DATA-DRIVEN DECISION MAKING IN OPERATIONS MANAGEMENT

by

XIAOYUE GONG

BACHELOR OF SCIENCE IN HONORS MATH, NEW YORK UNIVERSITY (2017)

BACHELOR OF SCIENCE IN INTERACTIVE MEDIA ARTS, NEW YORK UNIVERSITY (2017)

Submitted to the Sloan School of Management in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN OPERATIONS RESEARCH

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 2023

Signature of Author ....
Sloan School of Management
MAY 03

Certified by..
DAVID SIMCHI-LEVI
PROFESSOR OF ENGINEERING SYSTEMS
Thesis Supervisor

Accepted by..
GEORGIA PERAKIS
WILLIAM F. POUNDS PROFESSOR OF MANAGEMENT SCIENCE
Co-director, Operations Research Center

Data-Driven Decision Making in Operations Management

by

Xiaoyue Gong

Submitted to the Sloan School of Management
on May 2, 2023, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

Encouraged by the plethora of advances in artificial intelligence (AI) in the past decade, this thesis studies the length to which we can push various business operations with new technologies, in our theoretical understanding and practical performance alike. Towards this goal, this thesis develops data-driven decision-making methods for a selection of challenging emerging problems in supply chain and other business operations.

In the first module of the thesis (Chapter 2 and 3), we invent reinforcement learning methods with provable optimality guarantees for inventory management problems. The challenge in the inventory problems that we are interested in is that the demand distribution varies over time according to some natural cyclic patterns (such as weekly sales cycles), and we are in the *online* setting where we do not have prior knowledge of the demand distribution or access to prior data. Solutions to these inventory models have been carefully studied for decades in the *offline* setting where the cyclic demand distribution is known beforehand; however, very few results have been attained in the online setting. The complexity of the problem motivated us to introduce reinforcement learning into the picture. Our design of a reinforcement learning algorithm has an optimal $\tilde{\mathcal{O}}(\sqrt{T})$ regret bound for a number of inventory models with unknown cyclic demands that we study in these chapters.

In the second module of the thesis (Chapter 4), we study *online* assortment optimization for *reusable* resources. E-commerce platforms like Amazon and Expedia constantly endeavor to recommend more favorable assortments of products and services to their customers. The choice of assortment influences customer purchasing decisions, and can thus significantly impact the platform's revenue. We consider assortment optimization with reusable resources, which means that the product returns to the inventory once the customer has finished using it. Reusability arises in major applications including cloud services, physical storage, and make-to-order service. The unpredictability of the usage times means that planning ahead becomes more challenging. We show that a simple greedy policy is 1/2 competitive for online assortment optimization with reusable resources. This means that on average, the greedy policy earns at least half the revenue of a clairvoyant optimal policy which has access to much more information. This result is surprising because the greedy policy does not take into account the customer or usage time distributions, both of which are necessary to solve for the optimal policy.

In the third module of the thesis (Chapter 5), we develop practical solutions for the cloud service supply chain at Microsoft Azure. The cloud computing industry boomed in the past few years as digitization continues to take place globally and as remote work becomes more of a norm. A main challenge faced by cloud service providers is to deploy cloud server hardware under demand uncertainty, without incurring unnecessarily large operational costs. We formulate the underlying optimization problem as a two-stage stochastic program. We then develop exact Benders-type algorithms that exploit the structure of the second stage problem. We test our proposed algorithms with numerical experiments based on real production traces from Microsoft Azure, which demonstrate noticeable advantages of our algorithms over existing heuristics used in production. Given the large scale of the problem, our deployment policy could potentially lead to savings of hundreds of millions of dollars per year.

Thesis Supervisor: David Simchi-Levi
Title: Professor of Engineering Systems

Acknowledgments

I would like to thank my advisors Professor David Simchi-Levi and Professor James Orlin, for the guidance, wisdom and support that they have given me throughout my Ph.D. journey.

I would like to thank my committee members Professor John N. Tsitsiklis and Professor Negin Golrezaei, who have been helpful and supportive at various phases of my Ph.D. journey.

I would also like to thank my good friends, collaborators, teachers and classmates inside and outside MIT. The journey has been fun because of you.

I have to thank my undergraduate math professor Joel Spencer, for suggesting that I apply to MIT Operations Research Center's Ph.D. Program.

I would like to thank my parents. Mama and Papa have always believed that I can be anything I want and I can achieve anything I want. They are the ones who made the greatest sacrifice for my Ph.D., as they have endured years of being apart from me.

Finally, thank you to Mark. We met during a Ph.D. admission open house, and got engaged during our separate Ph.D studies. In spite of our geographical distance throughout the years, Mark made me feel absolutely adored and treasured every day. Thanks for making sure that I always feel like the shiniest star in the universe, even on the bleakest days.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The plethora of advances in artificial intelligence in recent years have brought forth exciting opportunities in various domains. This thesis studies to what extent we can push business operations with new technologies, ranging from theoretical understanding to practical performance improvements.

The first module of the thesis (Chapter 2 and 3) invents reinforcement learning methods with provable optimality guarantees for inventory management problems where the demand distribution varies over time according to some natural cyclic patterns (such as weekly sales cycles), and we are in the *online* setting where we do not have prior knowledge of the demand distribution or access to prior data. The complexity of the problem motivated us to introduce reinforcement learning into the picture. The second module of the thesis (Chapter 4) studies *online* assortment optimization for *reusable* resources. E-commerce platforms like Amazon and Expedia constantly endeavor to recommend more favorable assortments of products and services to their customers. The choice of assortment influences customer purchasing decisions, and can thus significantly impact the platform's revenue. We consider assortment optimization with reusable resources, which means that the product returns to the inventory once the customer has finished using it. Reusability arises in major applications including cloud services, physical storage, and make-to-order service. The unpredictability of the usage times means that planning ahead becomes more challenging. Finally in the last module (Chapter 5), we develop practical solutions for the cloud service supply chain at Microsoft Azure, to deploy cloud servers under demand uncertainty, without incurring unnecessarily large operational costs.

The rest of this section outlines these modules and our main contributions in each.

1.1 Overview of Module 1 (Chapters 2 and 3)

Chapter 2. Inventory management is one of the most fundamental problems in supply chain optimization (Zipkin (2000)). In this problem, at every time period, the retailer reviews her on-hand inventory and orders replenishment based on current inventory and constraints, with the goal of minimizing the total costs over the planning horizon.

Cyclic demands is a popular term used in literature to refer to stochastic demands whose distribution follow some cyclic pattern. This corresponds to the intuition that for many products, there are natural sales cycles, e.g. weekly cycles where the demand distribution on Wednesday can be very different from Saturday. Solutions to these inventory models have been carefully studied for decades in the offline setting where the cyclic demand distribution is known beforehand; however, few results have been attained in the online setting. The only existing result prior to our work is by Huh and Rusmevichientong (2014) which applies their algorithm to the *episodic* inventory model with unknown cyclic demand distributions. The *episodic* models is defined by its strong assumption that inventory is assumed to be discarded at the end of every demand cycle. In this chapter, we invent reinforcement learning methods for the episodic model. In Chapter 3, we invent additional machinery that allows us to remove this unreasonable assumption, and go from the episodic model to the non-discarding model.

Contributions in Chapter 2 (Gong and Simchi-Levi (2021))

- We prove that the regret of our policy, *Elimination-Based Half Q-Learning (HQL)*, is upper-bounded by $\tilde{\mathcal{O}}(\sqrt{T})$ regret for the online episodic lost-sales model with unknown cyclic demand distribution.

- We show that the regret for any algorithm on the inventory problems with unknown cyclic demand distributions is lower-bounded by $\mathcal{O}(\sqrt{T})$, thus proving the optimality of *HQL* (up to an $\mathcal{O}(\sqrt{\log T})$ factor).

- Compared with Huh and Rusmevichientong (2014), we are able to reduce the regret dependence on the cycle length from exponential dependence in their paper, to a low polynomial dependence in this chapter.

- Compared with existing reinforcement learning methods, we are able to remove the regret dependence on the cardinality of the state and action space from our regret bound for the inventory problems.

We design provably efficient reinforcement learning (RL) algorithms that leverage the structure of inventory problems. We apply the standard performance measure in online learning literature, *regret*, which is defined as the difference between the total expected cost of our policy and the total expected cost of the clairvoyant optimal policy that has full knowledge of the demand distributions a priori. This chapter analyzes, in the presence of unknown cyclic demands, the lost-sales model with zero lead time, and the multi-product backlogging model with positive lead times, fixed joint-ordering costs and order limits. For both models, in this chapter, we first study *episodic* models where inventory is discarded at the end of every cycle. Our RL policies *HQL* and *FQL* have $\tilde{\mathcal{O}}(\sqrt{T})$ regret bounds for the episodic lost-sales model and the episodic multi-product backlogging model, matching the regret lower bound that we prove in this chapter. The episodic models were first studied by Huh and Rusmevichientong (2014). In comparison, we are able to reduce the exponential dependence on the cycle length in their regret bound to a low polynomial dependence for our regret bound.

Chapter 3. The results in Chapter 2 are not entirely satisfactory to us because the assumption that inventory is discarded at the end of every demand cycle is unreasonable. To remove this assumption, i.e., to be able to work with the non-discarding models, we construct a bandit learning algorithm on top of the previous RL algorithms, named *Meta-HQL*.

Contributions in Chapter 3 (Gong and Simchi-Levi (2021))

- We prove that the regret of *Meta-HQL* is upper-bounded by $\tilde{\mathcal{O}}(\sqrt{T})$ regret for the online non-discarding lost-sales model with unknown cyclic demand distribution.

- Our regret bound is again unaffected by the cardinality of the state-action space.

- The design of *Meta-HQL* allows easy extension to some variants of the inventory model, such as when the cycle length is unknown beforehand, or when we are only allowed to order a fixed number of times during a cycle, etc.

- We conduct numerical experiments using a real sales dataset from Rossman, which show the superiority of our algorithms over algorithms that assume demand is independent and identically distributed over time.

1.2 Overview of Module 2 (Chapter 4)

Chapter 4. In this chapter, we consider an online assortment optimization problem where we have n substitutable products with fixed reusable capacities. In each period, a user with some preferences (potentially adversarially chosen) arrives to the seller's platform who offers a subset of products from the set of available products. The user selects a product with some probability given by the preference model and uses it for a random number of periods. The usage time is distributed i.i.d. according to some distribution that depends only on j. This selection generates a revenue for the seller. The goal of the seller is to find a policy that maximizes the expected cumulative revenue over a finite horizon T.

Contributions in Chapter 4 (Gong et al. (2022))

- Our main contribution is to show that a simple myopic policy (where we offer the myopically optimal assortment from the available products to each user) provides a good approximation for the problem. In particular, we show that the myopic policy is 1/2-competitive, i.e., the expected cumulative revenue of the myopic policy is at least half the expected revenue of the optimal policy with full information about the sequence of user preference models and the distribution of random usage times of all the products.

