of Auxiliary Problems`

▹ **Algorithm 14** with *Input:* Instance $\mathsf{P}(X, t_{\ell'})$, parameters $\delta_1 \leftarrow \epsilon$, $\delta_2 \leftarrow k\epsilon$; *Calls:* Algorithm 15

`// Approximation of Linearized Auxiliary Problems via LP Rounding`

▹ **Algorithm 15** with *Input:* Instance $\mathsf{P}(X, t)$, thresholds $(\theta_1, \ldots, \theta_{r_+}, \zeta)$, parameters $\delta_1 \leftarrow \epsilon$, $\delta_2 \leftarrow k\epsilon$

return x

Appendix B

APPENDIX FOR CHAPTER 3

B.1 Preliminary Results and Proofs in Section 3.2

We first make a remark on Assumption 1 and Assumption 2. Both of the assumptions can be partly motivated by [113]. Their paper considered the stochastic arrival model (without predictions) where the reward functions and the resource consumption functions are all linear. Suppose that, in our problem under the stochastic arrival model and with the linearity assumptions, the prediction $\hat{\mu}_n$ is obtained by observing n historical arrivals, solving the problem offline, and taking the optimal offline dual variable to be $\hat{\mu}_n$. Then by Theorem 1 in [113], we have

$$||\hat{\mu}_n - \mu^*||_1 = O\left(\sqrt{\frac{\log\log n}{n}}\right),$$

which provides a guideline to set $\epsilon(T)$ in Assumption 1. For example, if $n = \sqrt{T}$, that is, we have observed $\sqrt{T}$ historic arrivals, then $\epsilon(T)$ can be set as $\epsilon(T) = \tilde{O}(T^{-1/4})$. More generally, if $n = f(T)$ for some function f such that $f(T) \to \infty$ as $T \to \infty$, then $\epsilon(T)$ can be set as $\epsilon(T) = \tilde{O}(f(T)^{-1/2})$.

[113] solved their problem by performing dual gradient descent (in their Algorithm 3). More specifically, at each time period they used gradient information to update a hypothetical dual variable μ'_t. In the proof of Theorem 5 in [113], they showed that the difference of the depletion time of each resource by following $\mu'_1 \ldots, \mu'_T$ and following the optimal dual variable is upper bounded by $O(\log(T)\log\log(T))$. This shows if the prediction is perfect in our problem, then $\mu'_1 \ldots, \mu'_T$ obtained by their Algorithm 3 is a particular sequence of dual variables for which Assumption 2 is satisfied.

We state two structural results regarding the duality of the offline problem.

Lemma 14 (Weak Duality)**.** $\mathrm{OPT}(\gamma) \le D(\mu \mid \gamma)$ *for every* $\mu \in \mathbb{R}^m_+$.

Lemma 15 (Duality Gap)**.** $\min_{\mu\in\mathbb{R}^m_+} D(\mu \mid \gamma) \le \mathrm{OPT}(\gamma) + (m+1)\overline{r}$.

Lemma 14 is the standard weak duality result. Lemma 15 states that, even without any convexity assumptions, the duality gap of our problem is upper bounded by

a constant that is independent from the time horizon T. This can be shown via Shapley-Folkman Theorem (see Proposition 5.26 of [34] for a detailed explanation).

Proof of Lemma 14. This proof appears in [22]. We include it for the sake of completeness. It holds for any $\mu \in \mathbb{R}_+^m$ that

$$\begin{aligned}
\mathrm{OPT}(\gamma) &= \begin{bmatrix} \max_{x_t \in \mathcal{X}_t} & \sum_{t=1}^T r_t(x_t) \\ \text{s.t.} & \sum_{t=1}^T g_t(x_t) \le \rho T \end{bmatrix} \\
&\le \max_{x_t \in \mathcal{X}} \left\{ \sum_{t=1}^T r_t(x_t) + \mu^\top \rho T - \mu^\top \sum_{t=1}^T g_t(x_t) \right\} \\
&= \sum_{t=1}^T r_t^*(\mu) + T\mu^\top \rho T \\
&= D(\mu \mid \gamma),
\end{aligned}$$

where the first inequality is because we relax the constraint $\sum_{t=1}^T g_t(x_t) \le \rho T$ and $\mu \ge 0$, and the last equality utilizes the definition of $r^*(\cdot)$. ∎

Proof of Proposition 1. Let $\mathrm{conv}(\mathcal{X}_t) \subset \mathbb{R}_+^d$ denote the convex hull of $\mathcal{X}_t$. For each t, define the function $\tilde{r}_t : \mathrm{conv}(\mathcal{X}_t) \to \mathbb{R}_+$ by

$$\begin{aligned}
\tilde{r}_t(\tilde{x}) = \sup \Bigg\{ \sum_{k=1}^{d+1} \alpha^k r_t(x^k) \;\Bigg|\; \tilde{x} = \sum_{k=1}^{d+1} \alpha^k x^k,\ x^k \in \mathcal{X}_t, \\
\sum_{k=1}^{d+1} \alpha^k = 1,\ \alpha^k \ge 0 \Bigg\} \qquad \forall \tilde{x} \in \mathrm{conv}(\mathcal{X}_t).
\end{aligned}$$

$\tilde{r}_t$ is concave regardless of whether r_t is concave or not, and it can be viewed as a "concavification" of r_t on $\mathrm{conv}(\mathcal{X}_t)$. Similarly, for each t, define the function $\tilde{g}_t : \mathrm{conv}(\mathcal{X}_t) \to \mathbb{R}_+^m$ by

$$\begin{aligned}
\tilde{g}_t(\tilde{x}) = \inf \Bigg\{ \sum_{k=1}^{d+1} \alpha^k g_t(x^k) \;\Bigg|\; \tilde{x} = \sum_{k=1}^{d+1} \alpha^k x^k,\ x^k \in \mathcal{X}_t, \\
\sum_{k=1}^{d+1} \alpha^k = 1,\ \alpha^k \ge 0 \Bigg\} \qquad \forall \tilde{x} \in \mathrm{conv}(\mathcal{X}_t).
\end{aligned}$$

$\tilde{g}_t$ is convex regardless of whether g_t is convex or not.

Let (P) denote the optimization problem in Equation (3.1). Consider the following convex relaxation ($\tilde{\text{P}}$) of the optimization problem in Equation (3.1):

$$\max_{x_t \in \text{conv}(\mathcal{X}_t)} \sum_{t=1}^{T} \tilde{r}_t(\tilde{x}_t) \quad \text{s.t.} \quad \sum_{t=1}^{T} \tilde{g}_t(\tilde{x}_t) \leq \rho T;$$

and its Lagrangian dual problem ($\tilde{\text{D}}$):

$$\min_{\mu \in \mathbb{R}_+^m} \sum_{t=1}^{T} \tilde{r}_t^*(\tilde{x}_t) + \mu^\top \rho T \quad \text{where} \quad r_t^*(\mu) = \sup_{x \in \text{conv}(\mathcal{X}_t)} \{r_t(\tilde{x}) - \mu^\top \tilde{g}_t(\tilde{x})\}.$$

Because $0 \in \text{conv}(\mathcal{X}_t)$, $\tilde{g}_t(0) = 0$ for all t, and $\rho > 0$, ($\tilde{\text{P}}$) satisfies Slater's condition. Therefore by strong duality $\sup(\tilde{\text{P}}) = \inf(\tilde{\text{D}})$. By an application of the Sharpley-Folkman Theorem ([34], Proposition 5.26), there exists an optimal solution $\{\tilde{x}_t^*\}_{t=1}^T$ of ($\tilde{\text{P}}$) with the following property: let $I \subset [T]$ be the set of time periods where $\tilde{x}_t^* \notin \mathcal{X}_t$ for $t \in I$, then $|I| \leq m + 1$ and $\sum_{t \in [T] \setminus I} g_t(\tilde{x}_t^*) \leq \rho T$. Let $\tilde{\mu}^*$ be the optimal dual variable of ($\tilde{\text{D}}$) that induces $\{\tilde{x}_t^*\}_{t=1}^T$, and let $\{x_t^{\tilde{\mu}^*}\}_{t=1}^T$ be the actions induced by $\tilde{\mu}^*$ in the original primal (P). We prove that

$$\sum_{t=1}^{T} r_t(x_t^{\tilde{\mu}^*}) \geq \sum_{t=1}^{T} r_t(\tilde{x}_t^*) - (\bar{g}/\underline{g} + 1)(m+1)\bar{r}.$$

Let $S \subset [T]$ be the set of time periods such that $x_s^{\tilde{\mu}^*} \neq \tilde{x}_s^*$ for $s \in S$, and let $J = S \setminus I$. Then J is the set of time periods where the resource constraint becomes active when choosing the action induced by $\tilde{\mu}^*$. Because $|I| \leq m + 1$ and $x_t^{\tilde{\mu}^*} = \tilde{x}_t^*$ for $t \in [T] \setminus S$,

$$\sum_{t \in J} g_t(\tilde{x}_t^*) - \sum_{t \in J} g_t(x_t^{\tilde{\mu}^*}) \leq (m+1)\bar{g}.$$

Therefore

$$|\{t \in J : \tilde{x}_t^* \neq 0\}| \leq (m+1)\bar{g}/\underline{g},$$

so

$$\sum_{t \in J} r_t(\tilde{x}_t^*) - \sum_{t \in J} r_t(x_t^{\tilde{\mu}^*}) \leq (m+1)\bar{r}\bar{g}/\underline{g}.$$

Further, we also have

$$\sum_{t \in I} r_t(\tilde{x}_t^*) - \sum_{t \in I} r_t(x_t^{\tilde{\mu}^*}) \leq (m+1)\bar{r}$$

and $x_t^{\tilde{\mu}^*} = \tilde{x}_t^*$ for $t \in [T] \setminus S$. These together gives

$$\sum_{t=1}^{T} r_t(x_t^{\tilde{\mu}^*}) \geq \sum_{t=1}^{T} r_t(\tilde{x}_t^*) - (\bar{g}/\underline{g} + 1)(m+1)\bar{r}.$$

Finally, since $(\tilde{\text{P}})$ is a relaxation of (P),

$$\sup(\tilde{\text{P}}) = \sum_{t=1}^{T} r_t(\tilde{x}_t^*) \geq \text{OPT}(\gamma),$$

so we have

$$\max_{\mu \in \mathbb{R}_+^m} R(\text{GRD}_\mu \mid \gamma) \geq \sum_{t=1}^{T} r_t(x_t^{\tilde{\mu}^*}) \geq \text{OPT}(\gamma) - (\bar{g}/\underline{g} + 1)(m+1)\bar{r}.$$

■

Proof of Proposition 2. Let $x'_t(\mu) = \arg\max_{x \in \{0,1\}} \{r'_t(x) - \mu^\top g_t(x)\}$. Note that $x'_t(\hat{\mu}) = 0$ if and only if $r_t(0) - \hat{\mu}^\top g_t(0) > r_t(1) + \epsilon_t - \hat{\mu}^\top g_t(1)$, which after rearranging gives $\hat{\mu}^\top g_t(1) > r_t(1) - r_t(0) + \epsilon_t$. Then for any μ_t such that $||\hat{\mu} - \mu_t||_1 \leq \zeta$, we have

$$\begin{aligned}
\mathbb{P}(x'_t(\mu_t) = 0 \mid x'_t(\hat{\mu}) = 0) &= \mathbb{P}\Big(\mu_t^\top g_t(1) > r_t(1) - r_t(0) + \epsilon_t \\
&\qquad \mid \hat{\mu}^\top g_t(1) > r_t(1) - r_t(0) + \epsilon_t\Big) \\
&= \frac{\Phi((\mu_t - \hat{\mu})^\top g_t(1)/\sigma)}{\Phi(\hat{\mu}^\top g_t(1)/\sigma)} \\
&\approx 1 - \frac{\zeta m \|g_t(1)\|_\infty}{\hat{\mu}^\top g_t(1)}.
\end{aligned}$$

where Φ is the cumulative distribution function of the standard normal distribution, and the second equality follows since $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$. The case where $x'_t(\hat{\mu}) = 1$ can be handled similarly. Therefore the probability of following $\hat{\mu}$ and following μ_t consume different amount of resources is $O(\zeta)$, so by independency Assumption 2 is satisfied with probability $1 - O(\zeta^{cT}) = 1 - \exp(-c\zeta T)$ for some $c > 0$. ■

We make the following observation, which follows since Proposition 1 shows there exists a perfect dual variable for every arrival sequence.

Observation 6. *If an arrival sequence γ is (λ, δ)-stationary, then*

$$\min_{k=1,\ldots,\lfloor \frac{1}{\delta} \rfloor} (OPT(\gamma_{1:k\delta T}) + OPT(\gamma_{k\delta T+1:T})) \geq (1-\lambda) OPT(\gamma).$$

Proof of Proposition 3. Fix any $\delta > 0$. If $R(\text{GRD}_\mu \mid \gamma) = o(T)$, then since the total amount of available resources ρT scales linearly in T and every single action consumes constants amount of resources, the Dual-Adjusted Greedy Algorithm with dual variable μ never depletes resources. Therefore

$$R(\text{GRD}_\mu \mid \gamma_{1:k\delta T}) + R(\text{GRD}_\mu \mid \gamma_{k\delta T+1:T}) = R(\text{GRD}_\mu \mid \gamma)$$

for every $1 \le k \le \lfloor \frac{1}{\delta} \rfloor$, which shows γ is (δ, λ)-stationary for every $\lambda > 0$. From now on we assume that $R(\mathrm{GRD}_\mu \mid \gamma) = \Theta(T)$.

For any time periods s, t and any amount of resources $\rho' T \in \mathbb{R}_+^m$, we use $R(\mathrm{GRD}_\mu \mid \gamma_{s:t}, \rho' T)$ to denote the amount of reward obtained by the Dual-Adjusted Greedy Algorithm with dual variable μ on the $\gamma_{s:t}$-subproblem *with available amount of resources* $\rho' T$.

Fix an integer $1 \le k \le \lfloor \frac{1}{\delta} \rfloor$. If $k\delta T = o(T)$, then

$$\begin{aligned} R(\mathrm{GRD}_\mu \mid \gamma_{k\delta T+1:T}) &\ge R(\mathrm{GRD}_\mu \mid \gamma_{k\delta T+1:T}, \rho T) - k\delta T(m\bar{g}\bar{r}/\underline{g}) \\ &\ge R(\mathrm{GRD}_\mu \mid \gamma) - k\delta T\bar{r} - k\delta T(m\bar{g}\bar{r}/\underline{g}), \end{aligned}$$

where the first inequality follows since any algorithm can consume at most $m\bar{g}$ amount of resources in ℓ_1-norm in a single time period, which can be translated to at most $m\bar{g}\bar{r}/\underline{g}$ amount of reward; the second inequality follows since any algorithm can obtain at most $k\delta T\bar{r}$ amount of rewards in the first $k\delta T\bar{r}$ time periods. Since $k\delta T \in o(T)$, this shows

$$R(\mathrm{GRD}_\mu \mid \gamma_{1:k\delta T}) + R(\mathrm{GRD}_\mu \mid \gamma_{k\delta T+1:T})) \ge R(\mathrm{GRD}_\mu \mid \gamma) - o(T).$$

Similarly, we can also show this if $T - k\delta T = o(T)$.

Suppose $k\delta T, T - k\delta T = \Theta(T)$. Let $\rho' T \le \rho T$ be the amount of resources that is consumed by the Dual-Adjusted Greedy Algorithm with dual variable μ and arrivals γ, then

$$R(\mathrm{GRD}_\mu \mid \gamma) = R(\mathrm{GRD}_\mu \mid \gamma, \rho' T)$$

and

$$R(\mathrm{GRD}_\mu \mid \gamma_{1:k\delta T}) = R(\mathrm{GRD}_\mu \mid \gamma_{1:k\delta T}, k\delta\rho' T).$$

Condition on $\gamma \sim \mathcal{P}^T$, for each time period t, let $X_t \in \mathbb{R}_+^m$ be the amount of resources consumed at time period t. Then X_t's are independent with $\mathbb{E}[X_t] = \rho'$ and $0 \le ||X_t||_\infty \le \bar{g}$. By Hoeffding's inequality we have

$$\mathbb{P}_{\gamma\sim\mathcal{P}^T}\left(\sum_{t=1}^{k\delta T}(X_t)_{m'} - k\delta\rho'_{m'}T \ge \bar{g}^2\sqrt{k\delta T\log T}\right) \le \frac{1}{T^2},$$

where $(X_t)_{m'}$ denotes the m'-th coordinate of X_t. So by union bound we have

$$\mathbb{P}_{\gamma\sim\mathcal{P}^T}\left(\sum_{t=1}^{k\delta T} X_t - k\delta\rho' T \ge \bar{g}^2\sqrt{k\delta T\log T}\right) \le \frac{m}{T^2}. \tag{B.1}$$

Therefore

$$\begin{aligned}
&k\delta\mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[R\left(\mathrm{GRD}_\mu \mid \gamma\right)\right] - \mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[R(\mathrm{GRD}_\mu \mid \gamma_{1:k\delta T})\right] \\
&= \mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[R\left(\mathrm{GRD}_\mu \mid \gamma_{1:k\delta T}, \sum_{t=1}^{k\delta T} X_t\right)\right] - \mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[R(\mathrm{GRD}_\mu \mid \gamma_{1:k\delta T}, k\delta\rho' T)\right] \\
&\le \left(1 - \frac{m}{T^2}\right)\mathbb{E}_{\gamma\sim\mathcal{P}^T}\Big[R\big(\mathrm{GRD}_\mu \mid \gamma_{1:k\delta T}, k\delta\rho' T + \bar{g}^2\sqrt{k\delta T\log T}\big) \\
&\qquad\qquad - R(\mathrm{GRD}_\mu \mid \gamma_{1:k\delta T}, k\delta\rho' T)\Big] + \frac{m}{T^2}\cdot\bar{r}k\delta T \\
&\le \bar{r}\bar{g}^2\sqrt{k\delta T\log T}/\underline{g} + m\bar{r}k\delta/T \\
&= o(T).
\end{aligned}$$

where the first inequality follows by conditioning on two cases given by Eq. (B.1) and noticing that any algorithm can obtain at most $k\delta T\bar{r}$ amount of rewards in the first $k\delta T\bar{r}$ time periods, and the second inequality follows since any algorithm can consume at most $m\bar{g}$ amount of resources in ℓ_1-norm in a single time period, which can be translated to at most $m\bar{g}\bar{r}/\underline{g}$ amount of reward. Similarly, we also have

$$(1-k\delta)\mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[R\left(\mathrm{GRD}_\mu \mid \gamma\right)\right] - \mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[R(\mathrm{GRD}_\mu \mid \gamma_{k\delta T+1:T})\right] = o(T).$$

Hence

$$\mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[R\left(\mathrm{GRD}_\mu \mid \gamma\right) - (R(\mathrm{GRD}_\mu \mid \gamma_{1:k\delta T}) + R(\mathrm{GRD}_\mu \mid \gamma_{k\delta T+1:T}))\right] = o(T).$$

Note that $R\left(\mathrm{GRD}_\mu \mid \gamma\right) - (R(\mathrm{GRD}_\mu \mid \gamma_{1:k\delta T}) + R(\mathrm{GRD}_\mu \mid \gamma_{k\delta T+1:T}))$ is a function from $\mathcal{P}^T$ to $\mathbb{R}$ such that each γ_t is drawn independently. Moreover, it satisfies the bounded differences property with bound $m\bar{g}\bar{r}/\underline{g}$ since any algorithm can consume at most $m\bar{g}$ amount of resources in ℓ_1-norm in a single time period, which can be translated to at most $m\bar{g}\bar{r}/\underline{g}$ amount of reward. Therefore by McDiarmid's inequality we have

$$\begin{aligned}
\mathbb{P}_{\gamma\sim\mathcal{P}^T}\Big[&R\big(\mathrm{GRD}_\mu \mid \gamma\big) - \Big(R(\mathrm{GRD}_\mu \mid \gamma_{1:k\delta T}) + R(\mathrm{GRD}_\mu \mid \gamma_{k\delta T+1:T})\Big) \\
&\ge o(T) + \frac{m\bar{g}\bar{r}}{\underline{g}}\sqrt{\frac{2T\log T}{3}}\Big] \le \frac{1}{T^3}.
\end{aligned}$$

This implies $R\left(\mathrm{GRD}_\mu \mid \gamma\right) - (R(\mathrm{GRD}_\mu \mid \gamma_{1:k\delta T}) + R(\mathrm{GRD}_\mu \mid \gamma_{k\delta T+1:T})) = o(T)$ with probability at least $1 - \frac{1}{T^3}$ for any $1 \le k\delta T \le T$. Since $k\delta T$ can take at most T values, by union bound

$$\min_{k=1,\ldots,\lfloor\frac{1}{\delta}\rfloor}(R(\mathrm{GRD}_\mu \mid \gamma_{1:k\delta T}) + R(\mathrm{GRD}_\mu \mid \gamma_{k\delta T+1:T})) \ge R(\mathrm{GRD}_\mu \mid \gamma) - o(T)$$

with probability at least $1 - \frac{1}{T^2}$. Because $R(\mathrm{GRD}_\mu \mid \gamma) = \Theta(T)$, this implies γ is (δ, λ)-stationary for every $\lambda > 0$ with probability at least $1 - \frac{1}{T^2}$. ■

B.2 Details in Section 3.3.1

For completeness, we discuss the *Mirror Descent Algorithm* (MDA) given in [22].

Algorithm 17: Mirror Descent Algorithm (MDA)

Inputs: Initial dual solution μ_1, total time periods T, initial resources $G_1 = \rho T$, reference function $h(\cdot) : \mathbb{R}^m \to \mathbb{R}$, and step-size η

for *t from* 1 *to T* **do**

Receive request $(r_t, g_t, \mathcal{X}_t)$

Make the primal decision x_t and update the remaining resources G_t:

$$x_t \in \arg\max_{x \in \mathcal{X}_t, g_t(x) \leq G_t} \left\{ r_t(x) - \mu_t^\top g_t(x) \right\}$$
$$G_{t+1} \leftarrow G_t - g_t(x_t).$$

Obtain a sub-gradient of the dual function:

$$\phi_t \leftarrow -g_t(x_t) + \rho.$$

Update the dual variable by mirror descent:

$$\mu_{t+1} \leftarrow \arg\min_{\mu \in \mathbb{R}_+^m} \phi_t^\top \mu + \frac{1}{\eta} V_h(\mu, \mu_t), \tag{B.2}$$

where $V_h(x, y) := h(x) - h(y) - \nabla h(y)^\top (x - y)$ is the Bregman divergence.

end

The Mirror Descent Algorithm takes an initial dual variable, a step-size, and a *reference function* as inputs. At each time period t, the algorithm takes the action induced by the current dual variable μ_t, and performs a first-order update on the dual variable. For the updating step, note we can write the dual function in Equation (3.3) as $D(\mu \mid \gamma) := \sum_{t=1}^T D_t(\mu \mid \gamma)$ where the t-th term of the dual function is given by $D_t(\mu \mid \gamma) = r_t^*(\mu) + \mu^\top \rho$. Then it follows that $\phi_t := -g_t(x_t) + \rho$ is a sub-gradient of $D_t(\mu \mid \gamma)$ at μ_t under our assumptions by Danskin's Theorem (see, e.g., Proposition B.25 in [33]), and the algorithm uses ϕ_t to update the dual variable by performing a mirror descent step in Equation (B.2) with step-size η and reference function $h(\cdot)$. Intuitively, the Mirror Descent Algorithm tries to find dual variables via gradient information such that these dual variables induce actions with good primal performances. For more on mirror descent algorithms in general, see [30, 78, 120, 138].

We state the standard assumptions on choosing the reference function $h(\cdot)$ for mirror descent algorithms [30, 39, 119, 120]. These assumptions are applicable to all algorithms in our paper.

(a) $h(\mu)$ is either differentiable or essentially smooth [27] and Lipschitz in $\mathbb{R}_+^m$;

(b) $h(\mu)$ is σ-strongly convex with respect to the ℓ_1-norm in $\mathbb{R}_+^m$, i.e.,

$$h(\mu_1) \geq h(\mu_2) + \nabla h(\mu_2)^\top (\mu_1 - \mu_2) + \frac{\sigma}{2}||\mu_1 - \mu_2||_1^2$$

for all $\mu_1, \mu_2 \in \mathbb{R}_+^m$.

(c) $h(\mu)$ coordinately-wise separable, i.e., $h(\mu) = \sum_{j=1}^m h_j(\mu_j)$ where $h_j : \mathbb{R}_+ \to \mathbb{R}$ is an univariate function. Moreover, for every resource j the function h_j is σ'-strongly convex with respect to the ℓ_1-norm over $[0, \mu_j^{\max}]$ where $\mu_j^{\max} := \bar{r}/\rho_j + 1$.

B.3 Proofs in Section 3.3.3

Proof of Proposition 7. Let c be a positive integer such that $c > \max\{\frac{K}{\lambda - \lambda'}, \frac{1-\lambda}{\delta}\}$. Set $\rho = 1$, $\alpha^* = 1/c\delta$, and $\bar{r} = (\alpha^* - 1)/(1 - \lambda - 1/\alpha^*)$, then by our choice of c we have $\lambda < 1 - 1/\alpha^*$ and $\bar{r} > \alpha^*$. Consider two different types of arrivals $\gamma^1 = (r^1, g^1, \mathcal{X})$ and $\gamma^2 = (r^2, g^2, \mathcal{X})$, where $\mathcal{X} = \{0, 1\}$ (one can think of this as {reject, accept}). Set $r^1(1) = 1, g^1(1) = 1, r^2(1) = \bar{r}$, and $g^2(1) = \alpha^*$. Let $\hat{\mu} = 1 + 1/\log(T)$ be the prediction, then following $\hat{\mu}$ means taking action 0 for γ^1 and taking action 1 for γ^2. Because $\bar{r}/\alpha^* > \hat{\mu}$, one can verify that Assumptions 1 and 2 are satisfied.

Consider the following two instances.

- Instance one: the arrivals are stochastic where the state space is $\mathcal{S} = \{\gamma^1\}$, i.e., $\gamma_t = \gamma^1$ for every $t = 1, \ldots, T$. In this instance the optimum is to take action 1 for all arrivals. Note that following $\hat{\mu}$ would take action 0 for all arrivals, which means the prediction has bad quality.

- Instance two: the arrivals are adversarial where $\gamma_t = \gamma^1$ for $t = 1, \ldots, \frac{\alpha^*-1}{\alpha^*}T$ and $\gamma_t = \gamma^2$ for $t = \frac{\alpha^*-1}{\alpha^*}T + 1, \ldots, T$. In this instance the optimum is to take action 0 for γ^1 and take action 1 for γ^2. We have $\mathrm{PRD}(\gamma) = \mathrm{OPT}(\gamma) = \frac{\bar{r}}{\alpha^*}T$, which means the prediction is perfect. Moreover, since $\delta = 1/c\alpha^*$, one can verify that

$$\begin{aligned} & \min_{k=1,\ldots,\lfloor \frac{1}{\delta} \rfloor} \mathrm{OPT}(\gamma_{1:k\delta T}) + \mathrm{OPT}(\gamma_{k\delta T+1:T}) \\ = \; & \mathrm{OPT}(\gamma_{1:\frac{\alpha^*-1}{\alpha^*}T}) + \mathrm{OPT}(\gamma_{\frac{\alpha^*-1}{\alpha^*}T+1:T}) \\ = \; & \frac{\alpha^* - 1}{\alpha^*}T + \frac{\bar{r}}{\alpha^{*2}}T. \end{aligned}$$

Then we get

$$\begin{aligned} & 1 - \left(\mathrm{OPT}(\gamma_{1:\frac{\alpha^*-1}{\alpha^*}T}) + \mathrm{OPT}(\gamma_{\frac{\alpha^*-1}{\alpha^*}T+1:T})\right)/\mathrm{OPT}(\gamma) \\ = \quad & 1 - \left(\frac{\alpha^*-1}{\alpha^*}T + \frac{\bar{r}}{\alpha^{*2}}T\right)/\frac{\bar{r}}{\alpha^*}T \\ = \quad & 1 - \frac{1}{\alpha^*} - \frac{\alpha^*-1}{\bar{r}} \\ = \quad & \lambda, \end{aligned}$$

where the last equality follows by our choice of $\bar{r} = (\alpha^* - 1)/(1 - \lambda - 1/\alpha^*)$. Therefore γ is λ-nonstationary with respect to δ.

Note that no algorithm can distinguish instance one and instance two before time period $t = \frac{\alpha^*-1}{\alpha^*}T + 1$. For any algorithm, assume in instance one it satisfies $\mathrm{Regret}(\mathrm{ALG}) = o(T)$, then since the optimum is to take action 1 for all arrivals, at time period $t = \frac{\alpha^*-1}{\alpha^*}T+1$ the amount of resources left is at most $\frac{1}{\alpha^*}T-o(T)$. Therefore in instance two the algorithm can take action 1 for at most $\frac{1}{\alpha^{*2}}T+o(1)$ time periods, so the total rewards gained in instance two satisfies $R(\mathrm{ALG} \mid \gamma) = \frac{\alpha^*-1}{\alpha^*}T + \frac{\bar{r}}{\alpha^{*2}}T + o(T)$. Because $\mathrm{PRD}(\gamma) = \frac{\bar{r}}{\alpha^*}T$, in instance two we have

$$\begin{aligned} & \limsup_{T\to\infty}\left\{\frac{1}{T}\left((1-\lambda')\max\left\{\frac{1}{\alpha^*}\mathrm{OPT}(\gamma), \mathrm{PRD}(\gamma)\right\} - R(\mathrm{ALG} \mid \gamma)\right)\right\} \\ = \quad & \limsup_{T\to\infty}\left\{\frac{1}{T}\left((\lambda-\lambda')\frac{\bar{r}}{\alpha^*}T - o(T)\right)\right\} \\ = \quad & (\lambda-\lambda')\frac{\bar{r}}{\alpha^*} \\ > \quad & K\delta\bar{r}, \end{aligned}$$

where the last inequality follows since $c > K/(\lambda - \lambda')$ and $\delta = \alpha^*/c$.

■

B.4 Description of Algorithm 18

We list some notations used in Algorithm 18, which follow notations in [43]. Given initial dual solution μ_1 and step-size η:

(a) Let $x_t(\mu_1, \eta)$ and $\mu_t(\mu_1, \eta)$ be the action we take and the the dual variable we get after $t - 1$ iterations of the Mirror Descent Algorithm with initial dual solution μ_1, step-size η, and the same requests as the requests that Algorithm 18 received so far;

(b) Define $\theta_t(\mu_1, \eta) := \max_{s \le t} ||\mu_1 - \mu_s(\mu_1, \eta)||_1$ to be the maximum ℓ_1-distance from any updated dual variable used in the Mirror Descent Algorithm before time t to the initial dual variable;

(c) Define $\Phi_t(\mu_1, \eta) := \sum_{s=1}^{t} || - g_s(x_s(\mu_1, \eta)) + \rho||_\infty^2$ to be the running sum of squared ℓ_∞-norms of the dual functions' sub-gradients.

A high-level intuition behind the choices of step sizes is that, it is well-known [142] that the hindsight (asymptotically) optimal step size is η that satisfies

$$\eta = \frac{||\mu_1 - \mu^*||_1}{\sqrt{\Phi_T(\mu_1, \eta)}}.$$

Because $||\mu_1 - \mu^*||_1$ and $\Phi_T(\mu_1, \eta)$ are unknown a priori, at each time period t we use $\theta_t(\mu_1, \eta)$ as an approximation of $||\mu_1 - \mu^*||_1$ and use $\Phi_t(\mu_1, \eta)$ as an approximation of $\Phi_T(\mu_1, \eta)$, and these approximations can be proven to be accurate. Then we use bisection to find an approximate solution of the implicit function

$$\eta_t = \frac{\theta_t(\mu_1, \eta_t)}{\sqrt{\alpha \Phi_t(\mu_1, \eta_t) + \beta}},$$

where α, β are damping parameters. For a more detailed explanation, see [43]. Note that the Stochastic Arrival Algorithm does not need to know the accuracy parameter a.

B.5 Proofs in Section 3.4.1 and 3.4.2

Proof of Proposition 8. The proof technique is similar to the proof of Theorem 1 in [22], which we largely borrow. We break down the proof in three steps.

Step 1 (Primal performance.) First, we define the stopping time τ_A of Algorithm 18 as the first time less than T that there exists resource j such that $\sum_{t=1}^{\tau_A} \left(g_t\left(x_t\right)\right)_j + \bar{g} \ge \rho_j T$. Notice that τ_A is a random variable, and moreover, we will not violate the resource constraints before the stopping time τ_A. We here study the primal-dual gap until the stopping-time τ_A. Notice that before the stopping time τ_A, Algorithm 18 performs the mirror descent steps on the dual function with fine-tuned step sizes.

Consider a time $t \le \tau_A$ so that actions are not constrained by resources. Then the algorithm takes the action $x_t \in \arg\max_{x \in \mathcal{X}_t} \left\{r_t(x) - \mu_t^\top g_t(x)\right\}$, so we have that

$$r_t\left(x_t\right) = r_t^*\left(\mu_t\right) + \mu_t^\top g_t\left(x_t\right).$$

Algorithm 18: Stochastic Arrival Algorithm (SA)

Inputs: Prediction $\hat{\mu}$, total time periods T, initial resources $G_1 = \rho T$, reference function $h(\cdot) : \mathbb{R}^m \to \mathbb{R}$, and initial step-size η_1

Initialize $\mu_1 \leftarrow \hat{\mu}$

for *t from* 1 *to T* **do**

Receive request $(r_t, g_t, \mathcal{X}_t)$

Make the primal decision x_t and update the remaining resources G_t:

$$x_t \in \arg\max_{x \in \mathcal{X}_t, g_t(x) \le G_t} \left\{ r_t(x) - \mu_t^\top g_t(x) \right\}$$
$$G_{t+1} \leftarrow G_t - g_t(x_t).$$

Obtain a sub-gradient of the dual function:

$$\phi_t \leftarrow -g_t(x_t) + \rho.$$

Update the dual variable by mirror descent:

$$\mu_{t+1} \leftarrow \arg\min_{\mu \in \mathbb{R}_+^m} \phi_t^\top \mu + \frac{1}{\eta_t} V_h(\mu, \mu_t),$$

where $V_h(x, y) := h(x) - h(y) - \nabla h(y)^\top (x - y)$ is the Bregman divergence.

Tune the step size:

for $k = 2, 4, 8, 16, \ldots$ **do**

$t_k \leftarrow \lfloor t/2k \rfloor$

$\alpha^{(k)} \leftarrow 32^2 C_t^{(k)}$, $\beta^{(k)} \leftarrow (32 C_t^{(k)}(\bar{g} + \bar{\rho}))^2$ where $C_t^{(k)} := 2k + \log\left(60T \log^2(6t)\right)$

if `Root Finding Bisection`$(\eta_t, 2^{2^k}\eta_t; t_k, \alpha^{(k)}, \beta^{(k)}) < \infty$ **then**

$\eta_{t+1} \leftarrow$ `Root Finding Bisection`$(\eta_t, 2^{2^k}\eta_t; t_k, \alpha^{(k)}, \beta^{(k)})$.

end

end

end

Function `Root Finding Bisection`$(\eta_{\text{lo}}, \eta_{\text{hi}}; t', \alpha, \beta)$**:**

$\psi(\cdot) := \eta \to \theta_{t'}(\hat{\mu}, \eta) / \sqrt{\alpha \Phi_{t'}(\hat{\mu}, \eta) + \beta}$

if $\eta_{\text{hi}} \le \psi(\eta_{\text{hi}})$ **then return** ∞

if $\eta_{\text{lo}} > \psi(\eta_{\text{lo}})$ **then return** η_{lo}

while $\eta_{\text{hi}} > 2\eta_{\text{lo}}$ **do**

$\eta_{\text{mid}} \leftarrow \sqrt{\eta_{\text{hi}} \eta_{\text{lo}}}$

if $\eta_{\text{mid}} \le \psi(\eta_{\text{mid}})$ **then** $\eta_{\text{lo}} \leftarrow \eta_{\text{mid}}$ **else** $\eta_{\text{hi}} \leftarrow \eta_{\text{mid}}$.

end

if $\theta_{t'}(\hat{\mu}, \eta_{\text{hi}}) \le \theta_{t'}(\hat{\mu}, \eta_{\text{lo}}) \frac{\psi(\eta_{\text{hi}})}{\eta_{\text{hi}}}$ **then return** η_{hi} **else return** η_{lo}.

End Function

Let

$$\bar{D}(\mu \mid \mathcal{P}) = \frac{1}{T}\mathbb{E}_{\gamma\sim\mathcal{P}^T}[D(\mu \mid \gamma)] = \mathbb{E}_{(r,g,\mathcal{X})\sim\mathcal{P}}\left[r^*\left(\mu_t\right)\right] + \mu_t^\top \rho$$

be the expected dual objective at μ when requests are drawn i.i.d. from $\mathcal{P} \in \Delta(\mathcal{S})$. Let $\xi_t = \{\gamma_0, \ldots, \gamma_t\}$ and $\sigma\left(\xi_t\right)$ be the sigma-algebra generated by ξ_t. Adding the last two equations and taking expectations conditional on $\sigma\left(\xi_{t-1}\right)$ we obtain, because $\mu_t \in \sigma\left(\xi_{t-1}\right)$ and $(r_t, g_t, \mathcal{X}_t) \sim \mathcal{P}$, that

$$\begin{aligned}\mathbb{E}[r_t(x_t) \mid \sigma(\xi_{t-1})] &= \mathbb{E}_{(r,g,\mathcal{X})\sim\mathcal{P}}[f^*(\mu_t)] + \mu_t^\top \rho \\ &\quad + \mu_t^\top(\mathbb{E}[g_t(x_t) \mid \sigma(\xi_{t-1})] - \rho) \\ &= \bar{D}(\mu_t \mid \mathcal{P}) - \mathbb{E}\left[\mu_t^\top(\rho - g_t(x_t)) \mid \sigma(\xi_{t-1})\right], \end{aligned} \tag{B.3}$$

where the second equality follows the definition of the dual function.

Consider the process

$$Z_t = \sum_{s=1}^{t} \mu_s^\top\left(a_s - b_s\left(x_s\right)\right) - \mathbb{E}\left[\mu_s^\top\left(a_s - b_s\left(x_s\right)\right) \mid \sigma\left(\xi_{s-1}\right)\right],$$

which is martingale with respect to ξ_t (i.e., $Z_t \in \sigma\left(\xi_t\right)$ and $\mathbb{E}\left[Z_{t+1} \mid \sigma\left(\xi_t\right)\right] = Z_t$). Since τ_A is a stopping time with respect to ξ_t and τ_A is bounded, the Optional Stopping Theorem implies that $\mathbb{E}\left[Z_{\tau_A}\right] = 0$. Therefore,

$$\mathbb{E}\left[\sum_{t=1}^{\tau_A} \mu_t^\top\left(\rho - g_t\left(x_t\right)\right)\right] = \mathbb{E}\left[\sum_{t=1}^{\tau_A} \mathbb{E}\left[\mu_t^\top\left(\rho - g_t\left(x_t\right)\right) \mid \sigma\left(\xi_{t-1}\right)\right]\right].$$

Using a similar martingale argument for $f_t\left(x_t\right)$ and summing Eq. (B.3) from $t = 1, \ldots, \tau_A$ we obtain that

$$\begin{aligned}\mathbb{E}\left[\sum_{t=1}^{\tau_A} r_t\left(x_t\right)\right] &= \mathbb{E}\left[\sum_{t=1}^{\tau_A} \bar{D}\left(\mu_t \mid \mathcal{P}\right)\right] - \mathbb{E}\left[\sum_{t=1}^{\tau_A} \mu_t^\top\left(\rho - g_t\left(x_t\right)\right)\right] \\ &\geq \mathbb{E}\left[\tau_A \bar{D}\left(\bar{\mu}_{\tau_A} \mid \mathcal{P}\right)\right] - \mathbb{E}\left[\sum_{t=1}^{\tau_A} \mu_t^\top\left(\rho - g_t\left(x_t\right)\right)\right]. \end{aligned} \tag{B.4}$$

where the inequality follows from denoting $\bar{\mu}_{\tau_A} = \frac{1}{\tau_A}\sum_{t=1}^{\tau_A} \mu_t$ to be the average dual variable and using that the dual function is convex.

Step 2 (Complementary slackness). Consider the sequence of functions $w_t(\mu) = \mu^\top\left(\rho - g_t\left(x_t\right)\right)$, which capture the complementary slackness at time t. The sub-gradients are given by $\nabla_\mu w_t(\mu) = \rho - g_t\left(x_t\right)$, which are bounded as follows $\left\|\nabla_\mu w_t(\mu)\right\|_\infty \leq \left\|g_t\left(x_t\right)\right\|_\infty + \|\rho\|_\infty \leq \bar{g} + \bar{\rho}$. Therefore, Algorithm 18 applies online mirror descent to the sequence of functions $w_t(\mu)$ with the fine-tuned step sizes. To analyze the performance, we use the following lemma from [43].

Lemma 16 (Theorem 4 in [43]). *Under the assumptions and notations of our paper, the online mirror descent in Algorithm 18 with the proposed step sizes satisfies, with probability at least* $1 - \frac{1}{T}$*, that*

$$\sum_{t=1}^{\tau_A} \left(w_t\left(\mu_t\right) - w_t(\mu^*)\right) \leq CT^{\frac{1}{2}} ||\mu_1 - \mu^*||_1 \cdot \text{polylog}(T)^1$$

where $C > 0$ *is some constant.*

Because $||\mu_1 - \mu^*||_1 = ||\hat{\mu} - \mu^*||_1 \leq \kappa T^{-a}$, Lemma 16 states that $\sum_{t=1}^{\tau_A} w_t\left(\mu_t\right) - w_t(\mu^*) \leq \kappa C T^{\frac{1}{2}-a} \cdot \text{polylog}(T)$ with probability at least $1 - \frac{1}{T}$.

Step 3 (Putting it all together). For any $\mathcal{P} \in \Delta(\mathcal{S})$ and $\tau_A \in [0, T]$ we have that

$$\begin{aligned} \mathbb{E}_{\gamma \sim \mathcal{P}^T}[\text{OPT}(\gamma)] &= \frac{\tau_A}{T} \mathbb{E}_{\gamma \sim \mathcal{P}^T}[\text{OPT}(\gamma)] + \frac{T - \tau_A}{T} \mathbb{E}_{\gamma \sim \mathcal{P}^T}[\text{OPT}(\gamma)] \\ &\leq \tau_A \bar{D}\left(\bar{\mu}_{\tau_A} \mid \mathcal{P}\right) + (T - \tau_A)\bar{r}. \end{aligned} \tag{B.5}$$

where the inequality uses Lemma 14 and the fact that $\text{OPT}(\gamma) \leq \bar{r}T$. Therefore, with probability at least $1 - \frac{1}{T}$,

$$\begin{aligned} \text{Regret}(\text{SA} \mid \mathcal{P}) &= \mathbb{E}_{\gamma \sim \mathcal{P}^T}[\text{OPT}(\gamma) - R(\text{SA} \mid \gamma)] \\ &\leq \mathbb{E}_{\gamma \sim \mathcal{P}^T}\left[\text{OPT}(\gamma) - \sum_{t=1}^{\tau_A} r_t(x_t)\right] \\ &\leq \mathbb{E}_{\gamma \sim \mathcal{P}^T}\left[\text{OPT}(\gamma) - \tau_A D(\bar{\mu}_{\tau_A} \mid \mathcal{P}) + \sum_{t=1}^{\tau_A} w_t(\mu_t)\right] \\ &\leq \mathbb{E}_{\gamma \sim \mathcal{P}^T}\Big[\text{OPT}(\gamma) - \tau_A D(\bar{\mu}_{\tau_A} \mid \mathcal{P}) + \sum_{t=1}^{\tau_A} w_t(\mu^*) \\ &\qquad + CT^{\frac{1}{2}-a} \cdot \text{polylog}(T)\Big] \\ &\leq \mathbb{E}_{\gamma \sim \mathcal{P}^T} \underbrace{\left[(T - \tau_A)\bar{r} + \sum_{t=1}^{\tau_A} w_t(\mu^*) + CT^{\frac{1}{2}-a} \cdot \text{polylog}(T)\right]}_{\clubsuit}. \end{aligned} \tag{B.6}$$

where the first inequality follows from using that $\tau_A \leq T$ together with $r_t(\cdot) \geq 0$ to drop all requests after τ_A; the second is from Eq. (B.4); the third follows from Lemma 16; and the last from Eq. (B.5).

[1] Polylog(T) hides logarithmic terms in T. For explicit expressions see [43].