- We prove that in the case that the usage time distributions can depend on the type of each user, then there is no online algorithm with a non-trivial competitive ratio guarantee.

- We perform numerical experiments to compare the robustness and performance of myopic policy with other natural policies.

1.3 Overview of Module 3 (Chapter 5)

In the last chapter, we develop practical solutions for the cloud service supply chain at Microsoft Azure. With rapidly increasing demand for cloud services in recent years, it is of growing importance for cloud service providers to ensure that their data centers are ready to accommodate the demand for computing resources. The main challenge faced by providers is to deploy cloud servers under future demand uncertainty, without incurring unnecessarily large operational costs. In this chapter, we introduce the *cloud server deployment problem.* We formulate the underlying optimization problem as a two-stage stochastic program.

Contributions in Chapter 5

- We develop exact Benders-type algorithms that exploit the special structure of the second stage problem.

- We test our proposed algorithms with real production traces from Microsoft Azure, and quantify the advantages of our algorithms over existing heuristics used in production.

Chapter 2

Reinforcement Learning for Episodic Inventory Management with Unknown Cyclic Demands

2.1 Introduction

Inventory management is one of the most fundamental problems in supply chain optimization (Zipkin (2000)). In this problem, at every time period, the retailer reviews her on-hand inventory and orders replenishment based on current inventory and constraints, with the goal of minimizing the total costs over the planning horizon. For decades, various inventory models have been of tremendous interests to practitioners and researchers, and the theory under full knowledge of the demand distribution is well-studied.

However, in real applications, rarely does a retailer know the true demand distributions a priori. More often than not, the retailer must make decisions in an adaptive *online* manner, which assumes that no data is available beforehand and that data can only be collected as decisions are made. Time and again, we have seen that being able to do so efficiently is critical for businesses, especially for business expansions and for new product introductions. This is evidenced by the famous case of Target's failed expansion into the Canadian market in 2013-2015, which was largely due to over-optimistic inventory planning decisions made based on U.S. sales history (Lim (2016)). Moreover, trends and large events can also impact the reliability of past data. During the Covid-19 pandemic, various sectors have observed drastic changes in consumer behavior and consumption, rendering past demand data impractical. Even with plentiful valid past data, the most appropriate characterization of demand distributions remains ambiguous. Perakis and Roels (2008) find that this ambiguity can lead to confusion about the optimal inventory policy and incur large extra costs.

All these challenges call for adaptive online inventory decision models. Large swathes of research effort has achieved exciting results in pursuit of this goal, see, for example, Huh and Rusmevichientong (2009a), Agrawal and Jia (2022), Zhang et al. (2020) and Yuan et al. (2021). A notable caveat is that the vast majority of existing online inventory models make the strong assumption that demands are identically distributed across the time horizon, which is often far from reality. Real demand distributions are often time-varying and have cyclic patterns that correspond to natural customer behavior. Some commonly observed cyclic demand patterns include weekly, monthly and yearly (seasonal) patterns. Ehrenthal et al. (2014) analyze real data to show that ignoring seasonality in demand modeling can lead to a large profit loss.

Cyclic demands is a popular term used in the literature to refer to stochastic demands whose distributions follow a cyclic pattern. This corresponds to the intuition that for many products, the demands on Tuesdays are generated from one distribution, and the demands on Sundays are generated from a very different distribution. Cyclic demand modeling has been extensively studied in traditional inventory literature where demand distributions are known beforehand. Among those studies, Zipkin (1989) notes that techniques used to obtain the optimal policy for i.i.d. demands no longer work and new methods must be developed. Unfortunately, in the online setting, where demand distributions are unknown, incorporating cyclic stochastic patterns into demand modeling presents significant additional challenges. For example, the learning algorithm that combines stochastic gradient descent with bandits in Yuan et al. (2021) is able to achieve $\tilde{\mathcal{O}}(\sqrt{T})$ regret for the lost-sales model with fixed costs with i.i.d. demands against the clairvoyant optimal policy, but cannot be applied to the lost-sales model with cyclic demands even if there is no fixed cost. One reason is that their analysis relies on the fact that for (s, S) policies, after some transformation, once one coordinate of the decision space is fixed, the remaining coordinate is a convex optimization problem and stochastic gradient descent can be applied. This convexity no longer exists when the demand is cyclic instead of i.i.d., even when there are no fixed costs. Another difficulty is that for i.i.d. demands, the expected average cost for holding a fixed amount of inventory does not depend on time, but when demands are cyclic, this average cost is highly sensitive to time. These difficulties dismantle the analyses in prior works, leaving a gap at the intersection of online inventory management and cyclic demands.

Motivated by this long-standing gap between inventory theory and practice, we study online periodic-review inventory models with unknown cyclic stochastic demands. We design provably optimal bandits-atop-reinforcement-learning algorithms that cater to the structure of inventory problems. Our inspiration comes from the fact that many of the challenging properties of cyclic demands, such as the previously mentioned non-convexity and sensitivity to time, fall nicely into the regime of Markov decision processes (MDP), which can be analyzed under the framework of episodic reinforcement learning (RL).

We apply the standard performance measure in online learning literature, *regret*, to evaluate our algorithms. The regret of a policy is defined as the difference between the total expected cost of the evaluated policy over the planning horizon and that of the clairvoyant optimal policy with full information of the demand distributions a priori. Regret is also the performance measure used in prior online inventory literature (Huh and Rusmevichientong (2009a), Agrawal and Jia (2022)). We show that our policies are optimal for a number of models by proving matching regret lower bounds. By leveraging the structure of inventory models in reinforcement learning, we remove the regret dependence on the cardinality of the state-action space in our policies, which is a considerable improvement over existing RL algorithms.

In addition to theoretical guarantees, we test the practical performance of our policy with real sales data from one of the largest drugstore chains in Europe, Rossmann. We observe that our policy converges rapidly to the clairvoyant optimal policy. Our policy and the clairvoyant optimal policy dramatically outperform the best policy that models the demands as i.i.d. instead of cyclic, which manifests the practical impact of modeling cyclic demands.

Furthermore, our policies allow the input of *expert advice* to further improve performance in practice. We broadly define expert advice as any reasonable prior knowledge about either the cyclic demand distributions or the optimal policy, that can come from domain knowledge, experience or existing data. This also connects our online policies to offline learning. We demonstrate the benefit of this capability in our numerical experiments conducted with the Rossmann dataset. Importantly, some of our algorithms apply more generally to operations research problems beyond inventory management. Example applications include airline overbooking policy, portfolio optimization and online repeated second-price auctions, as discussed in Section 3.4.

2.2 Lost-Sales Problem Formulation

We formally describe a discrete-time finite-horizon periodic-review inventory problem for a single product under lost sales and censored cyclic stochastic demand. The horizon consists of a sequence of cycles of the same constant length H that partitions the T periods, where $H = \mathcal{O}(1)$. Let K denote the number of cycles in the horizon, $K = \frac{T}{H} = \Theta(T)$. As an example, a horizon of 5 years is $K = 261$ cycles of $H = 7$ days.

At the beginning of period t, the retailer reviews the current inventory x_t, and places an order that raises the inventory level up to a level $y_t \geq x_t$. The purchasing cost is assumed to be zero without loss of generality (for proof see Appendix A.4). Replenishment of $y_t - x_t$ units arrives instantly with zero lead time, after which an *unobserved* random demand D_t from unknown distribution F_t is realized. The demands are independent, but not necessarily identically distributed: the distributions within a cycle can be arbitrary. In mathematical terms, all demands D_t can be partitioned into H subsets based on the remainder of $t \mod H$, and demands in the same subset are independently generated from the same distribution.

The retailer applies the *on-hand* inventory, which is defined as the sum of the current inventory x_t and the instant replenishment $y_t - x_t$, to satisfy the realized demand D_t. If the demand does not exceed the on-hand inventory, the retailer pays a holding cost $o_t > 0$ for each unit of leftover inventory at the end of period t. The starting inventory for the next period is the leftover inventory $x_{t+1} = (y_t - D_t)^+$.

If the demand exceeds the on-hand inventory, the retailer pays a penalty of $p_t > 0$ for each unit of lost demand. However, the retailer is not aware of the amount of lost-sales penalty she pays because the lost part of the demand is unobserved. The starting inventory for the next period is then $x_{t+1} = 0$.

In summary, the cost function of period t is:

$$\text{Cost}_t(y_t) = o_t(y_t - D_t)^+ + p_t(D_t - y_t)^+ \tag{2.1}$$

which is unobservable because $(D_t - y_t)^+$ is the lost part of the demand.

Now we discuss the difference between the *episodic* model and the *non-discarding* model.

- **Model 1: Episodic Lost-Sales**

  The episodic model is defined by the following assumption:

  Assumption 1a: At the end of each cycle, excess inventory is discarded or salvaged at some arbitrary fixed price.

  In the case of salvaging, we subtract the salvaging revenue from the cost of the last step in a cycle so that Equation (2.1) holds for all $h \in [H]$.

  Even though we mainly use the episodic model as a building block for our analysis of the non-discarding model in Chapter 3, the episodic model itself is applicable to products or services that perish at the end of every cycle. This model was first introduced in Huh and Rusmevichientong (2014).

- **Model 2: Non-Discarding Lost-Sales**

This model is consistent with the inventory literature for non-perishable products: we never discard inventory. The leftover inventory at the end of a cycle always carries over to the next cycle. Instead of Assumption 1a for the episodic model, we have the following alternative Assumption 1b for the non-discarding model:

Assumption 1b: The expected time to deplete 1 unit of inventory is γ.

This is an often-used assumption in the literature (see for example Agrawal and Jia (2022)). In our chapter, γ can be as large as $\mathcal{O}(\sqrt{T})$. Note that our policy does not need to know γ. This assumption is only used in the analysis.

In this chapter, we study Model 1. Model 2 is studied in Chapter 3. For both the episodic and the non-discarding lost-sales models with zero lead time, we consider the following class of policies:

Cyclic base-stock policies/order-up-to policies are the class of policies that can be characterized by H key values, one for each time step in a cycle. For the same time step $h \in \{1, 2, \dots, H\}$ across cycles, the retailer always orders quantities that bring the inventory position to the hth key value known as the *base-stock level* for time step h, unless it is infeasible to do so based on the current inventory (i.e. the inventory is greater than the base-stock level). This base-stock level may be different for each time step in a cycle, but it is the same for the same time step across cycles.

Remark 1 *We consider only base-stock policies for the episodic and the non-discarding lost-sales models with zero lead time because they can be proven to be optimal for these models (Porteus (2002)).*

Assumptions 2: The range of the base-stock level we consider is $[m, M]$, where M is often associated with the warehouse capacity.