Note that $\sum_{t=1}^{\tau_A} w_t(\mu^*) \leq \sum_{t=1}^{\tau_A} w_t(\mu)$ for every $\mu \in \mathbb{R}_+^m$. We now discuss the choice of $\mu \in \mathbb{R}_+^m$ in order to upper bound $\sum_{t=1}^{\tau_A} w_t(\mu^*)$. If $\tau_A = T$, then set $\mu = 0$ to obtain that $\clubsuit \leq CT^{\frac{1}{2}-a} \cdot \text{polylog}(T)$. If $\tau_A < T$, then there exists a resource $j \in [m]$ such that $\sum_{t=1}^{\tau_A} \left(g_t\left(x_t\right)\right)_j + \bar{g} \geq \rho_j T$. Set $\mu = \left(\bar{r}/\rho_j\right) e_j$ with e_j being the j-th unit vector. This yields

$$\begin{aligned} & \sum_{t=1}^{\tau_A} w_t(\mu^*) \leq \sum_{t=1}^{\tau_A} w_t(\mu) = \sum_{t=1}^{\tau_A} \mu^\top \left(\rho - g_t\left(x_t\right)\right) \\ = \quad & \frac{\bar{r}}{\rho_j} \sum_{t=1}^{\tau_A} \left(\rho_j - \left(g_t\left(x_t\right)\right)_j\right) \leq \frac{\bar{r}}{\rho_j} \left(\tau_A \rho_j - \rho_j T + \bar{g}\right) = \frac{\bar{r}}{\rho_j} \bar{g} - \bar{r}\left(T - \tau_A\right), \end{aligned}$$

where the inequality follows because of the definition of the stopping time τ_A. Therefore, using that $\rho_j \geq \underline{\rho}$ for every resource $j \in [m]$, we have

$$\clubsuit \leq \frac{\bar{r}\bar{g}}{\underline{\rho}} + CT^{\frac{1}{2}-a} \cdot \text{polylog}(T).$$

Therefore $\text{Regret(SA)} \leq \frac{\bar{r}\bar{g}}{\underline{\rho}} + CT^{\frac{1}{2}-a} \cdot \text{polylog}(T)$ with probability at least $1 - \frac{1}{T}$. We conclude by noting that $\text{Regret(SA)} \leq \mathbb{E}_{\gamma \sim \mathcal{P}^T}[\text{OPT}(\gamma)] \leq \bar{r}T$, so we have

$$\text{Regret(SA)} \leq \frac{\bar{r}\bar{g}}{\underline{\rho}} + \bar{r} + CT^{\frac{1}{2}-a} \cdot \text{polylog}(T) \in \tilde{O}(\max\{T^{\frac{1}{2}-a}, 1\}).$$

■

Proof of Proposition 9. By Assumption 1, there exists a function $\psi(T)$ such that $||\hat{\mu} - \mu^*||_1 \leq \psi(T)$ and $\psi(T) = o(\epsilon(T))$. We break down the proof into two lemmas, which compares $R(\text{AA} \mid \gamma)$ with $\frac{1}{\alpha^*}\text{OPT}(\gamma)$ and $R(\text{PRD} \mid \gamma)$ separately.

Lemma 17. *Consider the Adversarial Arrival Algorithm* (AA) *under the adversarial arrival model. Given a prediction $\hat{\mu}$ with accuracy parameter a, it holds that:*

$$\limsup_{T \to \infty} \sup_{\gamma \in \mathcal{S}^T} \left\{ \frac{1}{T} \left(\frac{1}{\alpha^*} \text{OPT}(\gamma) - R(\text{AA} \mid \gamma) \right) \right\} \leq 0.$$

Proof of Lemma 17. The proof is drawn from the proof of Theorem 2 in [22]. The proof contains three steps, which is similar to the proof of Proposition 8.

Step 1 (Primal performance.) Fix an arrival sequence $\gamma \in \mathcal{S}^T$ and let $x^* \in \mathcal{X}_t$ be an optimal action in $\text{OPT}(\gamma)$ at time t. Let τ_A be the stopping time of Algorithm 2, which is defined similarly as in the proof of Proposition 8, then for $t \leq \tau_A$ we have

$x_t \in \arg\max_{x\in\mathcal{X}_t} \left\{r_t(x) - \mu_t^\top g_t(x)\right\}$, and thus $r_t\left(x_t\right) \geq r_t\left(x_t^*\right) - \mu_t^\top \left(g_t\left(x_t^*\right) - g_t\left(x_t\right)\right)$ and $0 = r_t(0) \leq r_t\left(x_t\right) - \mu_t^\top g_t\left(x_t\right)$. Therefore

$$\begin{aligned}
\alpha^* r_t\left(x_t\right) &= r_t\left(x_t\right) + (\alpha^* - 1) r_t\left(x_t\right) \\
&\geq r_t\left(x_t^*\right) + \mu_t^\top g_t\left(x_t\right) - \mu_t^\top g_t\left(x_t^*\right) + (\alpha^* - 1)\left(\mu_t^\top g_t\left(x_t\right)\right) \\
&= r_t\left(x_t^*\right) - \alpha^* \mu_t^\top \left(\rho - g_t\left(x_t\right)\right) + \alpha^* \mu_t^\top \rho - \mu_t^\top g_t\left(x_t^*\right) \\
&\geq r_t\left(x_t^*\right) - \alpha^* \mu_t^\top \left(\rho - g_t\left(x_t\right)\right),
\end{aligned}$$

where the second inequality is because $\alpha^* \mu_t^\top \rho - \mu_t^\top g_t\left(x_t^*\right) \geq 0$ by our definition of α^* and the fact that $\mu_t \geq 0$. Summing up over $t = 1, \ldots, \tau_A$ yields

$$\alpha^* \sum_{t=1}^{\tau_A} r_t\left(x_t\right) \geq \sum_{t=1}^{\tau_A} r_t\left(x_t^*\right) - \alpha^* \sum_{t=1}^{\tau_A} \mu_t^\top \left(\rho - g_t\left(x_t\right)\right). \tag{B.7}$$

Step 2 (Complementary slackness). Denoting, as before, $w_t(\mu) = \mu^\top \left(\rho - b_t\left(x_t\right)\right)$. As we have seen in the step 2 in the proof of Proposition 8 (the analysis is deterministic in nature), Algorithm 2 performs online mirror descent to the sequence of functions $w_t(\mu)$ with step size $\eta = c\epsilon(T)/T$ where $c > 0$ is an arbitrary scaling constant. By our assumption that the reference function $h(\cdot)$ is Lipschitz, there exists a constant $L > 0$ such that $V_h(\mu', \mu'') \leq L||\mu' - \mu''||_1$ for all $\mu', \mu'' \in \mathbb{R}_+^m$. By a standard result on online mirror descent (see, e.g., Appendix G of [22]), we have

$$\begin{aligned}
\sum_{t=1}^{\tau_A} w_t(\mu_t) &\leq \sum_{t=1}^{\tau_A} w_t(\mu^*) + \frac{(\bar{g} + \bar{\rho})^2 \eta}{2\sigma} \tau_A + \frac{1}{\eta} V_h(\mu^*, \mu_1) \\
&\leq \sum_{t=1}^{\tau_A} w_t(\mu^*) + \frac{c(\bar{g} + \bar{\rho})^2}{2\sigma} \epsilon(T) + \frac{\kappa L \psi(T) T}{c\epsilon(T)},
\end{aligned} \tag{B.8}$$

where the first inequality is the standard online mirror descent result, and the second inequality follows by the step size $\eta = c\epsilon(T)/T$ and the fact that $||\mu_1 - \mu^*||_1 = ||\hat{\mu} - \mu^*||_1 \leq \psi(T)$.

Step 3 (Putting it all together). We have

$$\begin{aligned}\mathrm{OPT}(\gamma) - \alpha^* R(\mathrm{AA} \mid \gamma) &\leq \sum_{t=1}^{T} r_t(x_t^*) - \alpha^* \sum_{t=1}^{\tau_A} r_t(x_t) \\ &\leq \sum_{t=\tau_A+1}^{T} r_t(x_t^*) + \alpha^* \sum_{t=1}^{\tau_A} w_t(\mu_t) \\ &\leq (T - \tau_A) \cdot \bar{r} + \alpha^* \sum_{t=1}^{\tau_A} w_t(\mu^*) \\ &\quad + \alpha^* \left(\frac{c(\bar{g} + \bar{\rho})^2}{2\sigma} \epsilon(T) + \frac{\kappa L \psi(T) T}{c\epsilon(T)} \right).\end{aligned}$$

where the first inequality follows because $\tau_A \leq T$ and $r_t(\cdot) \geq 0$, the second inequality is from Eq. (B.7), and the third inequality utilizes $r_t\left(x_t^*\right) \leq \bar{r}$ and Eq. (B.8). Similar to the proof of Proposition 8, we note that $\sum_{t=1}^{\tau_A} w_t(\mu^*) \leq \sum_{t=1}^{\tau_A} w_t(\mu)$ for every $\mu \in \mathbb{R}_+^m$ and discuss the choice of $\mu \in \mathbb{R}_+^m$ in order to upper bound $\sum_{t=1}^{\tau_A} w_t(\mu^*)$. If $\tau_A = T$, then set $\mu = 0$, and the result follows. If $\tau_A < T$, then there exists a resource $j \in [m]$ such that $\sum_{t=1}^{\tau_A} \left(g_t\left(x_t\right)\right)_j + \bar{g} \geq \rho_j T$. Set $\mu = \left(\bar{r} / \left(\alpha^* \rho_j\right)\right) e_j$ where e_j is the j-th unit vector and repeat the steps of the stochastic arrivals case to obtain:

$$\mathrm{OPT}(\gamma) - \alpha^* R(\mathrm{AA} \mid \gamma) \leq \frac{\bar{r}\bar{g}}{\underline{\rho}} + \alpha^* \left(\frac{c(\bar{g} + \bar{\rho})^2}{2\sigma} \epsilon(T) + \frac{\kappa L \psi(T) T}{c\epsilon(T)} \right),$$

which finishes the proof by noticing that $\epsilon(T)$ and $\psi(T)T/\epsilon(T)$ are both sub-linear in T. ■

Lemma 18. *Consider the Adversarial Arrival Algorithm* (AA) *under the adversarial arrival model. Given a prediction $\hat{\mu}$ with accuracy parameter a, it holds that:*

$$\limsup_{T \to \infty} \sup_{\gamma \in \mathcal{S}^T} \left\{ \frac{1}{T} \left(\mathrm{PRD}(\gamma) - R(\mathrm{AA} \mid \gamma) \right) \right\} \leq 0.$$

Proof of Lemma 18. Recall the updating rule

$$\mu_{t+1} \in \arg\min_{\mu \in \mathbb{R}_+^m} \phi_t^\top \mu + \frac{1}{\eta} V_h\left(\mu, \mu_t\right)$$

where $\phi_t = -g_t(x_t) + \rho$. Note that $\phi_t^\top \mu + \frac{1}{\eta} V_h\left(\mu, \mu_t\right)$ is convex in μ, and set its gradient of μ to zero yields

$$\phi_t + \frac{1}{\eta} (\nabla h(\mu) - \nabla h(\mu_t)) = 0,$$

where $h(\cdot)$ is the reference function. Because

$$||\phi_t||_\infty \leq ||g_t(x_t)||_\infty + ||\rho||_\infty \leq \bar{g} + \bar{\rho}$$

and by our assumption $h(\cdot)$ is σ-strongly convex with respect to the ℓ_1-norm in $\mathbb{R}_+^m$, we have

$$||\mu_{t+1} - \mu_t||_1 \leq \frac{\eta}{\sigma}||\phi_t||_\infty \leq \frac{c(\bar{g} + \bar{\rho})\epsilon(T)}{\sigma T}.$$

Therefore

$$||\mu_t - \hat{\mu}||_1 = ||\mu_t - \mu_1||_1 \leq \sum_{s=2}^{t} ||\mu_s - \mu_{s-1}||_1 \leq \frac{c(\bar{g} + \bar{\rho})\epsilon(T)}{\sigma T} t,$$

and hence

$$\sum_{t=1}^{T} ||\mu_t - \hat{\mu}||_1 \leq \left(\frac{c(\bar{g} + \bar{\rho})}{2\sigma} + 1\right)\epsilon(T). \tag{B.9}$$

Let $x_t^{\hat{\mu}}$ be the actions taken by the Prediction Algorithm at time t, then $\text{PRD}(\gamma) = \sum_{t=1}^{T} r_t(x_t^{\hat{\mu}})$. Because $\zeta > 0$ is a constant and $\epsilon(T) \in o(1)$,

$$||\mu_t - \hat{\mu}||_1 \leq \frac{c(\bar{g} + \bar{\rho})\epsilon(T)}{\sigma T} t \leq \zeta$$

for all t as $T \to \infty$. Therefore $\mu_1, \ldots, \mu_T$ is a sequence of dual variables that satisfies Assumption 2. Let τ_A^j be the depletion time of resources j of Algorithm 2 and τ_P^j be the depletion time of resources j of Algorithm 1. Then by Assumption 2 we have $|\tau_P^j - \tau_A^j| \in o(T)$ for all resource j. Moreover, since there are m resources, outside of all times between each τ_A^j and τ_P^j, T is partitioned into at most $m + 1$ consecutive time blocks, say $B_1, \ldots, B_k$ for some $k \leq m + 1$. Note that the set of feasible actions $\{x \mid x \in \mathcal{X}_t, g_t(x) \leq \text{amount of remaining resources}\}$ at time period t is the same for Algorithm 2 and Algorithm 1 for all $t \in \cup_{k'=1}^{k} B_{k'}$. Therefore both algorithms perform online mirror descent during time periods $B_1, \ldots, B_k$. Therefore similar to Eq. (B.8) we have

$$\sum_{t \in B_{k'}} w_t(\mu_t) \leq \sum_{t \in B_{k'}} w_t(\hat{\mu}) + \frac{c(\bar{g} + \bar{\rho})^2}{2\sigma}\epsilon(T) + \frac{\kappa L \psi(T) T}{c\epsilon(T)} \tag{B.10}$$

for each $B_{k'}$. Also, because $x_t \in \arg\max_{x \in \mathcal{X}_t, g_t(x) \leq G_t} \left\{r_t(x) - \mu_t^\top g_t(x)\right\}$, for $t \in \cup_{k'=1}^{k} B_{k'}$ we have

$$r_t(x_t) - \mu_t^\top g_t(x_t) \geq r_t(x_t^{\hat{\mu}}) - \mu_t^\top g_t(x_t^{\hat{\mu}}). \tag{B.11}$$

Because $w_t(\mu_t) = \mu_t^\top(\rho - g_t(x_t))$ and $w_t(\hat{\mu}) = \hat{\mu}^\top(\rho - g_t(x_t^{\hat{\mu}}))$, for each B'_k we get

$$\begin{aligned}
\sum_{t\in B_{k'}} (r_t(x_t^{\hat{\mu}}) - r_t(x_t)) &\le \sum_{t\in B_{k'}} (\mu_t^\top g_t(x_t^{\hat{\mu}}) - \mu_t^\top g_t(x)) \\
&= \sum_{t\in B_{k'}} \mu_t^\top(\rho - g_t(x)) - \sum_{t\in B_{k'}} \hat{\mu}^\top(\rho - g_t(x_t^{\hat{\mu}})) \\
&\quad + \sum_{t\in B_{k'}} (\hat{\mu} - \mu_t)^\top(\rho - g_t(x_t^{\hat{\mu}})) \\
&\le \sum_{t\in B_{k'}} w_t(\mu_t) - \sum_{t\in B_{k'}} w_t(\hat{\mu}) + \bar{\rho} \sum_{t\in B_{k'}} \|\mu_t - \hat{\mu}\|_1 \\
&\le \frac{c(\bar{g}+\bar{\rho})^2}{2\sigma}\epsilon(T) + \frac{\kappa L\psi(T)T}{c\epsilon(T)} + \left(\frac{c\bar{\rho}(\bar{g}+\bar{\rho})}{2\sigma} + \bar{\rho}\right)\epsilon(T).
\end{aligned}$$

where the first inequality follows from Eq. (B.11), the second inequality is because by Hölder's inequality $(\hat{\mu}-\mu_t)^\top(\rho - g_t(x_t^{\hat{\mu}})) \le ||\hat{\mu}-\mu_t||_1||\rho - g_t(x_t^{\hat{\mu}})||_\infty \le \bar{\rho}||\hat{\mu}-\mu_t||_1$, and the third inequality follows from Eq. (B.10) and Eq. (B.9). Therefore

$$\begin{aligned}
\mathrm{PRD}(\gamma) - R(\mathrm{AA} \mid \gamma) &= \sum_{t\in\cup_{k'=1}^k B_{k'}} \left(r_t(x_t^{\hat{\mu}}) - r_t(x_t)\right) \\
&\quad + \sum_{t\in[T]\setminus\cup_{k'=1}^k B_{k'}} \left(r_t(x_t^{\hat{\mu}}) - r_t(x_t)\right) \\
&\le k\Big(\frac{c(\bar{g}+\bar{\rho})^2}{2\sigma}\epsilon(T) + \frac{\kappa L\psi(T)T}{c\epsilon(T)} \\
&\qquad + \left(\frac{c\bar{\rho}(\bar{g}+\bar{\rho})}{2\sigma} + \bar{\rho}\right)\epsilon(T)\Big) \\
&\quad + \bar{r}\cdot|[T]\setminus\cup_{k'=1}^k B_{k'}| \\
&= o(T).
\end{aligned}$$

where the first inequality is because $r_t(x_t^{\hat{\mu}}) - r_t(x_t) \le \bar{r}$ for each t, and the second inequality is by noting that $k \le m$, $\epsilon(T) = o(T), \psi(T)T/\epsilon(T) = o(T)$, and $|[T]\setminus\cup_{k'=1}^k B_{k'}| \le \sum_{j=1}^m |\tau_P^j - \tau_A^j|$. This shows

$$\limsup_{T\to\infty} \sup_{\gamma\in\mathcal{S}^T} \left\{\frac{1}{T}\left(\mathrm{PRD}(\gamma) - R(\mathrm{AA} \mid \gamma)\right)\right\} \le 0.$$

■

Combine Lemma 17 and Lemma 18 gives Proposition 9. ■

B.6 Proofs in Section 3.4.3

Proof of Theorem 1. We divide the proof into three cases. The first case is that the underlying arrival model is stochastic and the algorithm never switches to the Adversarial Arrival Algorithm (i.e., the "for" loop in the algorithm is never broken), and in this case we show that Regret(MainALG) = $\tilde{O}(\max\{T^{\frac{1}{2}-a}, 1\})$. The second case is that the underlying arrival model is stochastic and yet the algorithm switches to the Adversarial Arrival Algorithm at some point, and we prove that this case happens with low probability. The third case is that the underlying arrival model is adversarial, and in this case we show that

$$\limsup_{T\to\infty} \sup_{\gamma\in\mathcal{S}^T} \left\{\frac{1}{T}(1-\lambda)\left(\max\left\{\frac{1}{\alpha^*}\mathrm{OPT}(\gamma), \mathrm{PRD}(\gamma)\right\} - R(\mathrm{MainALG} \mid \gamma)\right)\right\} \le \delta\bar{r},$$

regardless of whether the algorithm switches to the Adversarial Arrival Algorithm or not. To simplify the notation, throughout the proof we will assume δT and $1/\delta$ are integers. The roundings $\lfloor \delta T \rfloor$ and $\lceil 1/\delta \rceil$ in our algorithm will not affect the result of our analysis.

Case 1: Suppose the underlying arrival model is stochastic where each arrival γ_t is drawn i.i.d. from an underlying probability distribution $\mathcal{P} \in \Delta(\mathcal{S})$, and the algorithm never switches to the Adversarial Arrival Algorithm. Then the algorithm decomposes T time periods into $1/\delta$ time blocks, where each time block contains δT time periods and has at least $\delta T\rho$ amount of resources available. During each time block the algorithm performs the Stochastic Arrival Algorithm. Therefore, by our definition of $\mathrm{OPT}_s(\gamma_1, \ldots, \gamma_s)$ and the performance guarantee of the Stochastic Arrival Algorithm given by Proposition 8, we have

$$\begin{aligned}\mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[\sum_{k=0}^{1/\delta-1} \mathrm{OPT}(\gamma_{k\delta T+1:(k+1)\delta T}) - R(\mathrm{MainALG} \mid \gamma)\right] &= \tilde{O}\left(\frac{1}{\delta}\max\left\{(\delta T)^{\frac{1}{2}-a}, 1\right\}\right) \\ &= \tilde{O}\left(\max\left\{T^{\frac{1}{2}-a}, 1\right\}\right). \end{aligned} \tag{B.12}$$

For each time period t, let $D_t(\mu \mid \gamma_t) := r_t^*(\mu) + \mu^\top \rho$ be the t-term of the Lagrangian dual function Equation (3.3), then every $D_t(\mu \mid \gamma_t)$ is also i.i.d. By Lemma 14 and Lemma 15, for every arrival sequence γ we have

$$\mathrm{OPT}(\gamma_{1:s}) \le \sum_{t=1}^{s} D_t(\mu \mid \gamma_t) \quad \forall \mu \in \mathbb{R}_+^m \tag{B.13}$$

and

$$\min_{\mu\in\mathbb{R}_+^m} \sum_{t=1}^{s} D_t(\mu \mid \gamma_t) \le \mathrm{OPT}(\gamma_{1:s}) + (m+1)\bar{r} \tag{B.14}$$

for every time period s. Setting $s = T$, taking $\mu \in \mathbb{R}_+^m$ to be the minimizer, and taking the expected value of Eq. (B.13) gives

$$\mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[\mathrm{OPT}(\gamma)\right] \le \mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[\min_{\mu\in\mathbb{R}_+^m}\sum_{t=1}^{T} D_t(\mu \mid \gamma_t)\right]. \tag{B.15}$$

Taking expected value on both sides of Eq. (B.14) yields

$$\mathbb{E}_{(\gamma_{1:s})\sim\mathcal{P}^s}\left[\min_{\mu\in\mathbb{R}_+^m}\sum_{t=1}^{s} D_t(\mu \mid \gamma_t)\right] \le \mathbb{E}_{(\gamma_{1:s})\sim\mathcal{P}^s}[\mathrm{OPT}(\gamma_{1:s})] + (m+1)\bar{r}. \tag{B.16}$$

Therefore

$$\mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[\sum_{k=0}^{1/\delta-1}\min_{\mu\in\mathbb{R}_+^m}\sum_{t=k\delta T+1}^{(k+1)\delta T} D_t(\mu \mid \gamma'_t)\right] \le \mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[\sum_{k=0}^{1/\delta-1}\mathrm{OPT}(\gamma_{k\delta T+1:(k+1)\delta T})\right] + (m+1)\bar{r}/\delta. \tag{B.17}$$

Combine Eq. (B.15) and Eq. (B.17) we have

$$\begin{aligned}
\mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[\mathrm{OPT}(\gamma_{1:T})\right] &\le \mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[\min_{\mu\in\mathbb{R}_+^m}\sum_{t=1}^{T} D_t(\mu \mid \gamma_t)\right] \\
&\le \mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[\sum_{k=0}^{1/\delta-1}\min_{\mu\in\mathbb{R}_+^m}\sum_{t=k\delta T+1}^{(k+1)\delta T} D_t(\mu \mid \gamma'_t)\right] \\
&\le \mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[\sum_{k=0}^{1/\delta-1}\mathrm{OPT}(\gamma_{k\delta T+1:(k+1)\delta T})\right] + (m+1)\bar{r}/\delta. \qquad \text{(B.18)}
\end{aligned}$$

We conclude the proof of case 1 by combining Eq. (B.12) and Eq. (B.18)and noting that $(m+1)\bar{r}/\delta$ is a constant:

$$\mathrm{Regret}(\mathrm{MainALG}) = \mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[\mathrm{OPT}(\gamma) - R(\mathrm{MainALG} \mid \gamma)\right] = \tilde{O}(\max\{T^{\frac{1}{2}-a}, 1\}).$$

Case 2: Suppose the underlying arrival model is stochastic where each arrival γ_t is drawn i.i.d. from an underlying probability distribution $\mathcal{P} \in \Delta(\mathcal{S})$. We show that the probability that the algorithm switches to the Adversarial Arrival Algorithm is low. More specifically, we show that this probability is no more than $\frac{3+\delta}{\delta^2 T}$.

First we prove a Chernoff-like bound for sums with stopping times.

Lemma 19 (Stopping Time Chernoff)**.** *Consider a discrete-time random sequence with states $S_1, S_2, \ldots$ where each state S_t determines two values x_t and y_t with*

$x_t, y_t \in [0, c]$ for some constant $c > 0$. Suppse $\mathbb{E}[x_t \mid S_{t-1}] \leq \mathbb{E}[y_t \mid S_{t-1}]$. Then for every $0 < \epsilon < 1$ and every $\mu > 0$ we have

$$\mathbb{P}\left(\exists\, \tau \text{ such that } \sum_{t=1}^{\tau} x_t/(1+\epsilon) - \sum_{t=1}^{\tau} y_t/(1-\epsilon) \geq \epsilon\mu c\right) < \exp(-\epsilon^2\mu).$$

Proof of Lemma 19. Let $\phi_0 = 1$, and for $\tau = 1, 2, \ldots$, let $\phi_\tau = (1+\epsilon)^{\sum_{t=1}^{\tau} x_t/c}(1-\epsilon)^{\sum_{t=1}^{\tau} y_t/c}$. Then $\phi_0, \phi_1, \ldots$ is a non-negative super-martingale. Indeed, for $\tau \geq 1$ we have

$$\begin{aligned}\frac{\phi_\tau}{\phi_{\tau-1}} &= (1+\epsilon)^{x_\tau/c}(1-\epsilon)^{y_\tau/c} \\ &\leq (1+\epsilon x_\tau/c)(1-\epsilon y_\tau/c) \\ &\leq 1 + \epsilon x_\tau/c - \epsilon y_\tau/c.\end{aligned}$$

where the first inequality is because $x_t/c, y_t/c \in [0, 1]$ for every t. Because $\mathbb{E}[x_t \mid S_{t-1}] \leq \mathbb{E}[y_t \mid S_{t-1}]$, we get $\mathbb{E}[\phi_\tau/\phi_{\tau-1} \mid S_{\tau-1}] \leq 1$, which shows $\phi_0, \phi_1, \ldots$ is a non-negative super-martingale.

If the event in the statement happens at some τ, then

$$\exp\left(\sum_{t=1}^{\tau} \epsilon x_t/c(1+\epsilon) - \sum_{t=1}^{\tau} \epsilon y_t/c(1-\epsilon)\right) \geq \exp(\epsilon^2\mu).$$

Using $e^{\epsilon/(1-\epsilon)} < 1 + \epsilon$ we get

$$\phi_\tau = (1+\epsilon)^{\sum_{t=1}^{\tau} x_t/c}(1-\epsilon)^{\sum_{t=1}^{\tau} y_t/c} > \exp(\epsilon^2\mu).$$

Therefore

$$\begin{aligned}\mathbb{P}\Big(\exists\, \tau \text{ such that } \sum_{t=1}^{\tau} \frac{x_t}{1+\epsilon} - \sum_{t=1}^{\tau} \frac{y_t}{1-\epsilon} \geq \epsilon\mu c\Big) &\leq \mathbb{P}\Big(\exists\, \tau \text{ such that } \phi_\tau > \exp(\epsilon^2\mu)\Big) \\ &< \exp(-\epsilon^2\mu).\end{aligned}$$

where the second inequality follows by Doob's martingale inequality. ∎

To analyze the reward obtained so far by the algorithm at a certain time period, we revisit the proof of Proposition 8 and inherit all notations are ed from the proof of Proposition 8. Recall in Eq. (B.3) we have

$$\mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[r_t\left(x_t\right) \mid \sigma\left(\xi_{t-1}\right)\right] = \mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[\bar{D}\left(\mu_t \mid \mathcal{P}\right) - w_t(\mu_t) \mid \sigma\left(\xi_{t-1}\right)\right].$$

For any $x_t \in \mathcal{X}_t$ and $\mu_t \in \mathbb{R}_+^m$ we have $0 \le r_t(x_t), \bar{D}(\mu_t \mid \mathcal{P}) - w_t(\mu_t) \le \bar{r}$. Therefore, for the stopping time τ_A defined in the proof of Proposition 8, we can apply Lemma 19 on $\bar{D}(\mu_t \mid \mathcal{P}) - w_t(\mu_t)$ and $r_t(x_t)$, which gives

$$\begin{aligned}
&\mathbb{P}_{\gamma\sim\mathcal{P}^T}\Big(\sum_{t=1}^{\tau_A}\big(\bar{D}(\mu_t \mid \mathcal{P}) - w_t(\mu_t)\big) - \sum_{t=1}^{\tau_A} r_t(x_t) \\
&\qquad \ge \frac{2\epsilon'\bar{r}T}{1-\epsilon'} + (1+\epsilon')\epsilon'\mu'\bar{r}\Big) \\
&\le \mathbb{P}_{\gamma\sim\mathcal{P}^T}\Big(\sum_{t=1}^{\tau_A}\big(\bar{D}(\mu_t \mid \mathcal{P}) - w_t(\mu_t)\big) \\
&\qquad - \sum_{t=1}^{\tau_A} r_t(x_t)\frac{1+\epsilon'}{1-\epsilon'} \ge (1+\epsilon')\epsilon'\mu'\bar{r}\Big) \\
&\le \mathbb{P}_{\gamma\sim\mathcal{P}^T}\Big(\sum_{t=1}^{\tau_A}\frac{\bar{D}(\mu_t \mid \mathcal{P}) - w_t(\mu_t)}{1+\epsilon'} - \sum_{t=1}^{\tau_A}\frac{r_t(x_t)}{1-\epsilon'} \ge \epsilon'\mu'\bar{r}\Big) \\
&< \exp(-\epsilon'^2\mu').
\end{aligned}$$

where the first inequalities follows because $r_t(x_t) \le \bar{r}$ and $\tau_A \le T$, so

$$2\sum_{t=1}^{\tau_A} r_t(x_t)/(1-\epsilon') \le 2\epsilon'\bar{r}T/(1-\epsilon');$$

the second inequality is obtained by dividing $1+\epsilon'$ on both sides of the inequality; the third inequality utilizes Lemma 19. Plug in $\epsilon' = T^{-1/2}$ and $\mu = T\log(T)$ yields

(B.19)
$$\mathbb{P}_{\gamma\sim\mathcal{P}^T}\left(\sum_{t=1}^{\tau_A}(\bar{D}(\mu_t \mid \mathcal{P}) - w_t(\mu_t)) - \sum_{t=1}^{\tau_A} r_t(x_t) \ge (4\bar{r} + 2\bar{r}\log(T))\sqrt{T}\right) < \frac{1}{T}.$$

We will use Eq. (B.19) later in bounding the concentration of $R(\mathrm{SA} \mid \gamma)$.

Then we look to bound the concentration of $\mathrm{OPT}(\gamma)$. Because $0 \le r_t^*(\mu) \le \bar{r}$ for every $\mu \in \mathbb{R}_+^m$, by Hoeffding's inequality we have

(B.20)
$$\mathbb{P}_{\gamma\sim\mathcal{P}^T}\left(\sum_{t=1}^{T} D_t(\mu \mid \gamma_t) - \mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[\sum_{t=1}^{T} D_t(\mu \mid \gamma_t')\right] > y\right) \le \exp\left(-\frac{2y^2}{\bar{r}^2T}\right)$$

and

(B.21)
$$\mathbb{P}_{\gamma\sim\mathcal{P}^T}\left(\mathbb{E}_{\gamma'\sim\mathcal{P}^T}\left[\sum_{t=1}^{s} D_t(\mu \mid \gamma_t')\right] - \sum_{t=1}^{s} D_t(\mu \mid \gamma_t) > y\right) \le \exp\left(-\frac{2y^2}{\bar{r}^2 s}\right) \quad \forall s$$

for every $\mu \in \mathbb{R}_+^m$ and $y > 0$. Apply Eq. (B.13) and Eq. (B.16) to Eq. (B.20) and take μ to be the minimizer on the left hand side of Eq. (B.16) gives

(B.22)
$$\mathbb{P}_{\gamma\sim\mathcal{P}^T}\left(\mathrm{OPT}(\gamma) - \mathbb{E}_{\gamma\sim\mathcal{P}^T}[\mathrm{OPT}(\gamma)] > y + (m+1)\bar{r}\right) \le \exp\left(-\frac{2y^2}{\bar{r}^2T}\right).$$

Take $y = \sqrt{\bar{r}^2 T \log(T)/2}$ yields

$$\mathbb{P}_{\gamma\sim\mathcal{P}^T}\left(\mathrm{OPT}(\gamma) - \mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[\mathrm{OPT}(\gamma)\right] > \sqrt{\bar{r}^2 T\log(T)/2} + (m+1)\bar{r}\right) \le \frac{1}{T}. \tag{B.23}$$

Recall in the steps of Eq. (B.6), we have $R(\mathrm{SA} \mid \gamma) \ge \sum_{t=1}^{\tau_A} r_t(x_t)$ and $\bar{D}\left(\mu_t \mid \mathcal{P}\right) \ge \tau_A \bar{D}\left(\bar{\mu}_{\tau_A} \mid \mathcal{P}\right)$. Combine Eq. (B.19) and Eq. (B.23) gives that, for every $z > 0$,

$$\begin{aligned}
&\mathbb{P}_{\gamma\sim\mathcal{P}^T}\Big(\mathrm{OPT}(\gamma) - R(\mathrm{SA}\mid\gamma) \\
&\qquad \ge z + (4\bar{r} + 2\bar{r}\log(T))\sqrt{T} + \sqrt{\bar{r}^2 T\log(T)/2} + (m+1)\bar{r}\Big) \\
&\overset{(a)}{\le} \mathbb{P}_{\gamma\sim\mathcal{P}^T}\left(\mathbb{E}_{\gamma\sim\mathcal{P}^T}[\mathrm{OPT}(\gamma)] - R(\mathrm{SA}\mid\gamma) \ge z + (4\bar{r} + 2\bar{r}\log(T))\sqrt{T}\right) + \frac{1}{T} \\
&\overset{(b)}{\le} \mathbb{P}_{\gamma\sim\mathcal{P}^T}\left(\mathbb{E}_{\gamma\sim\mathcal{P}^T}[\mathrm{OPT}(\gamma)] - \sum_{t=1}^{\tau_A} r_t(x_t) \ge z + (4\bar{r} + 2\bar{r}\log(T))\sqrt{T}\right) + \frac{1}{T} \\
&\overset{(c)}{\le} \mathbb{P}_{\gamma\sim\mathcal{P}^T}\left(\mathbb{E}_{\gamma\sim\mathcal{P}^T}[\mathrm{OPT}(\gamma)] - \sum_{t=1}^{\tau_A}(\bar{D}(\mu_t\mid\mathcal{P}) - w_t(\mu_t)) \ge z\right) + \frac{2}{T} \\
&\overset{(d)}{\le} \mathbb{P}_{\gamma\sim\mathcal{P}^T}\left(\mathbb{E}_{\gamma\sim\mathcal{P}^T}[\mathrm{OPT}(\gamma)] - \tau_A\bar{D}(\bar{\mu}_{\tau_A}\mid\mathcal{P}) + \sum_{t=1}^{\tau_A} w_t(\mu_t) \ge z\right) + \frac{2}{T} \\
&\overset{(e)}{\le} \mathbb{P}_{\gamma\sim\mathcal{P}^T}\left((T-\tau_A)\bar{r} + \sum_{t=1}^{\tau_A} w_t(\mu^*) + CT^{\frac{1}{2}-a}\cdot\mathrm{polylog}(T) \ge z\right) + \frac{2}{T} \\
&\overset{(f)}{\le} \mathbb{P}_{\gamma\sim\mathcal{P}^T}\left(\frac{\bar{r}\bar{g}}{\underline{\rho}} + CT^{\frac{1}{2}-a}\cdot\mathrm{polylog}(T) \ge z\right) + \frac{3}{T}.
\end{aligned}$$

Here (a) follows by Eq. (B.23); (b) is because $R(\mathrm{SA}\mid\gamma) \ge \sum_{t=1}^{\tau_A} r_t(x_t)$; (c) follows by Eq. (B.19); (d) is because $\bar{D}\left(\mu_t \mid \mathcal{P}\right) \ge \tau_A\bar{D}\left(\bar{\mu}_{\tau_A}\mid\mathcal{P}\right)$; (e) holds since the last three steps of Eq. (B.6) is deterministic in nature; (f) follows from the last paragraph of the proof of Proposition 8. Take $z = \frac{\bar{r}\bar{g}}{\underline{\rho}}$ and note that

$$z + (4\bar{r} + 2\bar{r}\log(T))\sqrt{T} + \sqrt{\bar{r}^2 T\log(T)/2} + (m+1)\bar{r} \in O(\log(T)\sqrt{T}),$$

i.e., there exists a constant $C' > 0$ such that

$$z + (4\bar{r} + 2\bar{r}\log(T))\sqrt{T} + \sqrt{\bar{r}^2 T\log(T)/2} + (m+1)\bar{r} < C'\log(T)\sqrt{T}.$$

This gives

$$\mathbb{P}_{\gamma\sim\mathcal{P}^T}\left(\mathrm{OPT}(\gamma) - R(\mathrm{SA}\mid\gamma) > C'\log(T)\sqrt{T}\right) \le \frac{3}{T}. \tag{B.24}$$

Suppose the algorithm does not switch to the Adversarial Arrival Algorithm before time period $k'\delta T$ for some $k' \in \{0,\ldots,1/\delta - 1\}$. For $k = 0, 1, \ldots, k'-1$, the

algorithm performs the Stochastic Arrival Algorithm during each time block between time periods $k\delta T + 1$ and $(k+1)\delta T$. Apply Eq. (B.14) to Eq. (B.21) over each time block gives that, for every $y > 0$,

$$\begin{aligned}\mathbb{P}_{\gamma\sim\mathcal{P}^T}\Bigg(\mathbb{E}_{\gamma'\sim\mathcal{P}^T}\left[\sum_{t=k\delta T+1}^{(k+1)\delta T} D_t(\mu \mid \gamma'_t)\right] \\ - \mathrm{OPT}(\gamma_{k\delta T+1:(k+1)\delta T}) > y + (m+1)\bar{r}\Bigg) \le \exp\left(-\frac{2y^2}{\bar{r}^2\delta T}\right).\end{aligned} \tag{B.25}$$

Let X_k be the random variable such that

$$X_k = \mathbb{E}_{\gamma'\sim\mathcal{P}^T}\left[\sum_{t=k\delta T+1}^{(k+1)\delta T} D_t(\mu \mid \gamma'_t)\right] - \mathrm{OPT}(\gamma_{k\delta T+1:(k+1)\delta T}) - (m+1)\bar{r}$$

where $\gamma_{k\delta T+1:(k+1)\delta T} \sim \mathcal{P}^{\delta T}$. Then by Eq. (B.25) each X_k is an independent sub-Gaussian random variable with parameter $\sqrt{2/\bar{r}^2\delta T}$. Therefore $\sum_{k=0}^{k'-1} X_k$ is also a sub-Gaussian random variable with parameter at most $\sqrt{2/\bar{r}^2\delta T}$. Hence we get

$$\begin{aligned}&\mathbb{P}_{\gamma\sim\mathcal{P}^T}\Bigg(\mathbb{E}_{\gamma'\sim\mathcal{P}^T}\left[\sum_{t=1}^{k'\delta T} D_t(\mu \mid \gamma'_t)\right] \\ &\qquad - \sum_{k=0}^{k'-1} \mathrm{OPT}(\gamma_{k\delta T+1:(k+1)\delta T}) > y + (m+1)\bar{r}/\delta\Bigg) \\ &= \mathbb{P}_{\gamma\sim\mathcal{P}^T}\left(\sum_{k=0}^{k'-1} X_k > y\right)\end{aligned} \tag{B.26}$$

$$\le \exp\left(-\frac{2y^2}{\bar{r}^2\delta^3 T}\right). \tag{B.27}$$

Note that

$$\mathbb{E}_{\gamma'\sim\mathcal{P}^T}\left[\sum_{t=1}^{k'\delta T} D_t(\mu \mid \gamma'_t)\right] / k'\delta = \mathbb{E}_{\gamma'\sim\mathcal{P}^T}\left[\sum_{t=1}^{T} D_t(\mu \mid \gamma'_t)\right],$$

so combining Eq. (B.22) from time period $t = 1$ to time period $t = k'\delta T$ and Eq. (B.26) and using union bound we get

$$\begin{aligned}\mathbb{P}_{\gamma\sim\mathcal{P}^T}\Bigg(\mathrm{OPT}(\gamma_{1:k'\delta T}) - \sum_{k=0}^{k'-1} \mathrm{OPT}(\gamma_{k\delta T+1:(k+1)\delta T}) \\ > 2y + (m+1)\bar{r}/\delta\Bigg) \le \exp\left(-\frac{2y^2}{\bar{r}^2\delta^3 T}\right).\end{aligned}$$

Take $y = \sqrt{\bar{r}^2\delta^3 T\log(T)/2}$ yields

$$\mathbb{P}_{\gamma\sim\mathcal{P}^T}\Bigg(\mathrm{OPT}(\gamma_{1:k'\delta T}) - \sum_{k=0}^{k'-1}\mathrm{OPT}(\gamma_{k\delta T+1:(k+1)\delta T}) > \sqrt{2\bar{r}^2\delta^3 T\log(T)} + (m+1)\bar{r}/\delta\Bigg) \le \frac{1}{T}. \tag{B.28}$$

For $k = 0, 1, \ldots, k'-1$, let $R(\mathrm{SA} \mid \gamma_{k\delta T+1:(k+1)\delta T})$ denote the reward obtained by the Stochastic Arrival Algorithm during each time block between time periods $k\delta T + 1$ and $(k+1)\delta T$, then

$$R_{k'\delta T+1} = \sum_{k=0}^{k'-1} R(\mathrm{SA} \mid \gamma_{k\delta T+1:(k+1)\delta T}),$$

where R_t is the total amount of reward obtained between time periods 1 and $t-1$ as defined in the algorithm. Apply Eq. (B.24) on each time block shows that for each k we have

$$\mathbb{P}_{\gamma\sim\mathcal{P}^T}\Big(\mathrm{OPT}(\gamma_{k\delta T+1:(k+1)\delta T}) - R_{\delta T+1}(\mathrm{SA} \mid \gamma_{k\delta T+1}, \ldots, \gamma_{(k+1)\delta T}) > C'\log(T)\sqrt{T}\Big) \le \frac{3}{T}.$$

and therefore

$$\mathbb{P}_{\gamma\sim\mathcal{P}^T}\left(\sum_{k=0}^{k'-1}\mathrm{OPT}(\gamma_{k\delta T+1:(k+1)\delta T}) - R_{k'\delta T+1} > C'\log(T)\sqrt{T}\right) \le \frac{3k'}{T} \le \frac{3}{\delta T}. \tag{B.29}$$

Let L be a constant such that

$$L\log(T)\sqrt{T} > C'\log(T)\sqrt{T} + \sqrt{2\bar{r}^2\delta^3 T\log(T)} + (m+1)\bar{r}/\delta. \tag{B.30}$$

Combine Eq. (B.28) and Eq. (B.29) gives

$$\mathbb{P}_{\gamma\sim\mathcal{P}^T}\left(\mathrm{OPT}(\gamma_{1:k'\delta T}) - R_{k'\delta T+1} > L\log(T)\sqrt{T}\right) \le \frac{1}{T} + \frac{3}{\delta T} = \frac{3+\delta}{\delta T}.$$

Therefore

$$\begin{aligned}
&\mathbb{P}(\text{the algorithm switches to the Adversarial Arrival Algorithm incorrectly}) \\
&= \sum_{k'=0}^{1/\delta-1}\mathbb{P}(\text{the algorithm switches at time period } k'\delta T+1 \text{ incorrectly}) \\
&\le \sum_{k'=0}^{1/\delta-1}\mathbb{P}_{\gamma\sim\mathcal{P}^T}\Big(\mathrm{OPT}(\gamma_{1:k'\delta T}) - R_{k'\delta T+1} > L\log(T)\sqrt{T}\Big) \\
&\le \frac{3+\delta}{\delta^2 T}.
\end{aligned}$$

Case 3: Suppose the underlying arrival model is adversarial and the algorithm switches to the Adversarial Arrival Algorithm at time period $k'\delta T + 1$ for some $k' \in \{0, 1, \ldots, 1/\delta - 1, 1/\delta\}$. Here, to simplify the notation, we set $k' = 1/\delta$ if the algorithm never switches to the Adversarial Arrival Algorithm. For time periods t_1, t_2, let $R(\text{MainALG} \mid \gamma)[t_1, t_2]$ be the amount of rewards that the algorithm obtained between time periods t_1 and t_2.

Because the algorithm does not switch at time period $(k' - 1)\delta T + 1$, we have

$$\begin{aligned}
& R(\text{MainALG} \mid \gamma)[1, (k' - 1)\delta T] + L\log(T)\sqrt{T} \\
\geq\ & \text{OPT}(\gamma_{1:(k'-1)\delta T}) \\
\geq\ & \max\left\{\frac{1}{\alpha^*}\text{OPT}(\gamma_{1:(k'-1)\delta T}), \text{PRD}(\gamma_{1:(k'-1)\delta T})\right\} \\
\geq\ & \max\left\{\frac{1}{\alpha^*}\text{OPT}(\gamma_{1,k'\delta T}), \text{PRD}(\gamma_{1:(k'-1)\delta T})\right\} - \delta\bar{r}T, \qquad \text{(B.31)}
\end{aligned}$$

where the last inequality follows since the total rewards obtained in δT time periods is upper bounded by $\delta\bar{r}T$.