Assumptions 3: The retailer knows the cycle length H, but has *no prior knowledge* of the demand distributions except that they are cyclic.

Assumption 2 is commonly made in theory and in practice. As for Assumption 3, it is quite reasonable to assume that the cycle length is known because the cyclic patterns that real-life demand often follow are weekly, monthly or yearly (seasonal).

The objective of the retailer is to minimize the total expected cost over the entire horizon, or equivalently, to minimize the *regret* of her policy. The notion of regret from online learning is a widely adopted performance measure that is defined as the difference between the total cost of a feasible policy and that of a *clairvoyant* optimal policy OPT that knows the true demand distributions a priori.

For a feasible learning policy π, the regret of π over T periods is

$$\text{Regret}_{\pi}(T) = \mathbf{E}\left[\sum_{t=1}^{T} \text{Cost}_t^{\pi}\right] - \mathbf{E}\left[\sum_{t=1}^{T} \text{Cost}_t^{\text{OPT}}\right].$$

One problem with the regret formulation above is that the retailer cannot observe the realized cost function for the lost-sales models because demands are censored. To overcome this challenge, we use the following observable pseudo-cost (for both OPT and π), which does not change the regret of any policy (see Agrawal and Jia (2022)):

$$\text{Pseudo-Cost}_t(y_t) = o_t(y_t - D_t)^+ - p_t \min(y_t, D_t). \tag{2.2}$$

We allow the unit holding cost and the unit lost-sales penalty to be time-varying as long as they are cyclic in sync with the demand distributions. It follows that both the cost and pseudo-cost functions are also cyclic, so for notation, we will often replace the t subscript with the corresponding step h subscript, $h = 1, \ldots, H$. Furthermore, we will use superscript k to denote cycle k when necessary. The pseudo-cost for step h in cycle k is:

$$\text{Pseudo-Cost}_h(y_h^k) = o_h(y_h^k - D_h^k)^+ - p_h \min(y_h^k, D_h^k). \tag{2.3}$$

The reason why using this pseudo-cost does not affect the regret analysis of any policy is because the difference between the pseudo-cost and the cost is a term that is independent of the policy action y_t. Therefore,

$$\begin{aligned}\text{Regret}_\pi(T) &= \mathbf{E}\left[\sum_{t=1}^{T} \text{Cost}_t^\pi\right] - \mathbf{E}\left[\sum_{t=1}^{T} \text{Cost}_t^{\text{OPT}}\right] \\ &= \mathbf{E}\left[\sum_{k=1}^{K}\sum_{h=1}^{H} \text{Pseudo-Cost}_h^{k^\pi}\right] - \mathbf{E}\left[\sum_{k=1}^{K}\sum_{h=1}^{H} \text{Pseudo-Cost}_h^{k^{\text{OPT}}}\right].\end{aligned}$$

Key Observation 1: At any step h, after we choose action y_h that is a base-stock level, once the demand is realized, we can deduce what the pseudo-cost and leftover inventory would be for all possible base-stock levels that are lower than our chosen action because we can observe the pseudo-cost $o_h(y'_h - D_h)^+ - p_h \min(y'_h, D_h)$ for all $y'_h \leq y_h$. This property of the lost-sales model is called the *one-sided-feedback* structure, which is used by Yuan et al. (2021) for the lost-sales model with fixed costs when demands are i.i.d.

More formally, a model possesses the one-sided-feedback structure if whenever an action y is taken at time t, once the environmental randomness is realized, we can learn what the outcome would have been for any action at that time step that lies on *one side* of y, i.e., all $y' \leq y$ for the *lower*-sided-feedback structure (or all $y' \geq y$ for the *higher*-sided-feedback structure), given that single sample of the environment randomness. This structure is also present in several other applications beyond inventory management; see Section 3.4.

2.2.1 Preliminaries

Without loss of generality, we transform[1] the pseudo-costs to "rewards" so that the reward of any policy in any time period is bounded by $[0, 1]$. The goal of minimizing the total expected cost is equivalent to maximizing the total expected reward.

We formulate the episodic lost-sales model as an episodic MDP$(\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$, where $\mathcal{S}$ is the set of states that denote current inventory levels with $|\mathcal{S}| = S$, $\mathcal{A}$ is the set of actions of base-stock levels with $|\mathcal{A}| = A$, H is the episode length, $\mathbb{P}$ is the unknown transition matrix of the distribution over states for state-action pairs, and $r_h : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the random reward function at step h that depends on the realized demand. Recall that the action space contains only base-stock levels because base-stock policies are optimal for the episodic lost-sales models with zero lead time. If a base-stock level y is lower than the current inventory x, we say that action y is infeasible to state x. We discretize $[m, M]$ with step-size $\frac{M-m}{T^2}$, so the action space size $A = T^2$. In Lemma 2, we show that this discretization incurs $\mathcal{O}(M/K)$ total error. Recall $K = \Theta(T)$, so this error vanishes as T grows.

We introduce the following Q-learning setup, which shares many of the same notations with the setup in Jin, Allen-Zhu, Bubeck and Jordan's work (Jin et al. (2018)).

A policy π is a collection of functions $\{\pi_h : \mathcal{S} \to \mathcal{A}\}_{h \in [H]}$. We use $V_h^\pi : \mathcal{S} \to \mathbf{R}$ to denote the value function at step h under policy π, so that $V_h^\pi(x)$ gives the expected sum of remaining rewards under policy π until the end of the episode, starting from

[1]For the single-product lost-sales model, we take o_h and p_h to be their own additive inverses $-o_h$ and $-p_h$, and then normalize the negated unit costs down by a factor of $\Theta\big(M \max(|o_h|, |p_h|)\big)$. Then we shift the per-period reward to the right by 1 so that it is bounded by $[0, 1]$.

$x_h = x$:

$$V_h^\pi(x) := \mathbf{E}\Big[\sum_{h'=h}^{H} r_{h'}\big(x_{h'}, \pi_{h'}(x_{h'})\big)\Big| x_h = x\Big].$$

We use $Q_h^\pi : \mathcal{S} \times \mathcal{A} \to \mathbf{R}$ to denote the Q-value function at step h, so that $Q_h^\pi(x, y)$ gives the expected sum of remaining rewards under policy π until the end of the episode, starting from $x_h = x, y_h = y$:

$$Q_h^\pi(x, y) := \mathbf{E}\Big[r_h(x_h, y_h) + \sum_{h'=h+1}^{H} r_{h'}\big(x_{h'}, \pi_{h'}(x_{h'})\big)\Big| x_h = x, y_h = y\Big].$$

For the cyclic base-stock policies, it is easy to see that conditioning on the same feasible base-stock level, the reward and leftover inventory do not depend on the current inventory. Therefore, for the episodic lost-sales model with zero lead time, we can simplify the notation $Q_h(x, y)$ to $Q_h(y)$. We must emphasize that the feasibility of an action y still depends on the current state x.

There exists an optimal policy π^* on the finite MDP that has the highest reward and the smallest regret with respect to the clairvoyant optimal policy OPT in the original undiscretized problem. The optimal value functions are given by $V_h^*(x) = \sup_\pi V_h^\pi(x)$ for any $x \in \mathcal{S}$ and $h \in [H]$. The Bellman equations are:

$$\begin{cases} V_h^\pi(x) = Q_h^\pi\left(x, \pi_h(x)\right) \\ Q_h^\pi(x, y) = \mathbb{E}_{x', r_h \sim \mathbb{P}(\cdot|x,y)}\left[r_h + V_{h+1}^\pi\left(x'\right)\right] \\ V_{H+1}^\pi(x) = 0, \quad \forall x \in \mathcal{S} \end{cases} \begin{cases} V_h^*(x) = \max_y Q_h^*(x, y) \\ Q_h^*(x, y) = \mathbb{E}_{x', r_h \sim \mathbb{P}(\cdot|x,y)}\left[r_h + V_{h+1}^*\left(x'\right)\right] \\ V_{H+1}^*(x) = 0, \quad \forall x \in \mathcal{S} \end{cases}$$

Recall the definition of the regret of a policy π. Because of discretization, in the episodic model, the total regret Regret_{total} is the sum of the regret of π against π^*

and the regret of π^* against OPT,

$$\text{Regret}_{total} = \text{Regret}_{MDP} + \text{Regret}_{gap},$$

where

$$\begin{aligned}\text{Regret}_{MDP}(T) &= \mathbf{E}\left[\sum_{k=1}^{K}\sum_{h=1}^{H} r_h^{k^{\text{OPT}}}\right] - \mathbf{E}\left[\sum_{k=1}^{K}\sum_{h=1}^{H} r_h^{k^{\pi_k}}\right] \\ &= \sum_{k=1}^{K}\left[V_1^*\left(x_1^k\right) - V_1^{\pi_k}\left(x_1^k\right)\right].\end{aligned} \tag{2.4}$$

and x_1^k denotes the starting state that the adversary picks for episode $k = 1, \ldots, K$, and π_k denotes the policy the retailer chooses before starting the kth episode.

2.3 Related Literature

Our work stands at the intersection of several prominent areas. We discuss three streams of literature below. The first stream of literature is inventory management that relates to either online learning or to cyclic demands. The second stream is Q-learning, and the third stream is the rich feedback literature, where we also briefly juxtapose our work to *cross-learning* in contextual bandits.

2.3.1 Related Inventory Literature

A large amount of research has been conducted on offline learning for inventory models; we focus on discussing online inventory models. The earliest online inventory literature is Huh and Rusmevichientong (2009b), which analyzes both the perishable and non-perishable products for the lost-sales model with i.i.d. demands, and achieves $\mathcal{O}(\sqrt{T})$ regret against the newsvendor benchmark. Huh et al. (2009) achieve

$\mathcal{O}(T^{2/3})$ regret against the optimal base-stock policy for the lost-sales model with i.i.d. demands and a positive lead time. Zhang et al. (2020) achieve $\mathcal{O}(\sqrt{T})$ regret against the best base-stock policy (which is not necessarily optimal) for the lost-sales inventory model with i.i.d. demands and positive lead time L. Agrawal and Jia (2022) achieve $\tilde{\mathcal{O}}(\sqrt{T})$ regret bound with probability at least $1 - \frac{1}{T}$. for the same model with respect to the same benchmark, the best base-stock policy. Yuan et al. (2021) study the lost-sales problem with i.i.d. demands and a fixed ordering cost K and achieve $\tilde{\mathcal{O}}(\sqrt{T})$ regret against the optimal policy. Other online inventory research include Huh et al. (2011), Davoodi et al. (2019), and Chen and Shi (2019).