Because the algorithm releases the remaining $\rho(T - k'\delta T)$ amount of resources for the remaining $T - k'\delta T$ time periods and performs the Adversarial Arrival Algorithm, by Theorem 9

(B.32)

$$\begin{aligned}
& \max\left\{\frac{1}{\alpha^*}\text{OPT}(\gamma_{k'\delta T+1,T}), \text{PRD}(\gamma_{k'\delta T+1,T})\right\} - R(\text{MainALG} \mid \gamma)[k'\delta T + 1, T] \\
& \qquad = o(T).
\end{aligned}$$

Combining Eq. (B.31) and Eq. (B.32) gives

$$\begin{aligned}
& R(\text{MainALG} \mid \gamma) + \delta\bar{r}T \\
=\ & R(\text{MainALG} \mid \gamma)[1, (k' - 1)\delta T] + R(\text{MainALG} \mid \gamma)[k'\delta T + 1, T] \\
\geq\ & \max\left\{\frac{1}{\alpha^*}\text{OPT}(\gamma_{1:k'\delta T}), \text{PRD}(\gamma_{1:k'\delta T})\right\} - L\log(T)\sqrt{T} \\
& + \max\left\{\frac{1}{\alpha^*}\text{OPT}(\gamma_{k'\delta T+1:T}), \text{PRD}(\gamma_{k'\delta T+1:T})\right\} - o(T) \\
\geq\ & (1 - \lambda)\max\left\{\frac{1}{\alpha^*}\text{OPT}(\gamma), \text{PRD}(\gamma)\right\} - o(T),
\end{aligned}$$

where the last inequality follows since γ is (λ, δ)-stationary (Definition 4 and

Observation 6). Hence

$$\limsup_{T\to\infty} \sup_{\gamma\in\mathcal{S}^T} \left\{ \frac{1}{T}\Big((1-\lambda)\max\left\{ \frac{1}{\alpha^*}\mathrm{OPT}(\gamma), \mathrm{PRD}(\gamma) \right\} - R(\mathrm{MainALG} \mid \gamma) \Big) \right\} \le \delta\bar{r}.$$

Putting it all together. If the underlying arrival model is stochastic, combining case 1 and case 2 gives

$$\begin{aligned} & \mathrm{Regret}(\mathrm{MainALG}) \\ = \quad & \mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[\mathrm{OPT}(\gamma) - R(\mathrm{MainALG} \mid \gamma) \mid \text{never switches}\right] \mathbb{P}(\text{never switches}) \\ & + \mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[\mathrm{OPT}(\gamma) - R(\mathrm{MainALG} \mid \gamma) \mid \text{switches}\right] \mathbb{P}(\text{switches}). \end{aligned}$$

By case 1, $\mathbb{E}_{\gamma\sim\mathcal{P}^T}\left[\mathrm{OPT}(\gamma) - R(\mathrm{MainALG} \mid \gamma) \mid \text{never switches}\right] \in \tilde{O}(\max\{T^{\frac{1}{2}-a}, 1\})$. By case 2, $\mathbb{P}(\text{switches}) \le \frac{3+\delta}{\delta T}$. Since $\mathrm{OPT}(\gamma) \in O(T)$, we have

$$\mathrm{Regret}(\mathrm{MainALG}) = \tilde{O}(\max\{T^{\frac{1}{2}-a}, 1\}).$$

If the underlying arrival model is adversarial, case 3 shows

$$\limsup_{T\to\infty} \sup_{\gamma\in\mathcal{S}^T} \left\{ \frac{1}{T}\left((1-\lambda)\max\left\{ \frac{1}{\alpha^*}\mathrm{OPT}(\gamma), \mathrm{PRD}(\gamma) \right\} - R(\mathrm{MainALG} \mid \gamma) \right) \right\} \le \delta\bar{r}.$$

This completes the proof. ∎

B.7 Experiment Details

Synthetic Experiment

The detailed setups were the following. There were 25 products, where each product was randomly assigned a unique integer price in the range of $[1, 25]$ and an embedding that lied randomly in $\mathbb{S}^4$. There were 26 types of customers, consisting of 25 customers that each corresponded to exactly one unique product, and one no-customer type, corresponding to no product being selected in that time interval. For each customer type i (apart from the no-customer type), the probability that it would buy product j if recommended was $\mathrm{sigmoid}(e_i^\top \cdot e_j)/10$, where e_i and e_j were the d-dimensional embeddings for products i and j, respectively. For the no-customer type, the probabilities were zero - we could not recommend anything.

One instance contained $T = 1000$ time periods. To build the arrival sequence, we had a function N that maps each time period t to a probability of observing a no-customer

type at that time. If a customer did arrive, we chose its type uniformly at random. The initial inventory level was controlled by ρ. Modeling inventory shortages or excess inventory can be done by changing ρ. The price of each product was fixed at the start of the experiment and is held constant. To generate predictions, we first calculated at the true item counts in the demand sequence for each product. Depending on the arrival model, we applied various amounts of zero-mean Gaussian noise with variance σ to each of the true item counts. We then took these counts, compared them to our inventory, and determined the predicted shadow prices for each product. By changing σ, we were able to simulate predictions of different qualities.

The synthetic experiment was run on a MacBook Pro equipped with Apple's M2 Chip. The total compute time was under 20 hours. All offline optimization problems in the algorithms were solved by Gurobi.

We list all the (hyper)parameters used:

- Low inventory level: $\rho = .015$, medium inventory level: $\rho = .03$, and high inventory level: $\rho = .06$;
- Root finding bisection parameters: $\alpha = 10^6$, $\beta = 0$, $lo = 10^{-4}$, $hi = 1$;
- Perfect predictions: $\sigma = 0$, good predictions: $\sigma = 5$, and bad predictions $\sigma = 500$;
- Stochastic arrivals: $N(t) = 0.7$, nonstationary arrivals: $N(t) = .4 + \frac{3t}{5000}$, and adversarial arrivals: $N(t) = \mathbb{1}(t > 300)$;
- Parameters for the Main Algorithm (Algorithm 3): $\delta = \frac{1}{20}$ and $L = 7$.

H&M Experiment

The H&M dataset contains two years of online purchase data from H&M customers, consisting of dates, purchase prices, customer IDs, and product ID. For each product, there are basic categorical information about its type, appearance, and department. For computational reasons, we only considered the 5000 most purchased products during this experiment. Our goal was to simulate 90 days of the online marketplace where when a customer selects a product, we recommend three other products in return. Encoding the days using a start day s, we started by building a sequence of customer/no-customer arrivals for the 90 day window: Let $R := \max_{0 \le j \le 89}\{\text{Amount of customers in day } s + j\}$. We initialized an empty array of size $R \cdot 90$. For a day $s + j$, for every product that was purchased in that day, we

randomly placed this product in the array between indices jR and $(j+1)R-1$. We call this sequence of customer/no-customer interactions our demand sequence. Note that each entry in the tuple contained the product and the price for which it was purchased.

A product's price on a given day was set to be the price of that product purchased by some customer on a given day. To ensure that this process was deterministic, as there could be multiple customers purchasing the same product for different prices, we defined the product's price on that day to be the first time that product was purchased by a customer on that day. If no customer purchased that product, we made the assumption that the product was unavailable and took this under consideration when recommending products during the experiment, as we could not recommend a product that is not available. In order to facilitate this experiment, we built an accurate model which took in two products, along with their prices, and determined the probability that those two products were bought together. We did this using sklearn's Random Forest model. First, we created a 50-dimensional embedding for each product. This was done by creating a matrix where each (i, j) entry represented that the i-th product was bought by the j-th customer. Using a matrix factorization collaborative filtering algorithm, we were able to obtain a 50-dimensional embedding for each product. Next, for each product, we created a one-hot vector for "product_group_name", "graphical_appearance_no", "perceived_colour_value_id", "perceived_colour_master_id", "index_code", "index_group_no", and "garment_group_no", and concatenated these one-hot vectors to form a vector of length 102 that contains exactly 7 ones. Given two products, p_1 and p_2, we created the final 207-dimensional vector we fed into the Random Forest model by concatenating p_1 and p_2's one-hot vectors, adding in the dot product similarity metric between the p_1 and p_2's embeddings, and finally adding the prices for both items on that specific day. To train this model, we generated 100,000 positive instances, meaning a customer bought products p_1 and p_2 together on the same day, and 1,000,000 negative instances, where we randomly selected a product p_1 purchased by customer u and find a product p_2 that was available on that day but not bought by u. The trained model had an AUC of 0.78. For any two products, p_1 and p_2, that also contained correct price information for that day, we referred to this probability function as $f_{prob}(p_1, p_2)$, giving us the probability that items p_1 and p_2 were bought together on that specific day.

In executing the Main Algorithms, we modeled the random nature of recommending products to customers, that is, we did not know whether or not a customer would

select the products we recommended. To remedy this, we performed the following procedure to closely model real-world customer decision making. At each time step, we either saw a no-customer, which we would recommend no products, or we observed a product that the customer selected, say $p_{customer}$. Then the value of recommending some product p_{rec} was given by $f_{prob}(p_{customer}, p_{rec}) \cdot (r_t(p_{rec}) - \mu_{rec} g_t(p_{rec}))$, where $r_t(p_{rec})$ represented the current price of the recommended product, $g_t(p_{rec})$ was treated as being 1, since the customer would only consume one unit of the recommended product, and μ_{rec} was the current shadow price of the recommended item as predicted by the dual variable at time period t. We then recommended the top three products according to this above metric that also satisfied the inventory constraint. Note that if no three products existed to recommend, then we recommended no products. Once we recommended three products to the customer, the customer would pick each product with probability $f_{prob}(p_{customer}, p_{rec})$ and we in turn received the product's value along with the decrease in inventory only if the customer ended up buying the product. The customer could select anywhere from none to all of the recommended products, and the selections were assumed to be independent of each other.

To generate the prediction for each instance, we used 365 days of data before the starting day of our testing window. For every 5 day span, we added up all the products that were purchased within that interval. We took this and converted it into counts for the embedding vectors. This gave 50 streams of 73 data points each (one stream per embedding dimension and one data point for each of the $365/5$ combined points). From here, we ran our prediction algorithm (FB Prophet, ARIMA, and Exponential Smoothing) to generate 18 more data points (as 5×18 gives the full 90 days) for each of the 50 streams, converted these data points back into counts for the products themselves, and determined the shadow price for each product using these predicted demands for each product. This was done by solving the offline problem where the inventory levels are given by the predicted demand. The vector of shadow prices became our prediction.

Here, the prediction $\hat{\mu}$ is constructed from historical data that reflect such persistent demand patterns. Even though individual arrivals are modeled as stochastic, the realized arrival sequence is influenced by common underlying factors that induce correlation between past observations and future arrivals. As a result, for an arrival sequence $\gamma \sim \mathcal{P}^T$, the prediction $\hat{\mu}(\gamma)$ and the optimal dual variable $\mu^*(\gamma)$ are not independent, and the prediction error may scale sub-linearly in T, consistent with

our theoretical assumptions. More broadly, our results highlight that the stochastic model relevant for prediction-based decision making is often best viewed as one with *structured randomness*: while arrivals may be independent across periods conditional on latent parameters, predictions can exploit partial information about these parameters, leading to meaningful correlation between $\hat{\mu}$ and the realized arrival sequence.

When running the Main Algorithm (Algorithm 3), we performed sequential hypothesis testing to determine whether or not the arrival sequence was stochastic or not. We began by assuming the arrival sequence was stochastic. Then we performed the following offline hypothesis test: after allowing for a burn-in period of 20 days, for every $t \in \{25, 30, \ldots, 85\}$ we performed a one-sided one sample t-test on the number of arrivals in $[t-4, t]$ compared to the average number of arrivals in $[0, t-5]$. For sufficiently low p-value, chosen to be .05, the algorithm switched to be adversarial. Additionally, due to the large amounts of data used, using the bisection algorithm as written in the Stochastic Arrival Algorithm (Algorithm 18) was too computationally inefficient, so instead we used an approximation of this by selecting a set of η's, H, and computing

$$\eta_t := \arg\min_{\eta \in H} \left| \eta - \frac{\theta_t(\mu_1, \eta)}{\sqrt{\alpha \Phi_t(\mu_1, \eta) + \beta}} \right|$$

This method allowed quicker computation, as we were only running a constant with respect to T versions of the Mirror Descent Algorithm for each instance. The larger our set H was, the closer we would get to the solution outputted by the root finding bisection algorithm.

We list all the (hyper)parameters used:

- Prophet and Exponential Smoothing: default;
- ARIMA parameters: $p = 5$, $q = 2$, $d = 1$;
- Random Forest classifier: n_estimators = 100, max_depth = 18;
- $H = \{10^{-10(1-i/30)} \mid i \in [30]\}$;
- Stochastic Arrival Algorithm parameters: $\alpha = 1$, $\beta = 0$.

Appendix C

APPENDIX FOR CHAPTER 4

C.1 Preliminary Observations and Proofs

Proof of Lemma 2. By Assumption 4, we have

$$\begin{aligned} C(\mu_1, q_2^*) - C(\mu_1, q_1^*) &= C(\mu_1, q_2^*) - C(\mu_2, q_1^*) + C(\mu_2, q_1^*) - C(\mu_1, q_1^*) \\ &\overset{(a)}{\leq} C(\mu_1, q_2^*) - C(\mu_2, q_1^*) + \ell|\mu_1 - \mu_2| \\ &\overset{(b)}{\leq} C(\mu_1, q_2^*) - C(\mu_2, q_2^*) + \ell|\mu_1 - \mu_2| \\ &\overset{(c)}{\leq} 2\ell|\mu_1 - \mu_2|. \end{aligned}$$

Here (a) and (c) follow from $C(\mu, q)$ being ℓ-Lipchitz in μ, and (b) uses the definition of q_2^*. ■

Proof of Observation 3a). By Lemma 2,

$$C_t(\mu_t, q_t) - C_t(\mu_t, q_t^*) \leq 2\ell|\hat{\mu}_t - \mu_t|,$$

where ℓ is the Lipschitz constant of $C(\cdot, q)$. Therefore,

$$\begin{aligned} \mathcal{R}^{\pi^{\text{prediction}}}(T) &= \sup_{\boldsymbol{D} \in \mathcal{D}(v)} \mathbb{E}_{\boldsymbol{D}}^{\pi^{\text{prediction}}} \left\{ \sum_{t=1}^{T} \left(C_t(\mu_t, q_t) - C_t(\mu_t, q_t^*) \right) \right\} \\ &\leq 2\ell \cdot \sup_{\boldsymbol{D} \in \mathcal{D}(v)} \mathbb{E}_{\boldsymbol{D}}^{\pi^{\text{prediction}}} \left\{ \sum_{t=1}^{T} |\hat{\mu}_t - \mu_t| \right\}. \end{aligned}$$

Finally, by the construction of $\hat{\mu}_t$,

$$\sum_{t=1}^{T} |\hat{\mu}_t - \mu_t| \leq \sum_{t=1}^{T} |a_t - \mu_t| \leq T^a.$$

Taking $C = 2\ell$ gives the desired result. ■

For any time periods a, b, let

$$\mathcal{R}^{\pi}(T)[a, b] = \sup_{\boldsymbol{D} \in \mathcal{D}(v)} \mathbb{E}_{\boldsymbol{D}}^{\pi} \left\{ \sum_{t=a}^{b} \left(C_t(\mu_t, q_t) - C_t(\mu_t, q_t^*) \right) \right\}$$

be the regret incurred by the policy π from time a to time b. We make the following two useful observations:

Observation 7. *For any policy π, $\mathcal{R}^\pi(T)[a,b] \le C(b-a+1)$ for some universal constant $C \in (0,\infty)$.*

Proof of Observation 7. Because μ_t and q_t are both bounded, $C_t(\mu_t, q_t)$ is also bounded in $[C_{\min}, C_{\max}]$ where

$$C_{\min} = \inf_{\substack{\mu_t \in [\mu_{\min},\mu_{\max}], q_t \in Q \\ b_t \in [0,b_{\max}], h_t \in [0,h_{\max}]}} C_t(\mu_t, q_t)$$

and

$$C_{\max} = \sup_{\substack{\mu_t \in [\mu_{\min},\mu_{\max}], q_t \in Q \\ b_t \in [0,b_{\max}], h_t \in [0,h_{\max}]}} C_t(\mu_t, q_t).$$

Thus,

$$\mathcal{R}^\pi(T)[a,b] = \sup_{\boldsymbol{D} \in \mathcal{D}(v)} \mathbb{E}^\pi_{\boldsymbol{D}} \left\{ \sum_{t=a}^{b} \left(C_t(\mu_t, q_t) - C_t(\mu_t, q_t^*) \right) \right\} \le (C_{\max} - C_{\min})(b-a+1).$$

Take $C = C_{\max} - C_{\min}$ gives the desired result. As a remark, note that by footnote 3 we have $C_{\max} \le 3\max\{b_{\max}, h_{\max}\}(\delta + Q_{\max})$, so C can be taken as $3\max\{b_{\max}, h_{\max}\}(\delta + Q_{\max})$. ∎

Observation 8. *For any policy π such that, from time a to time b, π first estimates the mean at time t to be $\hat{\mu}_t \in [\mu_{\min}, \mu_{\max}]$ for every $a \le t \le b$ and then order $q_t \in \arg\min_{q \in Q} C_t(\hat{\mu}_t, q)$, we have $\mathcal{R}^\pi(T)[a,b] \le C \cdot \sup_{\boldsymbol{D} \in \mathcal{D}(v)} \mathbb{E}^\pi_{\boldsymbol{D}} \left\{ \sum_{t=a}^{b} |\hat{\mu}_t - \mu_t| \right\}$ for some universal constant $C \in (0,\infty)$.*

Proof of Observation 8. Let ℓ be the Lipschitz constant of $C(\cdot, q)$, then by Lemma 2

$$C_t(\mu_t, q_t) - C_t(\mu_t, q_t^*) \le 2\ell |\hat{\mu}_t - \mu_t|.$$

Therefore,

$$\begin{aligned} \mathcal{R}^\pi(T)[a,b] &= \sup_{\boldsymbol{D} \in \mathcal{D}(v)} \mathbb{E}^\pi_{\boldsymbol{D}} \left\{ \sum_{t=a}^{b} \left(C_t(\mu_t, q_t) - C_t(\mu_t, q_t^*) \right) \right\} \\ &\le 2\ell \cdot \sup_{\boldsymbol{D} \in \mathcal{D}(v)} \mathbb{E}^\pi_{\boldsymbol{D}} \left\{ \sum_{t=a}^{b} |\hat{\mu}_t - \mu_t| \right\}. \end{aligned}$$

Taking $C = 2\ell$ gives the desired result. ∎

Observation 8 implies that any policy π that estimates the mean accurately at each time achieves low regret.

Finally, we will make use of standard sub-Gaussian concentration:

Lemma 20 (Hoeffding's Inequality, e.g. [169]). *Let $\epsilon_1, \ldots, \epsilon_n$ be independent, mean-zero, sub-Gaussian variables with sub-Gaussian norm at most K:*

$$\mathbb{P}\left(|\epsilon_i| > t\right) \leq 2\exp\left(-\frac{t^2}{K^2}\right) \quad \textit{for all } t \geq 0.$$

Then for some universal constant C,

$$\mathbb{P}\left(\frac{1}{n}\left|\sum_{i=1}^{n}\epsilon_i\right| > t\right) \leq 2\exp\left(-\frac{nt^2}{CK^2}\right) \quad \textit{for all } t \geq 0.$$

Moreover, C is upper bounded by $144e$.

C.2 Proof of Lemma 3

Proof of Lemma 3. Because our bounds are all asymptotic, we ignore the rounding and write $n = \kappa T^{(1-v)/2}$ to simplify the notation.

By Observation 7, $\mathcal{R}^{\pi^{\text{fixed}}}(T)[1,n] \leq C_1 n < C_1 T^{(3+v)/4}$ for some universal constant $C_1 = 3\max\{b_{\max}, h_{\max}\}(\delta + Q_{\max}) \in (0,\infty)$.

From now on we consider the time period from $t = n+1$ to $t = T$. We first upper bound the total estimation error of the mean $\sum_{t=n+1}^{T}|\hat{\mu}_t - \mu_t|$. Note that

$$\begin{aligned}
\sum_{t=n+1}^{T}|\hat{\mu}_t - \mu_t| &\overset{(a)}{\leq} \sum_{t=n+1}^{T}\left|\frac{1}{n}\sum_{s=t-n}^{t-1}d_s - \mu_t\right| \\
&= \sum_{t=1}^{T}\left|\frac{1}{n}\sum_{s=t-n}^{t-1}(\mu_s + \epsilon_s) - \mu_t\right| \\
&= \sum_{t=n+1}^{T}\left|\frac{1}{n}\sum_{s=t-n}^{t-1}(\mu_s - \mu_t) + \frac{1}{n}\sum_{s=t-n}^{t-1}\epsilon_s\right| \\
&\leq \sum_{t=n+1}^{T}\left|\frac{1}{n}\sum_{s=t-n}^{t-1}(\mu_s - \mu_t)\right| + \sum_{t=n+1}^{T}\left|\frac{1}{n}\sum_{s=t-n}^{t-1}\epsilon_s\right|,
\end{aligned}$$

where (a) follows because $\hat{\mu}_t$ is the projection of $\frac{1}{n}\sum_{s=t-n}^{t-1}d_s$ on $[\mu_{\min}, \mu_{\max}]$.

We bound these two parts separately through the following two lemmas.

Lemma 21. *There exists a universal constant $C_2 \in (0,\infty)$ such that*

$$\sum_{t=n+1}^{T}\left|\frac{1}{n}\sum_{s=t-n}^{t-1}(\mu_s - \mu_t)\right| \leq C_2 T^{(3+v)/4}.$$

Proof of Lemma 21. For any $n+1 \le t \le T$ we have

$$\left|\frac{1}{n}\sum_{s=t-n}^{t-1}(\mu_s-\mu_t)\right| \le \frac{1}{n}\sum_{s=t-n}^{t-1}|\mu_s-\mu_t| \le \max_{t-n\le s\le t-1}|\mu_s-\mu_t|\,.$$

Also, by bounded demand variation,

$$\begin{aligned}
\sum_{t=n+1}^{T}\max_{t-n\le s\le t-1}|\mu_s-\mu_t|^2 &\overset{(a)}{=} \sum_{j=1}^{\lceil T/n\rceil}\sum_{i=1}^{n}\max_{(j-1)n+i\le s\le jn+i-1}|\mu_s-\mu_t|^2\\
&\overset{(b)}{=} \sum_{i=1}^{n}\sum_{j=1}^{\lceil T/n\rceil}\max_{(j-1)n+i\le s\le jn+i-1}|\mu_s-\mu_t|^2\\
&\overset{(c)}{\le} nV_{\boldsymbol{\mu}}\\
&\le \kappa T^{(1+v)/2},
\end{aligned}$$

where (a) is obtained by partitioning the sum into time windows, (b) is obtained by exchanging the summations, and (c) follows by the definition of demand variation $V_{\boldsymbol{\mu}}$ since $\{t_j = jn+i-1 : j = 0,1,\dots,\lceil T/n\rceil\}$ is a partition of $\{1,\dots,T\}$ for all $i = 1,\dots,n$.

By Cauchy–Schwarz inequality,

$$\begin{aligned}
\sum_{t=n+1}^{T}\max_{t-n\le s\le t-1}|\mu_s-\mu_t| &\le \sqrt{(T-n)\cdot\sum_{t=n+1}^{T}\max_{t-n\le s\le t-1}|\mu_s-\mu_t|^2}\\
&\le \sqrt{(T-n)\cdot\kappa T^{(1+v)/2}}\\
&\le \sqrt{\kappa}T^{(3+v)/4}.
\end{aligned}$$

Take $C_2 = \sqrt{\kappa}$ gives the desired result. ∎

Lemma 22. *There exists a universal constant $C_3 \in (0,\infty)$ such that*

$$\mathbb{E}_{\boldsymbol{D}}^{\pi^{\text{fixed}}}\left\{\sum_{t=n+1}^{T}\left|\frac{1}{n}\sum_{s=t-n}^{t-1}\epsilon_s\right|\right\} \le C_3 T^{(3+v)/4}.$$

Proof of Lemma 22. Because each D_s is sub-Gaussian, each ϵ_s is sub-Gaussian. Therefore there exists a constant $\delta_s < \infty$ where $\delta_s = \inf\{\delta' \ge 0 : \mathbb{E}[e^{\epsilon_s^2/\delta'^2}] \le 2\}$. Let $\delta = \max_{s=1,\dots,T}\delta_s$, then by Hoeffding's inequality, for any $n+1 \le t \le T$ we have

$$\mathbb{P}\left\{\left|\frac{1}{n}\sum_{s=t-n}^{t-1}\epsilon_s\right| \ge x\right\} \le 2\exp\left(-\frac{\rho(nx)^2}{\sum_{s=t-n}^{t-1}\delta_s^2}\right) \le 2\exp\left(-\frac{\rho n}{\delta^2}x^2\right),$$

where $\rho > 0$ is a universal constant. Therefore we have

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{D}}^{\pi^{\text{fixed}}}\left\{\left|\frac{1}{n}\sum_{s=t-n}^{t-1}\epsilon_s\right|\right\} &= \int_0^\infty \mathbb{P}\left\{\left|\frac{1}{n}\sum_{s=t-n}^{t-1}\epsilon_s\right| \geq x\right\} dx \\
&\leq \int_0^\infty 2\exp\left(-\frac{\rho n}{\delta^2}x^2\right) dx \\
&= \sqrt{\frac{\pi\delta^2}{\rho n}} \\
&= \sqrt{\frac{\pi\delta^2}{\rho\kappa}} T^{(v-1)/4}.
\end{aligned}$$

Hence

$$\mathbb{E}_{\boldsymbol{D}}^{\pi^{\text{fixed}}}\left\{\sum_{t=n+1}^{T}\left|\frac{1}{n}\sum_{s=t-n}^{t-1}\epsilon_s\right|\right\} \leq (T-n)\sqrt{\frac{\pi\delta^2}{\rho\kappa}}T^{(v-1)/4} \leq \sqrt{\frac{\pi\delta^2}{\rho\kappa}}T^{(3+v)/4}.$$

Take $C_3 = \sqrt{\frac{\pi\delta^2}{\rho\kappa}}$ gives the desired result. Note by Lemma 20 $1/\rho \leq 144e$, so $C_3 \leq \frac{\delta}{12}\sqrt{\frac{\pi}{e\kappa}}$. ■

Now we finish the proof of Lemma 3. With Lemma 21 and Lemma 22, we conclude that

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{D}}^{\pi^{\text{fixed}}}\left\{\sum_{t=n+1}^{T}|\hat{\mu}_t - \mu_t|\right\} &\leq \mathbb{E}_{\boldsymbol{D}}^{\pi^{\text{fixed}}}\left\{\sum_{t=n+1}^{T}\left|\frac{1}{n}\sum_{s=t-n}^{t-1}(\mu_s - \mu_t)\right|\right\} \\
&\quad + \mathbb{E}_{\boldsymbol{D}}^{\pi^{\text{fixed}}}\left\{\sum_{t=n+1}^{T}\left|\frac{1}{n}\sum_{s=t-n}^{t-1}\epsilon_s\right|\right\} \\
&\leq (C_2 + C_3)T^{(3+v)/4}.
\end{aligned}$$

Therefore by Observation 8 there exists a universal constant $C_4 = 2\ell \in (0, \infty)$ such that

$$\begin{aligned}
\mathcal{R}^{\pi^{\text{fixed}}}(T) &= \mathcal{R}^{\pi^{\text{fixed}}}(T)[1, n] + \mathcal{R}^{\pi^{\text{fixed}}}(T)[n+1, T] \\
&\leq C_1 T^{(3+v)/4} + C_4(C_2 + C_3)T^{(3+v)/4} \\
&\leq (C_1 + C_4(C_2 + C_3))T^{(3+v)/4}.
\end{aligned}$$

Take $C = (C_1 + C_4(C_2 + C_3))$ we get $\mathcal{R}^{\pi^{\text{fixed}}}(T) \leq CT^{(3+v)/4}$. ■

C.3 Proof of Theorem 2

Proof of Theorem 2. Because our bounds are all asymptotic, we ignore the roundings and write $n_i = \kappa T^{(1-v_i)/2}$ to simplify the notation. By Observation 7 $\mathcal{R}^{\pi^{\text{shrinking}}}(T)[1, T^{3/4}] \le C_1 T^{3/4}$ for some universal constant

$$C_1 = 3\max\{b_{\max}, h_{\max}\}(\delta + Q_{\max}) \in (0, \infty).$$

From now on we consider the time periods after $T^{3/4}$. Let ℓ be the smallest index such that $v_\ell \ge v$, then $v_\ell \le (1 + \frac{1}{\log T})v$, so

$$T^{v_\ell} \le T^{(1+1/\log T)v} = e^v T^v \le eT^v. \tag{C.1}$$

First, we show that $\hat{\mu}_t^j$ is close to μ_t when $j \ge \ell$ via the following lemma:

Lemma 23. *For every $j \ge \ell$ with the corresponding window size $n_j = \kappa T^{(1-v_j)/2}$, there exists a universal constant $\gamma > 0$ such that*

$$\mathbb{P}\left\{\sum_{t=n_j+1}^{T} \left|\hat{\mu}_t^j - \mu_t\right| \ge \left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) \cdot T^{(3+v_j)/4}\right\} \le \frac{2}{T^{3/2}}.$$

Proof of Lemma 23. Same as in the proof of Lemma 22, by Hoeffding's inequality for any $n_j + 1 \le t \le T$ we have

$$\mathbb{P}\left\{\left|\frac{1}{n_j}\sum_{s=t-n_j}^{t-1} \epsilon_s\right| \ge x\right\} \le 2\exp\left(-\frac{\rho(n_j x)^2}{\sum_{s=t-n_j}^{t-1}\delta_s^2}\right) \le 2\exp\left(-\frac{\rho n_j}{\delta^2}x^2\right),$$

where ρ and δ are the same as in the previous proof. Set $\gamma > 0$ to be large enough so that

$$\frac{\rho\kappa\gamma^2}{\delta^2} \ge \frac{5}{2}. \tag{C.2}$$

Take $x = \gamma\sqrt{\log T} \cdot T^{(v_j-1)/4}$ and plug in $n_j = \kappa T^{(1-v_j)/2}$ yields

$$\mathbb{P}\left\{\left|\frac{1}{n_j}\sum_{s=t-n_j}^{t-1} \epsilon_s\right| \ge \gamma\sqrt{\log T} \cdot T^{(v_j-1)/4}\right\} \le 2\exp\left(-\frac{\rho\kappa\gamma^2}{\delta^2}\log T\right) \le \frac{2}{T^{5/2}}.$$

Then we get

$$\begin{aligned}
&\mathbb{P}\left\{ \sum_{t=n_j+1}^{T} \left| \frac{1}{n_j} \sum_{s=t-n_j}^{t-1} \epsilon_s \right| \geq \gamma \sqrt{\log T}\, T^{(3+v_j)/4} \right\} \\
&\qquad \leq \mathbb{P}\left\{ \max_{n_j+1 \leq t \leq T} \left| \frac{1}{n_j} \sum_{s=t-n_j}^{t-1} \epsilon_s \right| \right. \\
&\qquad\qquad \left. \geq \gamma \sqrt{\log T}\, T^{(v_j-1)/4} \right\} \\
&\qquad \overset{(a)}{\leq} \sum_{t=n_j+1}^{T} \mathbb{P}\left\{ \left| \frac{1}{n_j} \sum_{s=t-n_j}^{t-1} \epsilon_s \right| \geq \gamma \sqrt{\log T}\, T^{(v_j-1)/4} \right\} \\
&\qquad \leq \frac{2}{T^{3/2}}.
\end{aligned}$$

where (a) follows by union bound. Note that since $V_{\boldsymbol{\mu}} \leq T^{v} \leq T^{v_j}$, by Lemma 21 we know $\sum_{t=n_j+1}^{T} \left| \frac{1}{n_j} \sum_{s=t-n_j}^{t-1} (\mu_s - \mu_t) \right| \leq \sqrt{\kappa} T^{(3+v_j)/4}$, and in the proof of Lemma 3 we have

$$\sum_{t=n_j+1}^{T} |\hat{\mu}_t^j - \mu_t| \leq \sum_{t=n_j+1}^{T} \left| \frac{1}{n_j} \sum_{s=t-n_j}^{t-1} (\mu_s - \mu_t) \right| + \sum_{t=n_j+1}^{T} \left| \frac{1}{n_j} \sum_{s=t-n_j}^{t-1} \epsilon_s \right|.$$

Therefore

$$\begin{aligned}
&\mathbb{P}\left\{ \sum_{t=n_j+1}^{T} \left| \hat{\mu}_t^j - \mu_t \right| \geq \left(\gamma \sqrt{\log T} + \sqrt{\kappa}\right) T^{(3+v_j)/4} \right\} \\
&\leq \mathbb{P}\left\{ \sum_{t=n_j+1}^{T} \left| \frac{1}{n_j} \sum_{s=t-n_j}^{t-1} (\mu_s - \mu_t) \right| + \sum_{t=n_j+1}^{T} \left| \frac{1}{n_j} \sum_{s=t-n_j}^{t-1} \epsilon_s \right| \right. \\
&\qquad \left. \geq \left(\gamma \sqrt{\log T} + \sqrt{\kappa}\right) T^{(3+v_j)/4} \right\} \\
&\leq \mathbb{P}\left\{ \sum_{t=n_j+1}^{T} \left| \frac{1}{n_j} \sum_{s=t-n_j}^{t-1} \epsilon_s \right| \geq \gamma \sqrt{\log T}\, T^{(3+v_j)/4} \right\} \\
&\leq \frac{2}{T^{3/2}}.
\end{aligned}$$

■

For each $j \geq \ell$, let E_j be the event $\left\{ \sum_{t=n_j+1}^{T} \left| \hat{\mu}_t^j - \mu_t \right| \geq \left(\gamma \sqrt{\log T} + \sqrt{\kappa}\right) \cdot T^{(3+v_j)/4} \right\}$, then $\mathbb{P}(E_j) \leq \frac{2}{T^{3/2}}$ for each j. First we assume that E_j does not happen for any $j \geq \ell$. We break the proof into three lemmas.

Lemma 24. *Each time an* ***if*** *condition happens, for the current i we have $i < \ell$. Therefore $i \le \ell$ throughout the algorithm.*

Proof of Lemma 24. Suppose an **if** condition happens at time t triggered by some index $j > i$, then $\sum_{s=t_{\mathbf{if}}}^{t} |\hat{\mu}_s^i - \hat{\mu}_s^j| \ge 2\left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) T^{(3+v_j)/4}$. Suppose for the sake of contradiction that $i \ge \ell$, then

$$\begin{aligned}
\sum_{s=t_{\mathbf{if}}}^{t} |\hat{\mu}_s^i - \hat{\mu}_s^j| &\overset{(a)}{\le} \sum_{s=t_{\mathbf{if}}}^{t} |\hat{\mu}_s^i - \mu_s| + \sum_{s=t_{\mathbf{if}}}^{t} |\hat{\mu}_s^j - \mu_s| \\
&\overset{(b)}{\le} \sum_{s=n_i+1}^{T} |\hat{\mu}_s^i - \mu_s| + \sum_{s=n_j+1}^{T} |\hat{\mu}_s^j - \mu_s| \\
&\overset{(c)}{<} \left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) \cdot T^{(3+v_i)/4} + \left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) \cdot T^{(3+v_j)/4} \\
&\overset{(d)}{<} 2\left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) \cdot T^{(3+v_j)/4},
\end{aligned}$$

where (a) comes from the triangle inequality and (b) is because $t_{\mathbf{if}} > T^{3/4}$ and $n_i, n_j \le \kappa T^{1/2}$; since $j > i \ge \ell$, by our assumption neither E_i nor E_j occurs, so we get (c); (d) follows since $v_j > v_i$. This contradicts with $\sum_{s=t_{\mathbf{if}}}^{t} |\hat{\mu}_s^i - \hat{\mu}_s^j| \ge 2\left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) T^{(3+v_j)/4}$. Therefore, we must have $i < \ell$. Because the index i never decreases and can only increase by 1 each time an **if** condition happens, $i \le \ell$ throughout the algorithm. ■

Lemma 25. *Suppose two consecutive* ***if*** *conditions occur at time t' and t'', then $\mathcal{R}^{\pi^{\text{shrinking}}}(T)[t', t''-1] \le C_2 T^{(3+v)/4}\sqrt{\log T}$ for some universal constant $C_2 \in (0, \infty)$.*

Proof of Lemma 25. At time t where $t' \le t \le t'' - 1$, by Lemma 24 $i < \ell$. We have

$$\begin{aligned}
\sum_{s=t'+1}^{t''-1} |\hat{\mu}_s^i - \mu_s| &\overset{(a)}{\le} \sum_{s=t'+1}^{t''-1} |\hat{\mu}_s^\ell - \mu_s| + \sum_{s=t'+1}^{t''-1} |\hat{\mu}_s^i - \hat{\mu}_s^\ell| \\
&\overset{(b)}{\le} \sum_{s=n_i+1}^{T} |\hat{\mu}_s^\ell - \mu_s| + \sum_{s=t'+1}^{t''-1} |\hat{\mu}_s^i - \hat{\mu}_s^\ell| \\
&\overset{(c)}{<} \left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) \cdot T^{(3+v_\ell)/4} + 2\left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) \cdot T^{(3+v_\ell)/4} \\
&\overset{(d)}{\le} 3e^{1/4}\left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) \cdot T^{(3+v)/4},
\end{aligned}$$

where (a) comes from the triangle inequality and (b) is because $t' > T^{3/4}$ and $n_i \le \kappa T^{1/2}$; the first part of (c) follows by our assumption that E_ℓ does not occur, and the second part of (c) follows since the **if** condition is not triggered between

time t' and time $t''-1$; (d) follows by Equation (C.1). Then by Observation 8 there exists a universal constant $C'=2\ell \in (0,\infty)$ such that

$$\mathcal{R}^{\pi^{\text{shrinking}}}(T)[t', t''-1] \le C' \sum_{s=t'}^{t''-1} |\hat{\mu}_s^i - \mu_s| \le 3e^{1/4}C' \left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) \cdot T^{(3+v)/4}.$$

Take $C_2 = 3e^{1/4}C' \left(\gamma + \sqrt{\kappa}\right)$ gives the desired result. ■

The above proof also works for $\mathcal{R}^{\pi^{\text{shrinking}}}(T)[T^{3/4}+1, t_{\text{first}}-1]$ if the first **if** condition happens at time t_{first}, or $\mathcal{R}^{\pi^{\text{shrinking}}}(T)[T^{3/4}+1, T]$ if the **if** condition never happens. Note that the index i never decreases and increases by 1 if and only if the **if** condition happens. Suppose that the last **if** condition happens at time t_{last}, then we have

$$\begin{aligned}
\mathcal{R}^{\pi^{\text{shrinking}}}(T)[T^{3/4}+1, t_{\text{last}}-1] &\overset{(a)}{\le} (\#\text{ of } \textbf{if} \text{ conditions happened}) \\
&\quad \cdot C_2 T^{(3+v)/4}\sqrt{\log T} \\
&\overset{(b)}{\le} k C_2 T^{(3+v)/4}\sqrt{\log T} \\
&\overset{(c)}{\approx} C_2 \log_{1+\frac{1}{\log T}}(T)\, T^{(3+v)/4}\sqrt{\log T} \\
&= C_2 \frac{\log T}{\log(1+\frac{1}{\log T})} T^{(3+v)/4}\sqrt{\log T} \\
&\overset{(d)}{\approx} C_2 T^{(3+v)/4} \log^{5/2} T.
\end{aligned}$$

for some universal constant $C_2 \in (0,\infty)$. Here (a) follows by Lemma 25, (b) is because the maximum index of v_i is k, (c) is because $v_{k-1} < 1 \le v_k$, and (d) follows by $\log(1+x) \approx x$ when x is small.

Finally, we analyze the time periods after t_{last}.

Lemma 26. *$\mathcal{R}^{\pi^{\text{shrinking}}}(T)[t_{\text{last}}, T] \le C_3 T^{(3+v)/4}\sqrt{\log T}$ for some universal constant $C_3 \in (0,\infty)$.*

Proof of Lemma 26. Because the **if** condition happens at t_{last}, by Lemma 24 either

$i < \ell$ or $i = \ell$. Suppose $i < \ell$, then similar to Lemma 25 we have

$$\begin{aligned}
\sum_{s=t_{\text{last}}}^{T} |\hat{\mu}_s^i - \mu_s| &\overset{(a)}{\leq} \sum_{s=t_{\text{last}}}^{T} |\hat{\mu}_s^\ell - \mu_s| + \sum_{s=t_{\text{last}}}^{T} |\hat{\mu}_s^i - \hat{\mu}_s^\ell| \\
&\overset{(b)}{\leq} \sum_{s=n_i+1}^{T} |\hat{\mu}_s^\ell - \mu_s| + \sum_{s=t_{\text{last}}}^{T} |\hat{\mu}_s^i - \hat{\mu}_s^\ell| \\
&\overset{(c)}{<} \left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) \cdot T^{(3+v_\ell)/4} + 2\left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) \cdot T^{(3+v_\ell)/4} \\
&\overset{(d)}{\leq} 3e^{1/4}\left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) \cdot T^{(3+v)/4},
\end{aligned}$$

where (a) comes from the triangle inequality and (b) is because $t_{\text{last}} > T^{3/4}$ and $n_i \leq \kappa T^{1/2}$; the first part of (c) follows by our assumption that E_ℓ does not occur, and the second part of (c) follows since the **if** condition is never triggered after t_{last}; (d) follows by Equation (C.1). By Observation 8, we get

$$\mathcal{R}^{\pi^{\text{shrinking}}}(T)[t_{\text{last}}, T] \leq 3e^{1/4}C'\left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right) \cdot T^{(3+v)/4}$$

for some universal constant $C' = 2\ell \in (0, \infty)$.

On the other hand, suppose $i = \ell$. Note $V_{\boldsymbol{\mu}} \leq T^v \leq T^{v_\ell}$ and after time t_{last} the Shrinking-Time-Window Policy just performs the Fixed-Time-Window Policy with variation parameter v_ℓ, so by Lemma 3

$$\mathcal{R}^{\pi^{\text{shrinking}}}(T)[t_{\text{last}}, T] \leq C''T^{(3+v_\ell)/4} \leq C''e^{1/4}T^{(3+v)/4}$$

for some universal constant $C'' \in (0, \infty)$, where the last inequality again follows by Equation (C.1).

Taking $C_3 = 3e^{1/4}\max\{C'\left(\gamma + \sqrt{\kappa}\right), C''\}$ we get

$$\mathcal{R}^{\pi^{\text{shrinking}}}(T)[t_{\text{last}}, T] \leq C_3T^{(3+v)/4}\sqrt{\log T}.$$

■

Combining everything above, in the case where E_j doesn't happen for any $j \geq \ell$, we get

$$\begin{aligned}
\mathcal{R}^{\pi^{\text{shrinking}}}(T) &= \mathcal{R}^{\pi^{\text{shrinking}}}(T)[1, T^{3/4}] + \mathcal{R}^{\pi^{\text{shrinking}}}(T)[T^{3/4} + 1, t_{\text{last}} - 1] \\
&\quad + \mathcal{R}^{\pi^{\text{shrinking}}}(T)[t_{\text{last}}, T] \\
&\leq C_1T^{3/4} + C_2T^{(3+v)/4}\log^{5/2}T + C_3T^{(3+v)/4}\sqrt{\log T}.
\end{aligned}$$

Now let us consider the case where E_j happens for some $j \geq \ell$. Because $\mathbb{P}\{E_j\} \leq \frac{2}{T^{3/2}}$ for each $j \geq \ell$, by union bound

$$\mathbb{P}\left\{E_j \text{ happens for some } j \geq \ell\right\} \leq \frac{2}{T^{3/2}} \cdot |\mathcal{V}| \leq \frac{2}{T},$$

where the last inequality follows by $|\mathcal{V}| = k \approx \log^2 T$. By Observation 7 we always have $\mathcal{R}^{\pi^{\text{shrinking}}}(T) \leq C_4 T$ for some universal constant $C_4 \in (0, \infty)$. Therefore in summary we have

$$\begin{aligned}
\mathcal{R}^{\pi^{\text{shrinking}}}(T) &\leq \mathbb{P}\{E_j \text{ doesn't happen for any } j \geq \ell\} \\
&\quad \cdot \left(C_1 T^{3/4} + C_2 T^{(3+v)/4} \log^{5/2} T + C_3 T^{(3+v)/4} \sqrt{\log T}\right) \\
&\quad + \mathbb{P}\{E_j \text{ happens for some } j \geq \ell\}\, C_4 T \\
&\leq C_1 T^{3/4} + C_2 T^{(3+v)/4} \log^{5/2} T + C_3 T^{(3+v)/4} \sqrt{\log T} + 2C_4.
\end{aligned}$$

Therefore, there exists a constant $C \in (0, \infty)$ such that

$$\mathcal{R}^{\pi^{\text{shrinking}}}(T) \leq C T^{(3+v)/4} \log^{5/2} T.$$

■

C.4 Proof of Proposition 10, Proposition 11, and Observation 3b)

Note that Proposition 10 and Observation 3b) can be easily deduced from Proposition 11 by setting $a = 1$ and $v = 1$ respectively, so it suffices to prove Proposition 11.