The aforementioned literature assumes that demands are i.i.d. across the horizon. However, we are interested in a more general scenario in which demand can have arbitrary distributions that follow a cyclic pattern. Moreover, costs and constraints can also be cyclic. In 1960, Karlin first proposed the offline stochastic inventory problem with demands from known periodic distributions (Karlin (1960)). Examples of follow-up papers on cyclic inventory models include Zipkin (1989) and Aviv and Federgruen (1997). Among these studies, Zipkin (1989) points out that techniques used to obtain the optimal policy for i.i.d. demands no longer work and new methods were developed for the offline setting.

However, managing inventory in the online setting with unknown cyclic demand distributions presents significant additional challenge. As previously discussed, the difficult properties of the cyclic demands problem, such as non-convexity and sensitivity to time, dismantle the analyses in prior works that model i.i.d. demand, leaving a blank at the intersection of online inventory management and cyclic demands. Since Yuan et al. (2021) uses a method that combines stochastic gradient descent with bandits, it is natural to wonder if a similar stochastic gradient descent approach can be applied to the cyclic demand problem. Unfortunately, no straightforward way exists to adapt their algorithm for the cyclic demand problem, because their policy relies on the partial convexity of the cycle pseudo-cost after fixing one parameter. This approach is not possible in the cyclic demand case. Furthermore, their technique of waiting for demand to accumulate sufficiently to in some sense "restart" works only for i.i.d. demand.

There is a small amount of online inventory literature that does not model demands as i.i.d. In Chen (2019), the demand distribution is allowed to change $\mathcal{O}(\log T)$ times; however, cyclic demands require the demand distribution to change $\Theta(T)$ times. Huh and Rusmevichientong (2014) study the cyclic demand problem for only the episodic model and have exponential regret dependence on the cycle length H, while our policies only have polynomial dependence. Their regret dependence on T for the episodic model is the same as ours for the episodic model (up to a $\log T$ factor). They do not have a result for the non-discarding case. In Chen (2019), the demand distribution is allowed to change $\mathcal{O}(\log T)$ times; however, cyclic demands require the demand distribution to change $\Theta(T)$ times. Cheung et al. (2020) studies non-stationary RL with the inventory problem as an application. They study a non-stationary environment in which the change in the demand distribution is limited by a given *variation budget*. In our setting, demand distributions are cyclic with no other assumptions on the distributions; hence arbitrarily large variation occurs within a cycle, but no variation (in terms of distributions) occurs across cycles. Therefore, their work is not applicable to the cyclic demands setting either.

2.3.2 Q-Learning Literature

To the best of our knowledge, our chapter is the first to apply *Q-learning* to inventory management with theoretical guarantees. Q-learning is a popular model-free

Table 2.1: Comparisons of Q-learning-based algorithms on applicable episodic MDPs.

Algorithms	**Regret**	**Time**	**Space**
Q-learning with optimism bonus in Jin et al. (2018)	$\tilde{\mathcal{O}}(\sqrt{H^3SAT})$	$\mathcal{O}(T)$	$\mathcal{O}(SAH)$
Aggregated Q-learning in Dong et al. (2019)	$\tilde{\mathcal{O}}(\sqrt{H^4MT}+\epsilon T)$	$\mathcal{O}(MAT)$	$\mathcal{O}(MH)$
FQL (our algorithm)	$\tilde{\mathcal{O}}(\sqrt{H^4T})$	$\mathcal{O}(SAT)$	$\mathcal{O}(SAH)$
HQL (our algorithm)	$\tilde{\mathcal{O}}(\sqrt{H^6T})$	$\mathcal{O}(SAT)$	$\mathcal{O}(SAH)$

Here M is the cardinality of the aggregate state-action space; ϵ is the largest difference between any pair of optimal state-action values associated with a common aggregate state-action pair. H, A, S, T are as defined in Section 2.2.1.

reinforcement learning method that does not require estimating the huge transition matrix in a large MDP (see Watkins and Dayan (1992)). The work most relevant to our chapter is Jin et al. (2018), which proves the optimality of Q-learning with optimism bonuses in tabular MDPs. Dong et al. (2019) improves upon Jin et al. (2018) when a good aggregation of the state-action pairs is given beforehand. Our algorithms substantially improve the regret bounds for the subset of problems that have full feedback/one-sided feedback (see Table 2.1).

Because our regret bounds are unaffected by the cardinality of the state and action space, our Q-learning algorithms can handle a large state-action space and even a continuous state-action space in some cases, which is the conventional setting in inventory problems. Meanwhile, if we discretize the state-action space optimally for Jin et al. (2018) and Dong et al. (2019), then applying Jin et al. (2018) to the simplest episodic backlogging inventory model gives a regret bound of $\mathcal{O}(T^{3/4}\sqrt{\log T})$. Applying Dong et al. (2019) with optimized aggregation yields us $\mathcal{O}(T^{2/3}\sqrt{\log T})$. In comparison, our algorithms HQL and FQL still have a regret bound of $\mathcal{O}(T^{1/2}\sqrt{\log T})$ after discretization because our regret bounds are independent of the cardinality of the state-action space. See detailed derivations in Appendix A.2.

This chapter is not concerned with optimizing the space needed to store Q, V-values, especially since the space we need does not exceed existing Q-learning algorithms. However, if storage space is of concern, then adaptively discretizing the state-action space could potentially help reduce the space needed to store Q, V-values. See for example Sinclair et al. (2019) on Q-learning with adaptive discretization of continuous space.

2.3.3 Rich Feedback Literature

Rich feedback structures have been studied in the bandit setting. For example, Zhao and Chen (2019) study bandit learning in problems with one-sided feedback. However, our episodic MDP setting presents new challenges. Our algorithm *HQL* can solve the setting in Zhao and Chen (2019) as a special case. Yuan et al. (2021) utilizes the one-sided feedback for an i.i.d. demand problem and is also located in the bandit setting.

The recent work of Dann et al. (2020) studies the regret of their reinforcement learning algorithm in general feedback graphs. However, they obtain worse regret bounds than those in our chapter on the families of problems that we consider. Their result matches the regret bound we obtain for the full-feedback setting, but it cannot recover our regret bound for our problems with the one-sided-feedback structure. The two works are largely complementary.

It is natural to associate our setting with the contextual bandit setting, where we think of each time step in an episode and each state as a different context. In particular, Balseiro et al. (2018) study *cross-learning* for contextual bandits. However, in our episodic MDP, the action chosen determines the next state, and the state, in turn, determines which actions are feasible. This dependence of the feasible action set on the state results in a genuinely sequential problem, so it fits well in the MDP framework and is not subsumed in contextual bandits. In comparison, for contextual bandits, the contexts (which correspond to states in our model) must be chosen adversarially at the beginning of the game or drawn independently from some distribution in each round. In other words, the context cannot depend on the actions chosen.

The full-feedback setting shares similarities with the generative model (Sidford et al. (2018)). However, the generative model is a strong oracle that can query any state-action transitions, while the full-feedback model can only query for a time step after having chosen an action from the feasible set based on the current state while accumulating regret.

Note that our definition of full feedback is different from the *full information feedback* in the non-stationary MDP literature such as Abbasi-Yadkori et al. (2013). In these works, the feedback allows observation of the complete transition probability for the next state, which includes the randomness that we refer to as "environmental randomness". By contrast, we are able to observe only one realized sample of the environmental randomness. In the inventory management case, this means that they assume that they would be able to observe the demand distribution as well, while we assume we are able to observe only one sample from the (partial) demand distribution for that time step. In inventory applications, it would be unrealistic to assume that we can observe the demand distributions.

2.4 Algorithm for the Episodic Lost-Sales Model

To utilize the one-sided-feedback structure, we want to choose high base-stock levels because they allow us to observe more information–this is exploration. On the other hand, we accrue regret as we choose each action. A higher base-stock level might lead to a higher starting inventory for the next time period and thus a smaller feasible action set for the next time period; therefore, exploration could negatively impact our exploitation in future periods. Our Algorithm 1 *Elimination-Based Half-Q-Learning Algorithm (HQL)* is designed in line with the key design principle of an effective learning algorithm: to delicately balance the trade-off between exploration and exploitation.

Define constants $\alpha_t = \frac{H+1}{H+t}, t \in \{1, 2, \ldots, T\}$. We define CB_1 (confidence bound 1) to be $\frac{8}{\sqrt{k-1}}(\sqrt{H^5 \iota})$, where $\iota = 9\log(AT)$. For notation, we use $\tilde{r}_{h,h'}$ to denote the cumulative reward from step h to step h'. We use $x'_{h+1}(x, y, \tilde{D}_h^k)$ to denote the (hypothetical) next inventory level given x, y and $\tilde{D}_h^k$ at time step h, where $\tilde{D}_h^k$ denotes the realized demand at step h of episode k. Similarly, $x'_{\tau_h^k(x,y)}()$ denotes the (hypothetical) inventory level outcome at time step $\tau_h^k(x, y)$ given the realized partial demand samples between time step h and time step $\tau_h^k(x, y)$. The notation $\tau_h^k(x, y)$ will be defined below.

Main Idea of *HQL*: At any episode k, we keep a "running set" A_h^k of all the actions that are possibly the best action for step h. We would like to continue observing the rewards and transitions for all actions that are still in the running set to obtain increasingly accurate estimates of these possibly best actions while keeping the regret of our actions small. To maximize the utility of the lower-sided feedback, we always select the largest action in A_h^k, letting us observe the most feedback. Sometimes, we might have so much inventory that we cannot choose from A_h^k; in that case, we order no new inventory. It follows by backwards induction that all Q, V-value functions for a time step are concave in the inventory level; therefore, ordering no inventory is with high probability the optimal action in this state and is larger than the largest action in A_h^k, which allows us to still observe the alternative

ALGORITHM 1: Elimination-Based Half-Q-Learning

```
Initialization: $Q_h(y) \leftarrow H, \forall (y,h) \in \mathcal{A} \times [H]$;   $A_h^0 \leftarrow \mathcal{A}, \forall h \in [H]$;
$A_{H+1}^k \leftarrow \mathcal{A}, \forall k \in [K]$;
for episode $k = 1, \ldots, K$ do
  Initiate empty partial-demand list $\mathbf{D}_k = [\,]$;
  for time step $h = 1, \ldots, H$ do
    if $x_h^k < \max\{A_h^k\}$ then
      Set base-stock level $y_h^k \leftarrow \max\{A_h^k\}$;
    else
      Order no new inventory $y_h^k \leftarrow x_h^k$;
    end if
    Observe realized partial demand $\min(y_h^k, \tilde{D}_h^k)$ and append to $\mathbf{D}_k$;
    Update $x_{h+1}^k \leftarrow x'_{h+1}(y_h^k, \min(y_h^k, \tilde{D}_h^k))$;
  end for
  for time step $h = H, \ldots, 1$ do
    for action $y' \in A_h^k$ do
      Simulate trajectory $x'_{h+1}, x'_{h+2}, \ldots, x'_{\tau_h^k(x,y)}$ as if we had chosen action $y'$ at
      step $h$ using stored list $\mathbf{D}_k$ until we find the next stopping time $\tau_h^k(x, y')$;
      Update $Q_h(y') \leftarrow (1-\alpha_k)Q_h(y') + \alpha_k[\tilde{r}_{h,\tau_h^k(x,y)} + V_{\tau_h^k(x,y)}(x'_{\tau_h^k}(x_h^k, y', \mathbf{D}_k))]$;
    end for
    Update $y_h^{k*} \leftarrow \arg\max_{y \in A_h^k} Q_h(y)$; Update $V_h(x) \leftarrow \max_{\text{feasible y given x}} Q_h(y)$;
    Update $A_h^{k+1} \leftarrow \{y \in A_h^k : |Q_h(y_h^{k*}) - Q_h(y)| \le CB_1\}$;
  end for
end for
```

rewards and transitions for all actions in the running set.

During each episode k, we act in real-time and keep track of the realized environmental randomness that we can observe, which is the realized partial demand $\min(y_h^k, D_h^k)$. At the end of the episode, we simulate the trajectories as if we had taken each action in A_h^k for $h = 1, \ldots, H$, and update the corresponding Q, V-value functions to shrink the running sets using confidence intervals. While we simulate the trajectories, we use the notion of a stopping time τ, where $\tau_h^k(x, y)$ denotes the next time the starting inventory allows the retailer to choose from a running set again in episode k (meaning that inventory is sufficiently low), starting from time step h, state x and action y. The stopping times are important because we have frequent updates to the Q, V-values only for actions in the running sets. When a stopping time is reached, the algorithm has a well-estimated value function at the current state. This value function is then used to update previous-state value functions using a delayed form of the Bellman equation. Note that the last stopping time τ for every episode is always $H+1$, because A^k_{H+1} is defined to be the entire action space.

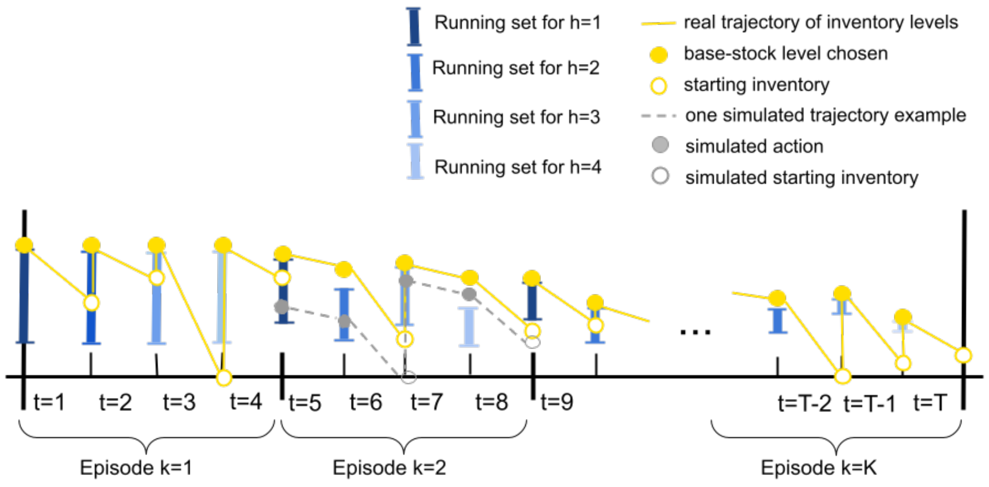


Figure 2-1: Application of *HQL* for an example episodic lost-sales problem.

Figure 2-1 illustrates *HQL*'s behavior for an example episodic lost-sales problem where episode length $H = 4$. The yellow lines represent the real trajectory of the inventory level. The gray dashed lines represent one of the simulated trajectories that *HQL* produces at the end of Episode 2 to update the Q, V-value functions and the running sets. For the simulated trajectory between $t = 7$ and $t = 9$, we can choose from the running set again only after Episode 2 ends (after t=8). Therefore,

$\tau_3^2 = H + 1 = 5$ by definition of A_{H+1}^k.

2.4.1 Regret Upper Bound of *HQL*

Let OPT denote the optimal policy for the episodic lost-sales model that knows the demand distributions a priori.

Theorem 1 *The total expected regret of* HQL *against* OPT *is*

$$\mathcal{O}\big(\sqrt{T\log T}\cdot H^3 M \max(|o_h|,|p_h|)\big)$$

for the episodic lost-sales model with zero lead time.

We briefly outline the proof of Theorem 1 here. The detailed proof can be found in Appendix A.1. To remove the regret dependence on the state and action space, our analysis incorporates the one-sided feedback into the analysis from Jin et al. (2018) and applies additional regret analysis techniques such as *shortfall decomposition* in Sutton and Barto (2018). To remove the regret dependence on the cardinality of the state-action space in the one-sided-feedback setting, we cannot adopt the main analysis in Jin et al. (2018). In comparison, when we discuss the multi-product backlogged model in Section 3.3, the presence of full feedback will allow us to more directly adopt the main techniques in Jin et al. (2018).

We first define a sequence of positive real values $\{\delta_h\}_{h=1}^{H+1}$ via the following backwards recursion. This sequence of positive values is useful in a proof by induction to give performance guarantees on *HQL*'s actions inside and outside the running sets.

$$\delta_h = H + (1+1/H)\delta_{h+1} + c\sqrt{H^3\iota}, \forall h \in [H],$$
$$\delta_{H+1} = 0$$

for some constant c. We know $\{\delta_h\}_{h=1}^{H+1}$ is a decreasing sequence, and $\delta_h \leq 4\sqrt{H^5\iota}$ (Appendix A.1.5). The following Lemma 1 gives performance guarantees on *HQL*'s actions inside and outside the running sets. The proof is provided in Appendix A.1.8.

Lemma 1 *For any $(h,k) \in [H] \times [K]$:*

1. *$\{\delta_h\}_{h=1}^{H}$ is a sequence of values that satisfy*

$$\max_{y \in A_h^k} |(Q_h^k - Q_h^*)(y)| \leq \delta_h / \sqrt{k-1} \tag{2.5}$$

   with high probability of at least $1 - 1/(AT)^5$.

2. *The optimal action y_h^* is in the running set A_h^k with probability of at least $1 - 1/(AT)^5$.*

3. *Any time* HQL *takes an action in A_h^k, the optimal Q-value of that action is within $3\delta_h/\sqrt{k-1}$ of the optimal Q-value of the optimal policy's action, with probability of at least $1 - 2/(AT)^5$.*

4. *Any time* HQL *cannot choose from A_h^k, its action to order no new inventory is the optimal action with probability of at least $1 - 1/(AT)^5$.*

The natural next step is to partition the time steps $h = 1, \ldots, H$ in each episode k into two sets, Γ_A^k and Γ_B^k, where Γ_A^k contains all the steps h where we are able to choose from the running set, and Γ_B^k contains all the steps h where we are unable to do so. Therefore, $\Gamma_A^k \sqcup \Gamma_B^k = [H], \forall k \in [K]$. Then we bound the per-episode regret by bounding over Γ_A^k and Γ_B^k. The difference between the expected total reward of

HQL and that of the optimal policy π^* is:

$$\begin{aligned}\text{Regret}_{MDP}(K) &= \left(V_1^* - V_1^{\pi_1}\right)(x_1^1) + \sum_{k=2}^{K}\left(V_1^* - V_1^{\pi_k}\right)(x_1^k) \\ &\leq H + \sum_{k=2}^{K}\Big(\sum_{h\in\Gamma_B^k}^{H}\frac{H}{A^5T^5} + \sum_{h\in\Gamma_A^k}\frac{\delta_h}{\sqrt{k-1}} + \sum_{h\in\Gamma_A^k}\frac{H}{A^5T^5}\Big) \\ &\leq \sum_{k=2}^{K}\frac{\mathcal{O}(\sqrt{H^7\iota})}{\sqrt{k-1}} \leq \mathcal{O}(H^3\sqrt{T\iota}).\end{aligned}$$

We have obtained an upper bound on the total regret of HQL against the optimal policy π^* on the MDP, which is denoted by Regret_{MDP}. Now we bound the additional regret incurred by discretization, denoted by Regret_{gap}:

Lemma 2 *The cumulative regret of the optimal policy π^* on the MDP against* OPT *(the optimal policy on the original problem) is $\mathcal{O}(\frac{M}{K})$, where we recall $K = \Theta(T)$.*

It follows that the total expected regret of HQL against OPT is

$$\begin{aligned}\text{Regret}_{total}(K) &= \text{Regret}_{MDP}(K) + \text{Regret}_{gap}(K) \\ &= \mathcal{O}\Big(H^3\sqrt{T\iota} + M/K\Big) = \mathcal{O}\Big(H^3\sqrt{T\log T}\Big).\end{aligned}$$

Finally, we multiply $\text{Regret}_{total}(K)$ by the factor $\mathcal{O}\big(M\cdot\max(|o_h|,|p_h|)\big)$ because we previously scaled the reward in Section 2.2.1. This implies a regret upper bound of

$$\mathcal{O}\big(H^3M\cdot\max(|o_h|,|p_h|)\sqrt{T\log T}\big).$$

2.4.2 Regret Lower Bound

We prove a regret lower bound for the episodic lost-sales model with zero lead time by constructing an example that can be reduced to a bandit problem.

Theorem 2 *The regret of any (randomized or deterministic) policy against* OPT *for the episodic lost-sales model with zero lead time is lower-bounded by*

$$\Omega\big(\sqrt{HT}M\max(|o_h|,|p_h|)\big).$$

Proof: We prove by constructing an instance. Even though the model is a lost-sales model, we assume that the algorithm is given the knowledge of what the demand is after it is realized for each time step. This only makes the problem easier and gives us a stronger lower bound.

Let $\eta > 0$ be a small absolute constant. Suppose for any step h in an episode that the demand distribution is $h + 100$ units w.p. $0.5 \pm \frac{\eta}{\sqrt{K}}$, and $h + 200$ units w.p. $0.5 \mp \frac{\eta}{\sqrt{K}}$. This difference between these two probabilities is not constant since $\frac{1}{\sqrt{K}} = \Theta(\frac{1}{\sqrt{T}}) \to 0$. Suppose the unit holding cost and the unit lost-sales penalty are the same, and suppose we are provided with the correct prior for the demand distribution as specified above.

All actions larger than $h + 200$ units and actions smaller than $h + 100$ units are worse than these two actions. Therefore, we can assume the base-stock levels are at most $h+200$. Note that after satisfying the demand of a previous period, the leftover inventory will be smaller than $h + 100$ units. Therefore, these two base-stock levels are always feasible.