Proof of Proposition 11. We construct the following worst-case problem instance: divide the time horizon T into cycles of length $T^{(1-v)/2}$ (since the analysis below is compatible with scaling, for simplicity we assume $T^{(1-v)/2}$ is an integer), so there are $T^{(1+v)/2}$ cycles. Assume that the demand distribution within each cycle is a Bernoulli distribution that equals to 1 with probability p and equals to 0 with probability $1 - p$. At the beginning of each cycle, we set the p of the upcoming cycle to be either $\frac{1}{2} + \frac{1}{\sqrt{20}} T^{(v-1)/4}$ or $\frac{1}{2} - \frac{1}{\sqrt{20}} T^{(v-1)/4}$, each with probability $\frac{1}{2}$. Set $Q = \mathbb{R}_+$ and $b_t = h_t = 1$ for all t, so the optimal ordering amount is the median of the demand distribution at each time.

First we show that the demand variation $V_{\boldsymbol{\mu}}$ is at most T^v. Note that since $\mu_t = p$ is fixed within each cycle, only the times between cycles contribute to the demand variation. Suppose the cycle changes between time t and $t + 1$, then

$$(\mu_{t+1} - \mu_t)^2 \leq \left(\frac{2}{\sqrt{20}} T^{(v-1)/4}\right)^2 = \frac{1}{5} T^{(v-1)/2}.$$

Because there are $T^{(1+v)/2}$ cycles,

$$V_{\mu} \le T^{(1+v)/2} \cdot \frac{1}{5} T^{(v-1)/2} = \frac{1}{5} T^{v}.$$

Then we add predictions into the instance. We divide the analysis into two cases.

Case 1: $a \ge \frac{3+v}{4}$. For each t we set a_t to be either $\frac{1}{2} + \frac{1}{\sqrt{20}} T^{(v-1)/4}$ or $\frac{1}{2} - \frac{1}{\sqrt{20}} T^{(v-1)/4}$, each with probability $\frac{1}{2}$. Then because each a_t and μ_t are i.i.d., a_t provides no information about μ_t. Hence the predictions are useless in this instance. Note that

$$\sum_{t=1}^{T} |a_t - \mu_t| \le T \cdot \frac{2}{\sqrt{20}} T^{(v-1)/4} = \frac{1}{\sqrt{5}} T^{(3+v)/4} \le \frac{1}{\sqrt{5}} T^{a},$$

so the prediction accuracy is within T^a.

Then we analyze the amount of regret incurred. For $i = 1, \ldots, T^{(1-v)/2}$, let P_i, Q_i be i.i.d. distributions respectively, where $P_i \sim$ Bernoulli $(\frac{1}{2} + \frac{1}{\sqrt{20}} T^{(v-1)/4})$ and $Q_i \sim$ Bernoulli $(\frac{1}{2} - \frac{1}{\sqrt{20}} T^{(v-1)/4})$. Then the Kullback-Leibler divergence of P_i from Q_i is

$$\begin{aligned} D_{\mathrm{KL}}(P_i \parallel Q_i) = &\left(\frac{1}{2} + \frac{1}{\sqrt{20}} T^{(v-1)/4}\right) \log\left(\frac{\frac{1}{2} + \frac{1}{\sqrt{20}} T^{(v-1)/4}}{\frac{1}{2} - \frac{1}{\sqrt{20}} T^{(v-1)/4}}\right) \\ &+ \left(\frac{1}{2} - \frac{1}{\sqrt{20}} T^{(v-1)/4}\right) \log\left(\frac{\frac{1}{2} - \frac{1}{\sqrt{20}} T^{(v-1)/4}}{\frac{1}{2} + \frac{1}{\sqrt{20}} T^{(v-1)/4}}\right). \end{aligned}$$

We show that $D_{\mathrm{KL}}(P_i \parallel Q_i) \le \frac{13}{20} T^{(v-1)/2}$. Let $x = \frac{1}{\sqrt{20}} T^{(v-1)/4}$, then because $v \in [0, 1]$, $x \in [0, \frac{1}{\sqrt{20}}]$. Note that the $D_{\mathrm{KL}}(P_i \parallel Q_i) = 13x^2$ for $x = 0$, and we have

$$\frac{d}{dx} D_{\mathrm{KL}}(P_i \parallel Q_i) \Big|_{x=0} = 0 = \frac{d}{dx} 13x^2 \Big|_{x=0}$$

and

$$\frac{d^2}{dx^2} D_{\mathrm{KL}}(P_i \parallel Q_i) = \frac{1}{(x^2 - 0.25)^2} < 26 = \frac{d^2}{dx^2} 13x^2$$

for $x \in [0, \frac{1}{\sqrt{20}}]$. This shows $13x^2 - D_{\mathrm{KL}}(P_i \parallel Q_i) = 0$ at $x = 0$, the first derivative is 0 at $x = 0$, and the second derivative is non-negative for $x \in [0, \frac{1}{\sqrt{20}}]$. This implies $D_{\mathrm{KL}}(P_i \parallel Q_i) \le 13x^2 = \frac{13}{20} T^{(v-1)/2}$.

Let $P = \sum_{i=1}^{T^{(1-v)/2}} P_i$ and $Q = \sum_{i=1}^{T^{(1-v)/2}} Q_i$ be the two possible demand distributions within a cycle, then because P_i's are i.i.d. and Q_i's are i.i.d.,

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{i=1}^{T^{(1-v)/2}} D_{\mathrm{KL}}(P_i \parallel Q_i) \le T^{(1-v)/2} \cdot \frac{13}{20} T^{(v-1)/2} = \frac{13}{20}.$$

We claim that within each cycle, any attempt to distinguish between P and Q has at least a constant probability of making a mistake, i.e., one cannot effectively estimate the p value within each cycle. Let $C : \{0,1\}^{T^{(1-v)/2}} \to \{P, Q\}$ be any classifier that takes the demand observations within a cycle as inputs and determine the true demand distribution of this cycle. Let $E = C^{-1}(P)$ be the event where C classifies the demand observations as from demand distribution P, then by Pinsker's inequality,

$$|P(E) - Q(E)| \leq \sqrt{\frac{1}{2} D_{\mathrm{KL}}\left(P \parallel Q\right)} \leq \sqrt{\frac{13}{40}},$$

where $P(E)$ is the probability of E happening under the condition that the true demand distribution is P, and the same for $Q(E)$. Therefore we have

$$\mathbb{P}\left\{C \text{ makes a mistake}\right\} = P(E^c) + Q(E) \geq P(E^c) + P(E) - \sqrt{\frac{13}{40}} = 1 - \sqrt{\frac{13}{40}}.$$

Hence for any classifier C the probability of making the wrong guess of p in the current cycle is at least $1 - \sqrt{\frac{13}{40}}$.

Note for demand distribution P_i, $q^* = 1$ and

$$\begin{aligned} C(\mu_{P_i}, 0) - C(\mu_{P_i}, 1) &= \frac{1}{2} + \frac{1}{\sqrt{20}} T^{(v-1)/4} - \left(1 - \left(\frac{1}{2} + \frac{1}{\sqrt{20}} T^{(v-1)/4}\right)\right) \\ &= \frac{1}{\sqrt{5}} T^{(v-1)/4}. \end{aligned}$$

and similarly for demand distribution Q_i, $q^* = 0$ and

$$\begin{aligned} C(\mu_{Q_i}, 1) - C(\mu_{Q_i}, 0) &= \left(1 - \left(\frac{1}{2} - \frac{1}{\sqrt{20}} T^{(v-1)/4}\right)\right) - \left(\frac{1}{2} - \frac{1}{\sqrt{20}} T^{(v-1)/4}\right) \\ &= \frac{1}{\sqrt{5}} T^{(v-1)/4}. \end{aligned}$$

Therefore each wrong guess of p incurs a difference between $C(\mu_t, q_t)$ and $C(\mu_t, q_t^*)$ by $\frac{1}{\sqrt{5}} T^{(v-1)/4}$, which incurs a regret of $C(\mu_t, q_t) - C(\mu_t, q_t^*) = \frac{1}{\sqrt{5}} T^{(v-1)/4}$. Therefore over T time periods the expected total regret of any General Policy π satisfies

$$\begin{aligned} \mathcal{R}^\pi(T) &\geq \left(1 - \sqrt{\frac{13}{40}}\right) \frac{1}{\sqrt{5}} T^{(v-1)/4} \cdot T \\ &= \left(1 - \sqrt{\frac{13}{40}}\right) \frac{1}{\sqrt{5}} T^{(3+v)/4} \\ &= \left(1 - \sqrt{\frac{13}{40}}\right) \frac{1}{\sqrt{5}} T^{\min\{(3+v)/4, a\}}, \end{aligned}$$

where the last equality follows from $a \geq \frac{3+v}{4}$. Take $c = \left(1 - \sqrt{\frac{13}{40}}\right) \frac{1}{\sqrt{5}}$ gives the desired result.

Case 2: $a < \frac{3+v}{4}$**.** For the first $T^{a-(1+3v)/4}$ time periods of each cycle, we set a_t to be either $\frac{1}{2} + \frac{1}{\sqrt{20}}T^{(v-1)/4}$ or $\frac{1}{2} - \frac{1}{\sqrt{20}}T^{(v-1)/4}$, each with probability $\frac{1}{2}$. Note that $a < \frac{3+v}{4}$ implies $a - \frac{1+3v}{4} < \frac{1-v}{2}$, so time periods of length $T^{a-(1+3v)/4}$ are indeed contained in each cycle, which has length $T^{(1-v)/2}$. For the other time periods we set $a_t = \mu_t$. Then, similar as in the case above, the predictions are useless for the first $T^{a-(1+3v)/4}$ time periods of each cycle in this instance. Since there are $T^{(1+v)/2}$ number of cycles,

$$\sum_{t=1}^{T} |a_t - \mu_t| \leq T^{(1+v)/2} \cdot T^{a-(1+3v)/4} \cdot \frac{2}{\sqrt{20}} T^{(v-1)/4} = \frac{1}{\sqrt{5}} T^a,$$

so the prediction accuracy is within T^a.

Again, the same analysis as the case above shows that for the first $T^{a-(1+3v)/4}$ time periods of each cycle, the probability of making the wrong guess of p in the current cycle is at least $1 - \sqrt{\frac{13}{40}}$. Also, each wrong guess of p incurs a regret of $\frac{1}{\sqrt{5}}T^{(v-1)/4}$. Because there are $T^{(1+v)/2}$ number of cycles, over T time periods the expected total regret of any General Policy π satisfies

$$\begin{aligned}
\mathcal{R}^{\pi}(T) &\geq \left(1 - \sqrt{\frac{13}{40}}\right) \frac{1}{\sqrt{5}} T^{(v-1)/4} \cdot T^{(1+v)/2} \cdot T^{a-\frac{1+3v}{4}} \\
&= \left(1 - \sqrt{\frac{13}{40}}\right) \frac{1}{\sqrt{5}} T^{a} \\
&= \left(1 - \sqrt{\frac{13}{40}}\right) \frac{1}{\sqrt{5}} T^{\min\{(3+v)/4, a\}},
\end{aligned}$$

where the last equality follows from $a < \frac{3+v}{4}$. Take $c = \left(1 - \sqrt{\frac{13}{40}}\right) \frac{1}{\sqrt{5}}$ gives the desired result. ∎

C.5 General Variation

Recall that for any sequence of means $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_T\}$, we define the demand variation to be its *quadratic variation*

$$V_{\boldsymbol{\mu}} = \max_{\{t_0, \ldots, t_K\} \in \mathcal{P}} \left\{ \sum_{k=1}^{K} \left| \mu_{t_k} - \mu_{t_{k-1}} \right|^2 \right\},$$

where $\mathcal{P}$ is the set of all partitions. The choice of quadratic variation follows from previous literature [95, 96]. We can also define the demand variation using what we will call θ*-variation* for some $0 \leq \theta < \infty$:

$$V_{\boldsymbol{\mu}} = \max_{\{t_0, \ldots, t_K\} \in \mathcal{P}} \left\{ \sum_{k=1}^{K} \left| \mu_{t_k} - \mu_{t_{k-1}} \right|^{\theta} \right\}.$$

In the special case of $\theta = 1$, this is the *total variation*, which has also been used extensively in previous literature [37, 52, 92, 122]. We prove the following lower bound, which generalizes Proposition 10.

Proposition 20 (Lower Bound: Nonstationary Newsvendor with θ-variation)**.** *Suppose the demand variation is defined using θ-variation and $V_{\mu} \le T^{v}$. For any variation parameter $v \in [0, 1]$, and any policy π (which may depend on the knowledge of v), we have*

$$\mathcal{R}^{\pi}(T) \ge cT^{(1+\theta+v)/(2+\theta)},$$

where $c > 0$ is a universal constant.

Proof of Proposition 20. Similar to the proof of Proposition 10, we construct the following worst-case problem instance: divide the time horizon T into cycles of length $T^{(2-2v)/(2+\theta)}$ (since the analysis below is compatible with scaling, for simplicity we assume $T^{(2-2v)/(2+\theta)}$ is an integer), so there are $T^{(\theta+2v)/(2+\theta)}$ cycles. Assume that the demand distribution within each cycle is a Bernoulli distribution that equals to 1 with probability p and equals to 0 with probability $1 - p$. At the beginning of each cycle, we set the p of the upcoming cycle to be either $\frac{1}{2} + \frac{1}{\sqrt{20}}T^{(v-1)/(2+\theta)}$ or $\frac{1}{2} - \frac{1}{\sqrt{20}}T^{(v-1)/(2+\theta)}$, each with probability $\frac{1}{2}$. Set $Q = \mathbb{R}_+$ and $b_t = h_t = 1$ for all t, so the optimal ordering amount is the median of the demand distribution at each time.

First we show that the demand variation V_{μ} is at most T^{v}. Note that since $\mu_t = p$ is fixed within each cycle, only the times between cycles contribute to the demand variation. Suppose the cycle changes between time t and $t + 1$, then

$$(\mu_{t+1} - \mu_t)^{\theta} \le \left(\frac{2}{\sqrt{20}}T^{(v-1)/(2+\theta)}\right)^{\theta} \le T^{\theta(v-1)/(2+\theta)}.$$

Because there are $T^{(\theta+2v)/(2+\theta)}$ cycles, $V_{\mu} \le T^{(\theta+2v)/(2+\theta)} \cdot T^{\theta(v-1)/(2+\theta)} = T^{v}$.

Then, following the calculations in the proof of Proposition 11, the probability of making the wrong guess of p in the current cycle is at least $1 - \sqrt{\frac{13}{40}}$. Also, each wrong guess of p incurs a regret of $\frac{1}{\sqrt{5}}T^{(v-1)/(2+\theta)}$ at each time period. Therefore over T time periods the expected total regret of any General Policy π satisfies

$$\mathcal{R}^{\pi}(T) \ge \left(1 - \sqrt{\frac{13}{40}}\right)\frac{1}{\sqrt{5}}T^{(v-1)/(2+\theta)} \cdot T = \left(1 - \sqrt{\frac{13}{40}}\right)\frac{1}{\sqrt{5}}T^{(1+\theta+v)/(2+\theta)}.$$

Take $c = \left(1 - \sqrt{\frac{13}{40}}\right)\frac{1}{\sqrt{5}}$ gives the desired result. ∎

C.6 Proof of Proposition 12

Proof of Proposition 12. Set $Q = \mathbb{R}_+$ and $b_t = h_t = 1$ for all t, so the optimal ordering amount is the median of the demand distribution at each time. We construct the following two problem instances:

Instance 1: for all t, set $D_t^{(1)} \sim Unif(0, 2)$. Then $\mu_t^{(1)} = 1$. Set $a_t^{(1)} = d_t^{(1)}$, where $d_t^{(1)}$ is the realization of $D_t^{(1)}$. Since $\mu_t^{(1)} = 1$ for all t, the variation parameter $v^{(1)} = 0$. Because $a_t^{(1)}$ is a constant away from $\mu_t^{(1)}$ with probability 1, $a^{(1)} = 1$. Because the optimal order amount is $q_t^{(1)*} = \mu_t^{(1)}$,

$$C(\mu_t^{(1)}, q_t^{(1)*}) = \mathbb{E}\left[b_t(D_t^{(1)} - \mu_t^{(1)})^+ + h_t(\mu_t^{(1)} - D_t^{(1)})^+\right] = \mathbb{E}\left[|D_t^{(1)} - 1|\right] = \frac{1}{2}.$$

Instance 2: for all t, set $D_t^{(2)} = \mu_t^{(2)} \sim Unif(0, 2)$. i.e., $D_t^{(2)}$ is a one-point distribution. Set $a_t^{(2)} = d_t^{(2)}$, where $d_t^{(2)}$ is the realization of $D_t^{(2)}$. Since $\mu_t^{(2)}$ changes by a constant amount from $\mu_{t-1}^{(2)}$ with probability 1 throughout $t = 2, \ldots, T$, the variation parameter $v^{(2)} = 1$. Because $a_t^{(2)} = \mu_t^{(2)}$ for every t, $a^{(2)} = 0$. Because the optimal order amount is $q_t^{(2)*} = \mu_t^{(2)}$,

$$C(\mu_t^{(2)}, q_t^{(2)*}) = \mathbb{E}\left[b_t(D_t^{(2)} - \mu_t^{(2)})^+ + h_t(\mu_t^{(2)} - D_t^{(2)})^+\right] = 0.$$

Note that a General Policy can only observe information on a_t's and d_t's. Because $a_t^{(1)} = d_t^{(1)}$ and $a_t^{(2)} = d_t^{(2)}$ have the same distribution for every t, no General Policies can distinguish between the two instances. For any General Policy π, let q_t be its output at time t. Without loss of generality, we may assume $q_t \in [0, 2]$ since any other ordering amount is clearly sub-optimal. Then

$$C(\mu_t^{(1)}, q_t) = \mathbb{E}\left[b_t(D_t^{(1)} - q_t)^+ + h_t(q_t - D_t^{(1)})^+\right] = \frac{q_t^2 + (2 - q_t)^2}{4} \geq \frac{1}{2}$$

and

$$C(\mu_t^{(2)}, q_t) = \mathbb{E}\left[b_t(D_t^{(2)} - q_t)^+ + h_t(q_t - D_t^{(2)})^+\right] = \frac{2 - q_t}{2} + \frac{q_t}{2} = 1.$$

Let $\mathcal{R}^\pi[I_1](T)$ denote the regret of π on instance 1 and $\mathcal{R}^\pi[I_2](T)$ denote the regret

of π on instance 2, then

$$\begin{aligned}\mathcal{R}^{\pi}[I_1](T) + \mathcal{R}^{\pi}[I_2](T) &= \sum_{t=1}^{T}\left(C(\mu_t^{(1)}, q_t) - C(\mu_t^{(1)}, q_t^{(1)*})\right) \\ &\quad + \sum_{t=1}^{T}\left(C(\mu_t^{(2)}, q_t) - C(\mu_t^{(2)}, q_t^{(2)*})\right) \\ &\geq \sum_{t=1}^{T}\left(\frac{1}{2} - \frac{1}{2}\right) + \sum_{t=1}^{T}(1-0) \\ &= T.\end{aligned}$$

Because $\mathcal{R}^{\pi}[I_1](T) + \mathcal{R}^{\pi}[I_2](T) \geq T$, any General Policy π incurs regret at least $0.5T$ on at least one of the instances. Since no General Policy can distinguish between the two instances, we can always choose the worse one of the two instances to feed to the policy. Also, since $v^{(1)} = 0$, $a^{(1)} = 1$, $v^{(2)} = 1$, $a^{(2)} = 0$, in both instances we have $a \neq \frac{3+v}{4}$. Therefore for any General Policy π there always exists a problem instance with $a \neq \frac{3+v}{4}$ such that $\mathcal{R}^{\pi}(T) \geq 0.5T = \Omega(T^{\max\{(3+v)/4, a\}})$ on the instance. ■

C.7 Proof of Theorem 3

Proof of Theorem 3. Because our bounds are all asymptotic, we ignore the rounding and write $n = \kappa T^{(1-v)/2}$ to simplify the notation. We slightly abuse the notation and reuse C_1, C_2 as constants that differ from the statement of Theorem 3.