The best base-stock level is one of these two actions ($h + 100$ units and $h + 200$ units) because the median demand is the optimal base-stock level. Therefore, the goal of any learning algorithm is to learn what the median demand is. If the algorithm uses a base-stock level $h + 100 + 100p$ for $p \in [0, 1]$, this is equivalent to choosing $h+100$ with probability $1-p$ and $h+200$ with probability p. Therefore, at each step h, the algorithm is solving a standard bandit learning problem on two actions. It is a well-known result that in this case, each step h will incur at least a $\Omega(\sqrt{K})$ regret across the K episodes. Specifically, at any step of any episode, the probability of any algorithm choosing the wrong action is lower-bounded by $\frac{1}{12}$ - roughly speaking, this is because the demand distributions are within $O(1/\sqrt{K})$, so $O(K)$ samples are insufficient to distinguish (with high probability) between them. This intuition can be formalized using the Kullback-Leibler divergence to quantify the information gained per-sample; details can be found as (Slivkins, 2019, Corollary 2.9). (Note that our algorithm receives twice as much feedback as a usual bandit algorithm; this only changes T to $2T$ in the context of (Slivkins, 2019, Corollary 2.9), which can be pushed onto the choice of small η above.)

The conclusion is that at each time step, the algorithm incurs at least $\Omega(\frac{1}{12\sqrt{K}})$ expected regret. This regret at step h across the K episodes sums up to $\Omega(\sqrt{K})$ expected regret. Since there are H time steps with demand distributions independent from each other, the total regret of this example is lower bounded by $\Omega(H\sqrt{K}) = \Omega(\sqrt{HT})$. Then we put back the $\Theta\big(M \cdot \max(|o_h|, |p_h|)\big)$ factor because in the preliminaries we scaled the costs down to have the reward for each step bounded by 1. $\square$

This regret lower bound shows that our algorithm *HQL* is optimal in terms of T dependence for the episodic lost-sales model with cyclic demands when the lead time is zero.

Chapter 3

Reinforcement Learning for Non-Discarding Inventory Management with Unknown Cyclic Demands

3.1 From Episodic to Non-Discarding: Bandits atop Reinforcement Learning

To go from the episodic inventory models to the non-discarding inventory models, we propose an algorithm, *Meta-HQL* that builds a bandit learning algorithm on top to govern multiple copies of the previous policy *HQL*.

Our key observation here is that if the first time step of each cycle happens to have the highest base-stock policy, then given zero lead time, as long as we set the purchasing cost and discarding cost to be zero, discarding or not at the end of the last cycle does not change the cost. Note that the purchasing cost and the discarding cost can both be set to zero. The former is done without loss of generality because of a standard reduction, while the latter is because there is no discarding cost allowed in the non-discarding model; thus, we can set an arbitrary discarding cost for our hypothetical algorithm that allows discarding at the end of the cycles.

We state this insight more formally in the following proposition.

Proposition 1 (**Key Observation 2**) *Consider a sequence of periodic demand distributions $D_1, \ldots, D_H, D_1, \ldots, D_H, \ldots$. The infinite horizon problem has an optimal solution given by base-stock levels $B_1, B_2, \ldots, B_H, B_1, \ldots, B_H, \ldots$ for some constants $B_1, B_2, \ldots, B_H$. Furthermore, if time step $h_{\max} \in \arg\max_{i\in[H]} B_i$, then the episodic problem with the sequence of demands being $D_{h_{\max}}, \ldots, D_H, D_1, \ldots, D_{h_{\max}-1}, \ldots$ (with indices modulo H) has an optimal solution given by the same base-stock levels*

$$B_{h_{\max}}, \ldots, B_H, B_1, \ldots, B_{h_{\max}-1}, \ldots.$$

The proof is obtained by combining and adapting several results in Zipkin (1989) and is presented in Appendix A.3. The intuition is that if we want to choose a higher optimal base-stock level in the second time step than in the first time step, then either the leftover inventory from the first time step can be discarded and we order replenishment for the second time step or the leftover inventory can carry over to the second time step and we order less replenishment to make up the difference. Recall the purchasing cost and discarding cost are set to zero, so these two scenarios are equivalent in cost.

We prove in Appendix A.3 that the regret of the optimal policy for the infinite-horizon problem, when applied to the finite-horizon problem, is upper-bounded by a constant additive difference with respect to the regret of the optimal policy for the finite-horizon problem. Then Proposition 1 implies that, hypothetically, for the finite-horizon problem, if the retailer knows a priori which time step $h_{\max}$ has (one of) the largest optimal base-stock levels in a cycle, then she could apply an $h_{\max}$-shifted copy of *HQL* to the non-discarding model to learn the optimal base-stock levels. However, we assumes that the retailer has no information of the demand distributions beforehand (at least until we discuss incorporating *expert advice* in Section 3.2). Thus, the retailer must learn what $h_{\max}$ is and what the optimal base-stock levels are simultaneously. Both pieces of knowledge help inform each other and require each other. *Meta-HQL* balances the exploration-exploitation trade-off so that both pieces of information can be learned together.

Main idea of *Meta-HQL*: *Meta-HQL* treats the non-discarding problem roughly as a bandit problem with H arms. Each arm $w = 1, \dots, H$ corresponds to a *w-shifted* copy of the original episodic problem. A *w-shifted* problem means that each cycle begins at step w and ends at step $w - 1$. For example, if a cycle is a week that starts on Monday, then a 5-shifted problem means that now we consider a week to start on Friday and end on the next Thursday. When we pull an arm w, it means that we apply the corresponding w-shifted version of *HQL* for an episode (H time periods). The *w-shifted HQL* is defined as a version of *HQL* that treats time step w as the beginning of an episode. In addition, the *w-shifted HQL* trims the running sets of the other time steps of each episode such that the upper bound of the running sets of the other time steps is never higher than the upper bound of the running set for step w. Note that this trimming does not affect the performance of the *w-shifted HQL* in the case where w is correctly identified as a time step with the highest optimal base-stock level. For the incorrect arms w where w in fact has a low optimal base-stock level, the trimming might cause the per-episode reward to decrease further.

How much time is spent exploring each arm is controlled by a *phase counter* j. At each phase j, we maintain an *arm set* W_j that contains all the remaining arms that have not been eliminated by the time the phase counter reaches j. We begin with $j = 0$ and no arms eliminated: $W_1 = [H] := \{1, \ldots, H\}$. When the phase counter reaches j, we play each remaining arm in W_j for $t_j = 2^j$ more episodes. Once we finish iterating through the remaining arms in W_j in this way, we enter the next phase where the phase counter is increased by 1. We eliminate an arm from the arm set after the j-th phase if its estimated per-episode reward is much worse than that of the best performing arm. We continue this process until the horizon ends.

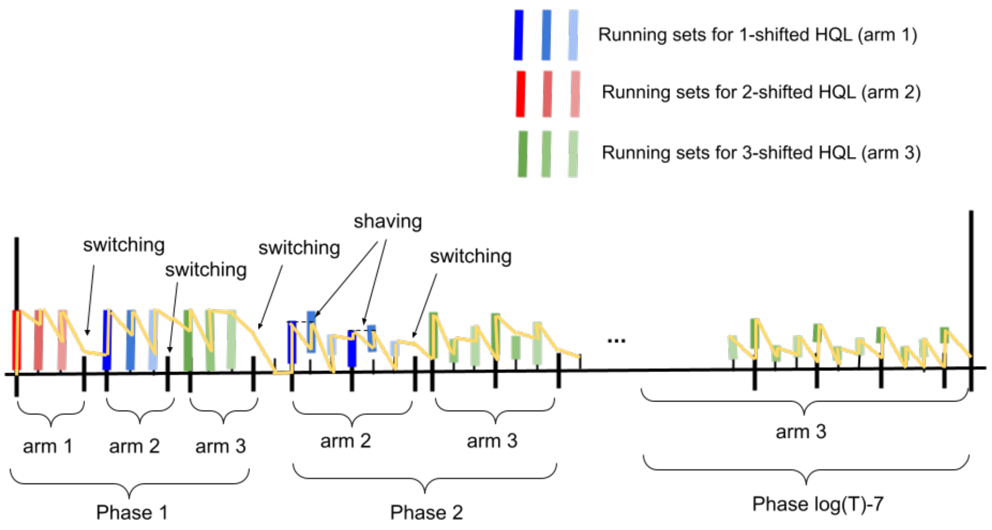

Running sets for 1-shifted HQL (arm 1)
Running sets for 2-shifted HQL (arm 2)
Running sets for 3-shifted HQL (arm 3)
switching
switching
switching
shaving
switching
...
arm 1
arm 2
arm 3
arm 2
arm 3
arm 3
Phase 1
Phase 2
Phase log(T)-7

Figure 3-1: Example of *Meta-HQL* for a non-discarding lost-sales problem.

Switching between arms: To switch from playing an arm $w \in W_j$ to another arm w', at this time we have already played arm w for 2^j episodes, which finishes with time step $w-1$ of some episode k. Playing the next remaining arm w' begins with time step w' of some episode $k' > k$. During the in-between time steps $w, w+1, \dots, w'$, we do not order any new inventory and let the inventory decrease until it is below the base-stock level that *w'-shifted HQL* would choose for the next w' time step.

In Algorithm 2, for each arm w, we keep a different set of Q-value $Q^{(w)}$ and V-value $V^{(w)}$ estimates. Let $\mathrm{CB}_2(K_j) = 2C_2 \cdot \frac{\sqrt{H^7 \log T}}{\sqrt{K_j}}$ denote the confidence bound used to update the arm set.

ALGORITHM 2: Meta-HQL

```
Initialization: $W_j = [H], \forall j = 1, \dots, \log \frac{T}{H}$.
$Q_h^{(w)}(x, y) \leftarrow H, \forall (w, y, h) \in W_t \times \mathcal{A} \times [H]$.
for $j = 0, 1, \dots,$ do
  for arm $w$ in $W_j$ do
    Once possible, play arm $w$ for $2^j$ rounds. Specifically, order no
    replenishment until the next time step $w$ when inventory is sufficiently low
    to apply the $w$-shifted HQL for $2^j \times H$ time periods. Update $Q_h^{(w)}$ and $V_h^{(w)}$
    according to the $w$-shifted HQL. No discarding inventory.
    In each round, eliminate all the arms $w$ in $W_t$ s.t.
    $V_1^{(w^*)} - V_1^{(w)} \geq 2 \cdot CB_2(K_j)$, where $w^* \leftarrow \arg\max_w V_1^{(w)}$ and $K_j$ is the total
    number of rounds played for arm $w$ so far.
  end for
end for
```

3.1.1 Regret Upper Bound of *Meta-HQL*

In Theorem 3, we state the main theorem of this chapter.

Theorem 3 Meta-HQL *achieves* $\mathcal{O}\big(H^{3.5} M \cdot \max(|o_h|, |p_h|) \sqrt{T \log T}\big)$ *regret for the non-discarding lost-sales model with zero lead time.*

The proof of Theorem 3 relies on a number of lemmas that bound the different parts of the regret that might occur in *Meta-HQL*: the part that is accumulated during the time periods when we are pulling some arm, and the part that occurs during the time periods when we are switching arms.

Lemma 3 *The total regret accumulated during switching between arms is upper-bounded by $\mathcal{O}(H\gamma \log^2 T)$.*

The regret bound in Lemma 3 is attributed to our design that the length of each phase j is 2^j, meaning that there are at most $H \log T$ switches between arms during the entire horizon.

Lemma 4 *With probability at least $1 - T^{-4}$,* Meta-HQL *never eliminates the best arm.*

This is because our estimated per-episode reward of each arm is close to the true optimal per-episode reward for that arm, especially for the correct arm that dictates the time step with the highest optimal base-stock level as the beginning of each episode. For incorrect arms, trimming causes the estimated per-episode reward to decrease further, which only helps matters.

The proof for Theorem 3 is provided in Appendix A.3.

3.1.2 Regret Lower Bound for the Non-Discarding Lost-Sales Model

We provide a regret lower bound for the non-discarding model by slightly modifying the instance we constructed for Theorem 2 such that the episodic example is equivalent to a non-discarding example by means of Proposition 1.

Corollary 1 *For any algorithm (randomized or deterministic), its expected regret against* OPT *for the non-discarding single-product lost-sales model with zero lead time is lower-bounded by $\Omega\big(\sqrt{HT}M \max(|o_h|, |p_h|)\big)$.*

Proof: We slightly modify the instance we constructed for Theorem 2.

Suppose for any step h in an episode, the demand distribution is $500\times(H-h)+100$ units with probability $0.5 \pm \frac{1}{\sqrt{K}}$, and $500 \times (H-h) + 200$ units with probability $0.5 \mp \frac{1}{\sqrt{K}}$. Note that the first time step in each episode has the highest optimal base-stock level. Therefore, by Proposition 1, we know that this problem is equivalent to an episodic lost-sales problem. Then the same reasoning and proof of Theorem 2 provide us with the same regret lower bound. □

Again, this regret lower bound shows that our algorithm *Meta-HQL* is optimal for the non-discarding lost-sales model with cyclic demands when the lead time is zero.

Remark 2 *In this chapter, we assume that the cycle length is given. Fortunately, the design of having a bandit learning algorithm as a meta algorithm on top of reinforcement learning algorithms bestows upon us additional power to give convenient solutions to some variations of the inventory models. One example is the variation where we can only make a fixed limited number Z of replenishment orders in a cycle.*

For the meta bandit problem, we include $H \times \binom{H}{Z}$ arms instead of H arms. Each arm represents a pair of a time step in the cycle and a choice of which Z time steps in a cycle to order. Recall that originally, each arm only represents a time step in the cycle. Then we have the same regret bound multiplied by $\binom{H}{Z}$.

Another example is the variation where the cycle lengh is not known a priori. Then it is often more important to get the cycle length right. Fortunately, the design of Meta-HQL *allows us to identify the cycle length simultaneously as we find the optimal policy. There are at least two ways for us to achieve this goal. Let $\overline{H}$ denote an upper bound on the true minimal cycle length. The value of $\overline{H}$ should be known, or we can choose a large constant to be the upper bound.*

An inefficient way: we know $\overline{H}!$ must be an integer multiple of the true cycle length. This means that $\overline{H}!$ can also be considered as the cycle length. (This is the same idea that was used in our Mimic-QL *algorithm). Then we will just treat $\overline{H}!$ as the cycle length H in* Meta-HQL*, and we immediately get the same regret bound where H is replaced by $\overline{H}!$.*

A more efficient way: in Meta-HQL*, for the bandit problem, we include $\overline{H}^2/2$ arms instead of H arms. Each arm represents a pair of a cycle length and a time step in the cycle. Recall that originally, each arm only represents a time step in the cycle. Then we have the same regret bound where H is replaced by $\overline{H}^2/2$.*

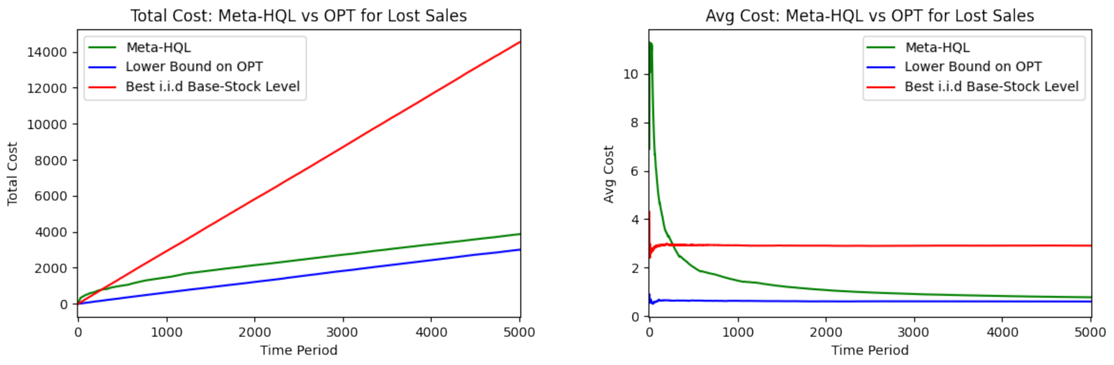
3.2 Numerical Experiments

For our numerical experiments, we use two dataset: one synthetic dataset and one publicly available sales dataset for 1,115 Rossmann stores from Jan 1, 2013 to July 31, 2015. Rossmann is one of the largest drugstore chains that operates over 4,000 stores across Europe. In 2019 Rossmann had more than €10 billion turnover in Germany, Poland, Hungary, the Czech Republic, Turkey, Albania, Kosovo and Spain. This dataset contains the daily sales of over 1,115 stores (Kaggle (2015)).

For the synthetic dataset We construct an instance of a non-discarding lost-sales problem with zero lead time, where *Meta-HQL* observes only the pseudo-costs while accumulating costs. The horizon is of length $T = 5000$, with demand cycle of length $H = 5$. The demands are randomly generated to have cyclic distributions: $\mathbf{D} = [9, 4, 1, 0, 6]$ with independent noise generated from a discrete uniform distribution Unif$[-1, 1]$, with step-size $\frac{1}{10}$. The holding cost and the lost-sales penalty are both 1 per unit. We consider base-stock levels in $[0, 10]$ with step-size $\frac{1}{10}$.

We evaluate the performance of *Meta-HQL* with respect to a stronger benchmark than the clairvoyant OPT. This benchmark is the optimal cyclic base-stock policy that has knowledge of all the realized demands a priori. We refer to this benchmark as "Lower Bound on OPT".

We also compare with the best offline policy in hindsight that models the demand distribution as i.i.d, which is a single base-stock level for all time periods. This policy is the policy that incurs the least inventory cost among all such policies, and can be obtained by gradient descent or binary search. We refer to this benchmark as "Best i.i.d. Base Stock Level". Note that this policy in principle should outperform all existing inventory methods in related literature, as they also model demand as i.i.d.



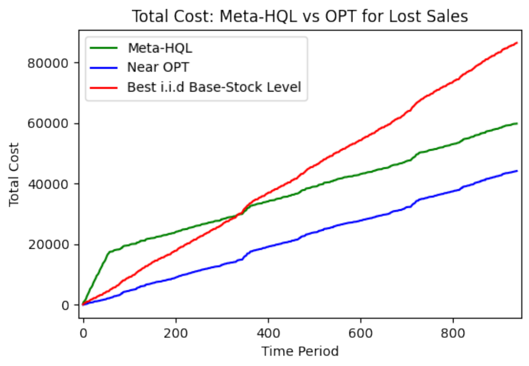
(a) Total Cost Comparison. (b) Average Cost Comparison.

Figure 3-2: Performance comparison of *Meta-HQL* and *OPT* on synthetic data.

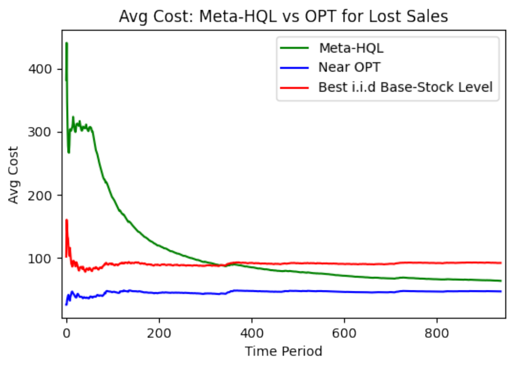
Figures 3-2a&3-2b show that the average cost of *Meta-HQL* per time step rapidly converges to the optimal average cost, and dramatically outperforms the best offline policy that models demand as i.i.d. instead of cyclic once *Meta-HQL* gets past the initial exploration-concentrated phase. The best offline policy that models i.i.d. demands is far off from the performance of the policies that model demand distributions as cyclic.

For the Rossmann sales dataset This dataset is publicly available on Kaggle (see Kaggle (2015)) and contains daily sales of 1,115 stores from Jan 1, 2013 to July 31, 2015. We use the sales data from one of the stores. The dataset contains the daily turnover and does not contain product information. We use the turnover amount for each day divided by 20€ as a proxy for the demand for that day.

With real data, it is more difficult to compute the offline optimal policy that is a cyclic base-stock policy. Therefore, we find a near-optimal cyclic base-stock policy, referred to as “Near OPT”. We evaluate the performance of *Meta-HQL* with respect to this near OPT benchmark, as well as the best offline policy that models the demand distributions as i.i.d.. The holding cost and the lost-sales penalty are both 1 per unit. For our policy *Meta-HQL*, we consider base-stock levels in the range of $[0, 500]$, with step size 1.



(a) Total Cost Comparison.



(b) Average Cost Comparison.

Figure 3-3: Performance comparison of *Meta-HQL* and *OPT* on real Rossmann sales data.

The relative behavior of the three policies on the real Rossmann sales dataset remains very similar to that on the synthetic dataset. Figures 3-3a&3-3b show that the average cost of *Meta-HQL* per time step rapidly converges to the near optimal average cost, and outperforms the best offline policy that models demand as i.i.d.

instead of cyclic once *Meta-HQL* gets past the initial exploration-concentrated phase.

A practitioner might be concerned about the performance of *Meta-HQL* at the beginning of the horizon. Note that the large costs incurred at the beginning of the horizon are caused by the unnecessarily large initial base-stock range (up to 500 units) that we ask *Meta-HQL* to explore. These large costs can be avoided by having a more modest set of initial base-stock range of inventory level.

For example, although unnecessary for the regret analysis, our policies allow the additional input of *expert advice* in case the retailer possesses (partial) knowledge of the demand distributions beforehand to further improve the performance of our algorithms in real applications.