Because $\pi_t^{\text{PERP}} = \pi_t^{\text{prediction}}$ for t from 1 to n, by Observation 3

$$\mathcal{R}^{\pi^{\text{PERP}}}(T)[1,n] = \mathcal{R}^{\pi^{\text{prediction}}}(T)[1,n] \leq C_1' T^a$$

for some universal constant $C_1' \in (0, \infty)$. Also, by Observation 7, since

$$n = \kappa T^{(1-v)/2} < \kappa T^{(3+v)/4},$$

there exists some universal constant $C_1'' \in (0, \infty)$ such that

$$\mathcal{R}^{\pi^{\text{PERP}}}(T)[1,n] \leq C_1'' T^{(3+v)/4}.$$

Take $C_1 = \max\{C_1', C_1''\}$ we get

$$\mathcal{R}^{\pi^{\text{PERP}}}(T)[1,n] \leq C_1 T^{\min\{(3+v)/4, a\}}\sqrt{\log T}.$$

Now we consider the time periods after time $n+1$. First, same as in the proof of Lemma 22, by Hoeffding's inequality for any $n+1 \leq t \leq T$ we have

$$\mathbb{P}\left\{\left|\frac{1}{n}\sum_{s=t-n}^{t-1}\epsilon_s\right| \geq x\right\} \leq 2\exp\left(-\frac{\rho(nx)^2}{\sum_{s=t-n}^{t-1}\delta_s^2}\right) \leq 2\exp\left(-\frac{\rho n}{\delta^2}x^2\right),$$

where ρ and δ are the same as in the previous proof. Set $\gamma > 0$ to be large enough so that

$$\frac{\rho\kappa\gamma^2}{\delta^2} \geq 2. \tag{C.3}$$

Take $x = \gamma\sqrt{\log T} \cdot T^{(v-1)/4}$ and plug in $n = \kappa T^{(1-v)/2}$ yields

$$\mathbb{P}\left\{\left|\frac{1}{n}\sum_{s=t-n}^{t-1}\epsilon_s\right| \geq \gamma\sqrt{\log T}\cdot T^{(v-1)/4}\right\} \leq 2\exp\left(-\frac{\rho\kappa\gamma^2}{\delta^2}\log T\right) \leq \frac{2}{T^2}.$$

Then we get

$$\begin{aligned}
&\mathbb{P}\left\{\sum_{t=n+1}^{T}\left|\frac{1}{n}\sum_{s=t-n}^{t-1}\epsilon_s\right| \geq \gamma\sqrt{\log T}\,T^{(3+v)/4}\right\} \\
&\qquad\leq \mathbb{P}\left\{\max_{n+1\leq t\leq T}\left|\frac{1}{n}\sum_{s=t-n}^{t-1}\epsilon_s\right| \right. \\
&\qquad\qquad \left. \geq \gamma\sqrt{\log T}\,T^{(v-1)/4}\right\} \\
&\qquad\overset{(a)}{\leq} T\cdot\mathbb{P}\left\{\left|\frac{1}{n}\sum_{s=t-n}^{t-1}\epsilon_s\right| \geq \gamma\sqrt{\log T}\,T^{(v-1)/4}\right\} \\
&\qquad\leq \frac{2}{T}.
\end{aligned}$$

where (a) follows by union bound. Note Lemma 21 says

$$\sum_{t=n+1}^{T}\left|\frac{1}{n}\sum_{s=t-n}^{t-1}(\mu_s - \mu_t)\right| \leq \sqrt{\kappa}T^{(3+v)/4},$$

and in the proof of Lemma 3 we have

$$\sum_{t=n+1}^{T}\left|\hat{\mu}_t^{\text{fixed}} - \mu_t\right| \leq \sum_{t=n+1}^{T}\left|\frac{1}{n}\sum_{s=t-n}^{t-1}(\mu_s-\mu_t)\right| + \sum_{t=n+1}^{T}\left|\frac{1}{n}\sum_{s=t-n}^{t-1}\epsilon_s\right|.$$

Therefore

$$\begin{aligned}
&\mathbb{P}\left\{\sum_{t=n+1}^{T}\left|\hat{\mu}_t^{\text{fixed}} - \mu_t\right| \geq \left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right)\cdot T^{(3+v)/4}\right\} \\
&\leq \mathbb{P}\left\{\sum_{t=n+1}^{T}\left|\frac{1}{n}\sum_{s=t-n}^{t-1}(\mu_s-\mu_t)\right| + \sum_{t=n+1}^{T}\left|\frac{1}{n}\sum_{s=t-n}^{t-1}\epsilon_s\right| \geq \left(\gamma\sqrt{\log T} + \sqrt{\kappa}\right)\cdot T^{(3+v)/4}\right\} \\
&\leq \mathbb{P}\left\{\sum_{t=n+1}^{T}\left|\frac{1}{n}\sum_{s=t-n}^{t-1}\epsilon_s\right| \geq \gamma\sqrt{\log T}\cdot T^{(3+v)/4}\right\} \leq \frac{2}{T}.
\end{aligned}$$

Because once the **if** condition in PERP is triggered we break the **for** loop, the **if** condition can happen at most once thoughout the algorithm. We consider two cases separately depending on whether the **if** condition happens or not:

Case 1: the if condition happens at some time s. First, suppose

$$\sum_{t=n+1}^{T} \left|\hat{\mu}_t^{\text{fixed}} - \mu_t\right| < (\gamma\sqrt{\log T} + \sqrt{\kappa}) \cdot T^{(3+v)/4},$$

then because the **if** condition happens at time s,

$$\begin{aligned} T^a &= \sum_{t=1}^{T} |a_t - \mu_t| \\ &\overset{(a)}{\geq} \sum_{t=1}^{s} \left|\hat{\mu}_t^a - \mu_t\right| \\ &\overset{(b)}{\geq} \sum_{t=1}^{s} \left|\hat{\mu}_t^a - \hat{\mu}_t^{\text{fixed}}\right| - \sum_{t=1}^{s} \left|\hat{\mu}_t^{\text{fixed}} - \mu_t\right| \\ &\overset{(c)}{\geq} T^{(3+v)/4}. \end{aligned}$$

where $a_t = \hat{\mu}_t^a$ gives (a), (b) follows by triangle inequality, and (c) follows by the the **if** condition. This shows $a \geq \frac{3+v}{4}$, so $\min\left\{\frac{3+v}{4}, a\right\} = \frac{3+v}{4}$. Note that between time $n+1$ and time $s-1$ we have:

$$\begin{aligned} \sum_{t=n+1}^{s-1} \left|\hat{\mu}_t^a - \mu_t\right| &\overset{(a)}{\leq} \sum_{t=n+1}^{s-1} \left|\hat{\mu}_t^a - \hat{\mu}_t^{\text{fixed}}\right| + \sum_{t=n+1}^{s-1} \left|\hat{\mu}_t^{\text{fixed}} - \mu_t\right| \\ &\overset{(b)}{\leq} (2\gamma\sqrt{\log T} + 2\sqrt{\kappa} + 1) \cdot T^{(3+v)/4}, \end{aligned}$$

where (a) follows by triangle inequality and (b) follows by the **if** condition (note by the algorithm's construction $s \geq n+1$; for the simplicity of writing we assume $s-1 \geq n+1$, and the case $s = n+1$ follows similarly). Then by Observation 8, let ℓ be the Lipschitz constant, we have

$$\begin{aligned} \mathcal{R}^{\pi^{\text{PERP}}}(T)[n+1, s-1] &= \sup_{\boldsymbol{D} \in \mathcal{D}(v)} \mathbb{E}_{D,a}^{\pi^{\text{PERP}}} \left\{ \sum_{t=n+1}^{s-1} \left(C\left(\mu_t, q_t\right) - C\left(\mu_t, q_t^*\right)\right) \right\} \\ &\overset{(a)}{\leq} 2\ell \cdot \sup_{\boldsymbol{D} \in \mathcal{D}(v)} \mathbb{E}_{D,a}^{\pi^{\text{PERP}}} \left\{ \sum_{t=n+1}^{s-1} \left|\hat{\mu}_t^a - \mu_t\right| \right\} \\ &\leq 2\ell(2\gamma\sqrt{\log T} + 2\sqrt{\kappa} + 1) \cdot T^{(3+v)/4}, \end{aligned}$$

where (a) is because $\pi^{\text{PERP}} = \pi^{\text{prediction}}$ before time s. Hence $\mathcal{R}^{\pi^{\text{PERP}}}(T)[n+1, s-1]$ is on the order of $T^{(3+\nu)/4}\sqrt{\log T}$, so there exists some universal constant $C_2 \in (0, \infty)$ such that

$$\mathcal{R}^{\pi^{\text{PERP}}}(T)[n+1, s-1] \leq C_2 T^{(3+\nu)/4}\sqrt{\log T}.$$

Also, since after time s we have $\pi^{\text{PERP}} = \pi^{\text{fixed}}$, by Lemma 3 there exists some universal constant $C_3 \in (0, \infty)$ such that

$$\mathcal{R}^{\pi^{\text{PERP}}}(T)[s, T] = \mathcal{R}^{\pi^{\text{fixed}}}(T)[s, T] \leq \mathcal{R}^{\pi^{\text{fixed}}}(T) \leq C_3 T^{(3+\nu)/4}\sqrt{\log T}.$$

In summary, if

$$\sum_{t=n+1}^{T} \left|\hat{\mu}_t^{\text{fixed}} - \mu_t\right| < (\gamma\sqrt{\log T} + \sqrt{\kappa}) \cdot T^{(3+\nu)/4},$$

we have

$$\begin{aligned}
\mathcal{R}^{\pi^{\text{PERP}}}(T) &= \mathcal{R}^{\pi^{\text{PERP}}}(T)[1, n] + \mathcal{R}^{\pi^{\text{PERP}}}(T)[n+1, s-1] + \mathcal{R}^{\pi^{\text{PERP}}}(T)[s, T] \\
&\leq (C_1 + C_2 + C_3)\, T^{(3+\nu)/4}\sqrt{\log T} \\
&= (C_1 + C_2 + C_3)\, T^{\min\{(3+\nu)/4, a\}}\sqrt{\log T},
\end{aligned}$$

where the last equality is because $\min\left\{\frac{3+\nu}{4}, a\right\} = \frac{3+\nu}{4}$.

Second, suppose $\sum_{t=n+1}^{T} \left|\hat{\mu}_t^{\text{fixed}} - \mu_t\right| \geq (\gamma\sqrt{\log T} + \sqrt{\kappa}) \cdot T^{(3+\nu)/4}$. By Observation 7 there exists some universal constant $C_4 \in (0, \infty)$ such that $\mathcal{R}^{\pi^{\text{PERP}}}(T) \leq C_4 T$. Therefore combining the above two scenarios we get

$$\begin{aligned}
\mathcal{R}^{\pi^{\text{PERP}}}(T) \leq\ & \mathbb{P}\left\{ \sum_{t=n+1}^{T} \left|\hat{\mu}_t^{\text{fixed}} - \mu_t\right| < (\gamma\sqrt{\log T} + \sqrt{\kappa})\, T^{(3+\nu)/4} \right\} \\
& \cdot (C_1 + C_2 + C_3)\, T^{\min\{(3+\nu)/4, a\}}\sqrt{\log T} \\
& + \mathbb{P}\left\{ \sum_{t=n+1}^{T} \left|\hat{\mu}_t^{\text{fixed}} - \mu_t\right| \geq (\gamma\sqrt{\log T} + \sqrt{\kappa})\, T^{(3+\nu)/4} \right\} C_4 T \\
\leq\ & (C_1 + C_2 + C_3)\, T^{\min\{(3+\nu)/4, a\}}\sqrt{\log T} + \frac{2}{T} \cdot C_4 T \\
=\ & (C_1 + C_2 + C_3)\, T^{\min\{(3+\nu)/4, a\}}\sqrt{\log T} + 2C_4.
\end{aligned}$$

Hence $\mathcal{R}^{\pi^{\text{PERP}}}(T) \leq C T^{\min\{(3+\nu)/4, a\}}\sqrt{\log T}$ for some universal constant $C \in (0, \infty)$.

Case 2: the if condition does not happen. In this case $\pi^{\text{PERP}} = \pi^{\text{prediction}}$, so by Observation 3a) we immediately have $\mathcal{R}^{\pi^{\text{PERP}}}(T) \leq C_5 T^a\sqrt{\log T}$ for some universal

constant $C_5 \in (0, \infty)$. Also, following the same analysis as the part of Case 1 where an **if** condition has not happened, i.e., between time $n+1$ and time $s-1$, we get

$$\mathcal{R}^{\pi^{\text{PERP}}}(T)[n+1, T] \leq 2l(2\gamma\sqrt{\log T} + 2\sqrt{\kappa} + 1) \cdot T^{(3+v)/4} \leq C_6 T^{(3+v)/4}\sqrt{\log T}$$

for some universal constant $C_6 \in (0, \infty)$. Then we have

$$\mathcal{R}^{\pi^{\text{PERP}}}(T) = \mathcal{R}^{\pi^{\text{PERP}}}(T)[1, n] + \mathcal{R}^{\pi^{\text{PERP}}}(T)[n+1, T] \leq (C_1 + C_6)\, T^{(3+v)/4}\sqrt{\log T}.$$

Hence take $C = \max\{C_5, C_1 + C_6\}$ we have

$$\mathcal{R}^{\pi^{\text{PERP}}}(T) \leq C T^{\min\{(3+v)/4, a\}}\sqrt{\log T}.$$

■

C.8 Unknown v and Known a

We design a policy for the case of unknown v and known a. Our policy utilizes the famous Exp3 algorithm (Exponential-weight Algorithm for Exploration and Exploitation) as a subroutine. For the sake of completeness we state Exp3 in our setting and its regret bound below. One can refer to [13] for a more detailed discussion on Exp3.

Proposition 21 (Corollary 3.2 in [13])**.** *Exp3 achieves worst-case regret*

$$\mathcal{R}^{\text{Exp3}}(T) \leq \min\{\mathcal{R}^{\pi^{\text{shrinking}}}(T), \mathcal{R}^{\pi^{\text{prediction}}}(T)\} + 2\sqrt{2(\ln 2)(e-1)C_{\max}} \cdot \sqrt{T}.$$

We refer the readers to [13] for the proof. Note that Exp3 incurs an additive $\sqrt{T}$ regret on top of $T^{\min\{(3+v)/4, a\}}$, which is a lower order term if $a > \frac{1}{2}$. On the other hand, if $a \leq \frac{1}{2}$, we have $T^{\min\{(3+v)/4, a\}} = T^a$, so we can simply apply the Prediction Policy. This idea gives the policy of unknown v and known a.

Observation 9 (Upper Bound: Unknown v and Known a)**.** *For any variation parameter $v \in [0, 1]$ and any accuracy parameter $a \in [0, 1]$, the Divide-Into-Cases Policy π^{Divide} achieves worst-case regret*

$$\mathcal{R}^{\pi^{\text{Divide}}}(T) \leq C T^{\min\{(3+v)/4, a\}} \log^{5/2} T,$$

where C is a universal constant.

Proof of Theorem 9. If $a \leq \frac{1}{2}$, then $T^{\min\{(3+v)/4, a\}} = T^a$. The result follows from Observation 3. If $a > \frac{1}{2}$, then $\sqrt{T} = O(T^{\min\{(3+v)/4, a\}})$. The result follows from Proposition 21 and Theorem 2. ■

Algorithm 19: Exp3 (with $\pi^{\text{shrinking}}$ and $\pi^{\text{prediction}}$)

Inputs: $\pi^{\text{shrinking}}$ from Algorithm 5 and $\pi^{\text{prediction}}$ from Algorithm 6

Initialization: $w^{\text{shrinking}}(1) = w^{\text{prediction}}(1) = 1$ and parameter $\gamma = \min\left\{1, \sqrt{\frac{2\ln 2}{(e-1)C_{\max}T}}\right\}$

for $t = 1, \ldots, T$ **do**

  $p^{\text{shrinking}}(t) \leftarrow (1-\gamma)\frac{w^{\text{shrinking}}(t)}{w^{\text{shrinking}}(t)+w^{\text{prediction}}(t)} + \frac{\gamma}{2}$ and

  $p^{\text{prediction}}(t) \leftarrow (1-\gamma)\frac{w^{\text{prediction}}(t)}{w^{\text{shrinking}}(t)+w^{\text{prediction}}(t)} + \frac{\gamma}{2}$

  $\pi_t \leftarrow \pi_t^{\text{shrinking}}$ with probability $p^{\text{shrinking}}(t)$ and $\pi_t \leftarrow \pi_t^{\text{prediction}}$ with probability $p^{\text{prediction}}(t)$

  Obtain d_t

  if $\pi_t \leftarrow \pi_t^{\text{shrinking}}$ **then**

    $\delta_t = (b_t(d_t - q_t^{\text{shrinking}})^+ + h_t(q_t^{\text{shrinking}} - d_t)^+)/p^{\text{shrinking}}(t)$

    $w^{\text{shrinking}}(t+1) = w^{\text{shrinking}}(t)\exp(-\gamma\delta_t/2)$

  end

  else

    $\delta_t = (b_t(d_t - q_t^{\text{prediction}})^+ + h_t(q_t^{\text{prediction}} - d_t)^+)/p^{\text{prediction}}(t)$

    $w^{\text{prediction}}(t+1) = w^{\text{prediction}}(t)\exp(-\gamma\delta_t/2)$.

  end

end

Algorithm 20: Divide-Into-Cases Policy

Inputs: accuracy parameter $a \in [0,1]$, $\pi^{\text{shrinking}}$ from Algorithm 5, and $\pi^{\text{prediction}}$ from Algorithm 6

if $a \le \frac{1}{2}$ **then**

  $\pi \leftarrow \pi^{\text{prediction}}$

end

else

  $\pi \leftarrow \pi^{\text{Exp3}}$.

end

C.9 Experiment Details

In the synthetic experiment we used triple exponential smoothing (Holt Winters) to generate the demand sequences. Triple exponential smoothing takes in the following parameters: data smoothing factor $\alpha \in [0, 1]$, trend smoothing factor $\beta \in [0, 1]$, seasonal change smoothing factor $\gamma \in [0, 1]$, and season length $L \in \mathbb{N}$. Given historical observations $x_1, \ldots, x_t (t \geq L)$, triple exponential smoothing outputs $\hat{x}_{t+m}$ for $m \geq 1$, which is an estimate of x_{t+m}, according to the following formula:

$$\begin{aligned} s_0 &= x_0 \\ s_t &= \alpha \frac{x_t}{x_{t-L}} + (1-\alpha)(s_{t-1} + b_{t-1}) \\ b_t &= \beta(s_t - s_{t-1}) + (1-\beta) b_{t-1} \\ c_t &= \gamma \frac{x_t}{s_t} + (1-\gamma) c_{t-L} \\ x_{t+m} &= (s_t + m b_t) c_{t-L+1+(m-1) \bmod L}, \end{aligned}$$

where s_t is the smoothed value of the constant part for time t, b_t is the sequence of best estimates of the linear trend that are superimposed on the seasonal changes, and c_t is the sequence of seasonal correction factors.

In our experiment, we first generated a demand sequence for 30 time periods where the demand at each time period is drawn uniformly between 80 and 120. This was treated as the historical observations and was fixed throughout the experiment. Then for each set of parameters $(\alpha, \beta, \gamma, L)$, which we will specify later in each case, we used triple exponential smoothing to generate the mean of demands for 365 time periods, where at each time period we also added a random Gaussian noise with mean equals to 0 and variance equals to 5. The true demand at each time period was generated as a Poisson variable with the corresponding mean.

We ran two sets of experiments:

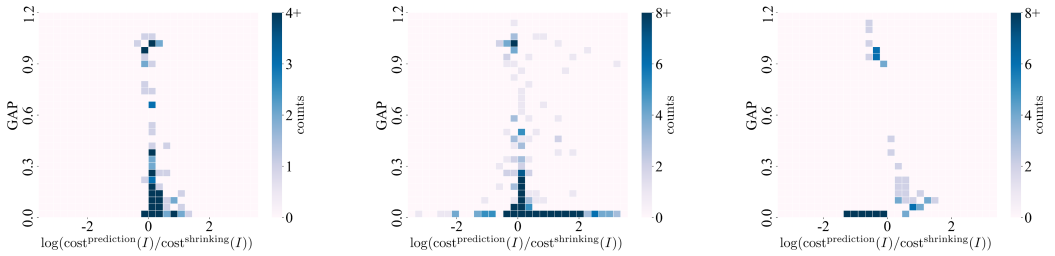
- **Fixed v:** We fixed a single set of parameters $(\alpha, \beta, \gamma, L) = (0.5, 0.5, 0.5, 30)$ for the demand sequence, and generated 1,000 different predictions of this demand sequence, each from a set of "predicted" parameters $(\hat{\alpha}, \hat{\beta}, \hat{\gamma}, \hat{L})$ where each $\hat{\alpha}, \hat{\beta}, \hat{\gamma}$ was sampled uniformly at random from $[0.2, 0.8]$ and $\hat{L}$ from $\{10, 20, 30\}$. Thus the variation parameter v was fixed, and the accuracy parameter a varied across instances.

- **Fixed a:** We generated 1,000 demand sequences by selecting the parameters $(\alpha, \beta, \gamma, L)$ uniformly at random where each α, β, γ was sampled uniformly at

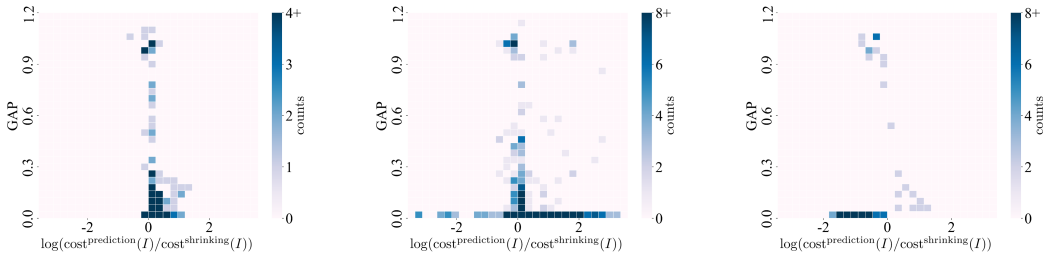
random from $[0.2, 0.8]$ and L from $\{10, 20, 30\}$. We then generated predictions by changing each parameter 10% (e.g., α becomes either 1.1α or 0.9α) and using the corresponding sequence. Thus the variation parameter v varied across instances, but the accuracy parameter a was (roughly) fixed.

C.10 Additional Experimental Results

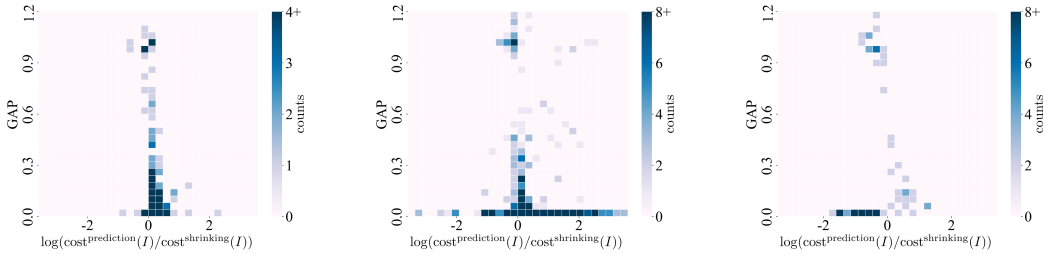
In our experiments on real data, we used four popular forecasting method to generate predictions for each instance: Exponential Smoothing (Holt Winters), ARIMA, Prophet, and LightGBM. Experiments on the datasets yielded the histograms in Figure 4.4. To show the performance of PERP is robust to the forecasting method, in Figure C.1 we further divide the three histograms in Figure 4.4 into twelve histograms separated by the four forecasting methods.



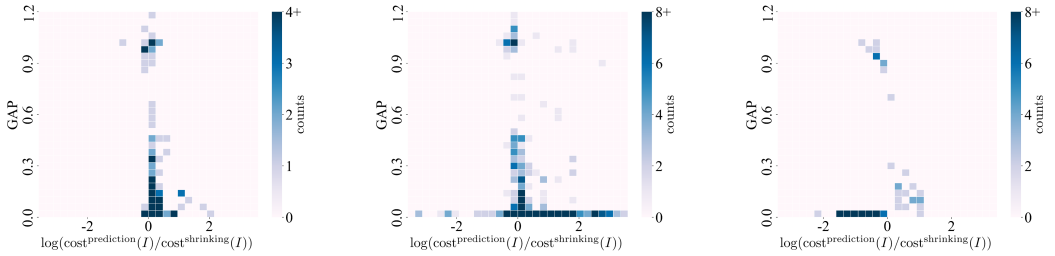
(a) GAPs with Exponential Smoothing forecasts. Left to right: Rossmann, Wikipedia, Restaurant.



(b) GAPs with ARIMA forecasts. Left to right: Rossmann, Wikipedia, Restaurant.



(c) GAPs with Prophet forecasts. Left to right: Rossmann, Wikipedia, Restaurant.



(d) GAPs with ELightGBM forecasts. Left to right: Rossmann, Wikipedia, Restaurant.

Figure C.1: Histograms of GAPs divided by forecasting methods across the Rossmann dataset, the Wikipedia dataset, and the Restaurant dataset.