Suppose the retailer has the following knowledge of the demand distributions in a cycle: the historical demand for Saturdays has never been larger than 247 units; the historical demand for Thursdays has never been larger than 427 units; the historical demand for Fridays has never been larger than 378 units. Then we can adjust the starting running sets in *Meta-HQL* to incorporate this mild expert advice.

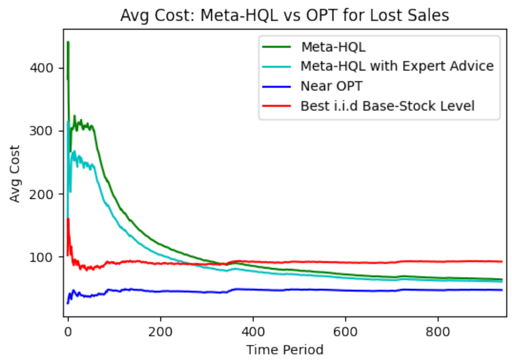


Figure 3-4: Performance of *Meta-HQL* with expert advice on real Rossmann sales data.

Figure 3-4 shows that with this mild expert advice, the performance of *Meta-*

HQL significantly improves, especially in the short term. This ability to incorporate expert advice is beneficial for real applications of our policies, and connects our online learning policies to offline learning.

3.3 Multi-Product Backlogging Models

In this section, we extend the results in the previous sections to multi-product backlogging models, where the extra demand is backlogged and not lost when it exceeds the on-hand inventory.

The retailer has N products, each with a different inventory. At the beginning of step h, the retailer reviews the starting inventory, which is a vector denoted by $\mathbf{x_h}$ that includes the current inventory and replenishment in the pipeline. The retailer orders replenishment at a purchasing cost of $c_{i,h}$ per unit of product i. If any replenishment is ordered, a fixed joint-ordering cost F_h is incurred. Let $\mathbf{y}_h$ denote the vector of ordered replenishment units for the n products at step h. A replenishment order arrives after a deterministic lead time, which can be product-dependent. For ease of presentation, we use a common lead time L for all products.

The retailer receives the replenishment $\mathbf{y}_{h-L}$ that was ordered L steps ago. This replenishment $\mathbf{y}_{h-L}$ plus the current inventory is the *on-hand* inventory at time step h, which the retailer uses to satisfy the demand in time step h. Let $\mathbf{I}_h \in \mathbb{R}^n$ denote the vector of the *on-hand* inventory of the n products at time step h. Let $D_{i,h}$ denote the random demand from unknown distribution $F_{i,h}$ that is realized during step h for product i. For each product i, the demand distributions are cyclic.

If demand exceeds the on-hand inventory of product i, unmet demand carries over to the next time step as negative inventory. The retailer pays a backlogging cost $b_{i,h} > 0$ for each unit of unmet demand for each product i. If for some product i, demand is less than the on-hand inventory, the retailer pays a holding cost $o_{i,h} > 0$ for each unit of leftover inventory. At the end of step h, the possibly negative remaining inventory and the updated replenishment in the pipeline become the starting inventory $\mathbf{x}_{h+1}$ for the next step.

If the application calls for it, this model can add in the constraints *order limits*, which requires that for any time step h, the replenishment for each product i ordered in that time step has to satisfy that $u_{h,i} \leq y_{h,i} \leq \ell_{i,h}$ for any i, h. The order limits can apply to combinations of products as well.

All the cost parameters can be time-varying as long as they are periodic in cycles of H time periods. For ease of presentation, we use the following cost function, but our proof remains valid if additional terms exist.

$$\text{Cost}_h = F_h \cdot \mathbf{1}\{\mathbf{y}_h \neq \mathbf{0}\} + \mathbf{c}_h^\top(\mathbf{y}_h) + o_h^\top(\mathbf{I}_h + \mathbf{y}_{h-L} - \mathbf{D}_h)^+ + b_h^\top(\mathbf{D}_h - \mathbf{I}_h - \mathbf{y}_{h-L})^+.$$

Now we discuss the difference between the *episodic* model and the *non-discarding* model.

- **Model 1: Episodic Multi-Product Backlogging**

  Similarly to the episodic lost-sale model:

  Assumption 1a: At the end of each cycle, excess inventory is discarded or salvaged at some arbitrary fixed price.

  In the case of salvaging, we subtract the salvaging revenue from the cost of the last step in a cycle so that Equation (3.3) holds for all $h \in [H]$.

  The preliminaries are similar to those in Section 2.2.1 and are thus relegated to Appendix A.5. We emphasize that we no longer simplify notation $Q(x, y)$ to $Q(x)$ because we consider general policies.

- **Model 2: Non-Discarding Multi-Product Backlogging**

  This model is consistent with inventory literature for non-perishable products: we never discard inventory. The leftover inventory at the end of a cycle always carries over to the next cycle. Instead of Assumption 1a for the episodic model, we have the following alternative Assumption 1b for the non-discarding model:

  Assumption 1b: The expected time needed to deplete 1 unit of inventory is at most γ.

  This is an often-used assumption in literature (see for example Agrawal and Jia (2022)). Note that our policy does not need to know γ. This assumption is used only in the analysis.

The objective is to minimize the expected total cost. Note that for both the episodic and the non-discarding multi-product backlogging models, the decision variables $\mathbf{y}_h$ for the multi-product backlogging model are not restricted to cyclic base-stock policies, because cyclic base-stock policies are not necessarily optimal for this model.

3.3.1 Algorithm for the Episodic Backlogging Model: *FQL*

Full-Q-Learning (*FQL*) is a simple variant of *HQL*, designed for the multi-product backlogging problem. This method allows for a more general policy space than *HQL*: our action space is no longer restricted to cyclic base-stock policies because they are no longer the optimal class of policies for the multi-product backlogging model. *FQL* achieves optimal $\mathcal{O}(\sqrt{T})$ regret because there is richer feedback in the backlogging model than in the lost-sales model.

Key Observation 3: at any time step h, after the retailer takes an action, once the demand is realized, we can deduce the reward and the leftover inventory for any other replenishment decision as well. This is the *full-feedback* structure, a stronger case of the one-sided-feedback structure.

ALGORITHM 3: Full-Q-Learning

```
Initialization: $Q_h(x, y) \leftarrow H, \forall (y, h) \in \mathcal{A} \times [H]$.
for episode $k = 1, \dots, K$ do
    Initiate empty demand list $\mathbf{D}_k = []$;
    for time step $h = 1, \dots, H$ do
        Order new inventory $y_h^k \leftarrow \min\{\underset{\text{feasible y given } x_h^k}{\arg\max} Q_h(x_h^k, y)\}$;
        Observe realized $\tilde{D}_h^k$ and append to $\mathbf{D}_k$; Update $x_{h+1}^k \leftarrow x'_{h+1}(x_h^k, y_h^k, \tilde{D}_h^k)$;
    end for
    for time step $h = H, \dots, 1$ do
        for state $x \in \mathcal{S}$ and action $y \in \mathcal{A}$ do
            Update $V_{h+1}(x'_{h+1}(x, y, \tilde{D}_h^k)) \leftarrow \max_{\text{feasible y'}} Q_{h+1}(x'_{h+1}(x, y, \tilde{D}_h^k), y')$;
            Update $Q_h(x, y) \leftarrow (1 - \alpha_k) Q_h(x, y) + \alpha_k \big[r_h(x, y) + V_{h+1}\big(x'_{h+1}(x, y, \tilde{D}_h^k)\big)\big]$;
        end for
    end for
end for
```

Theorem 4 FQL *achieves* $\mathcal{O}\Big(\sqrt{T \log T} \cdot H^2 n \max\big(M|o_h|, M|b_h|, |F|\big)\sqrt{n(L+1)}\Big)$ *regret against the optimal policy for the n-product episodic backlogging model with fixed joint-ordering cost F, lead time L and order limits.*

In the case of FQL, for the sake of continuity we use similar notations and analysis as in Jin et al. (2018) but adapted to our full-feedback setting. The proof is simpler than the proof for *HQL* and is done by making a few moderate modifications to the analysis in Jin et al. (2018); see Appendix A.6. Because *FQL* leverages the full feedback, it shrinks the concentration bounds much faster than do existing algorithms, resulting in a significantly lower regret bound that is unaffected by the state-action space.

The regret lower bound (Theorem 2) applies to the episodic backlogging model as well.

3.3.2 Algorithms for the Non-Discarding Backlogging Model

For the non-discarding single-product backlogging model with zero lead time, we can easily see that *Meta-HQL* applies to this model as well and achieves the same regret bound. However, we can do better by utilizing Key Observation 3–the full feedback of backlogging models again to simplify the process of determining the correct shift of cycles. We can use *FQL* instead of *HQL* for the shifted copies of algorithms that are governed by the top bandit learning algorithm. This method reduces the regret upper bound down by a factor of H. The proof is an easy modification of the proof of Theorem 3 and is thus omitted.

For the non-discarding multi-product backlogging model, we cannot apply *Meta-HQL* because the base-stock policies are not necessarily optimal for this very general model. We take a different approach by utilizing the following observation:

Key Observation 4: even though the cycle length of demand distributions is fixed, the cycle length used in the retailer's policy is a variable that can be optimized to morph the episodic models to non-discarding models. This process is possible because a sequence of demand distributions that is cyclic in H is also cyclic in any integer multiple of H.

To utilize Key Observation 4, consider an *intermediate MDP* that has J as the episode length and discards inventory after every J time steps, where J is an integer multiple of the true cycle H. Specifically, we take J to be the closest integer multiple of H to $T^{1/6}$ (see details in Appendix A.7). The number of cycles is $K = \frac{T}{J}$. We use an underline on notation related to the intermediate MDP. For example, we use $\underline{\text{OPT}}$ to denote the optimal policy for the intermediate MDP, while OPT is the optimal policy for the original non-discarding model. $\underline{\text{Cost}}$ denotes the total expected cost of a policy in the intermediate MDP. The rest of the setup is consistent with the original non-discarding problem: the fixed joint-ordering costs and the unit holding and backlogging costs of the intermediate MDP are the same as those in the non-discarding model. We set the discarding price for the hypothetical intermediate MDP to zero. Discarding at the end of a cycle can make $\underline{\text{Cost}}$ either higher or lower than the Cost of the original problem due to the positive lead time. Recall that our actions are not limited to base-stock policies. According to Key Observation 4, *FQL* applies to the intermediate MDP and achieves $\tilde{\mathcal{O}}(J^2\sqrt{T})$ regret with respect to $\underline{\text{OPT}}$.

We propose a policy *Mimic-QL* for the non-discarding multi-product backlogged model. *Mimic-QL* simulates *FQL* for the intermediate MDP on the side, and mimics *FQL* when possible. In Algorithm 4, $\underline{x}_h^k[i]$ denotes the state vector of inventory and replenishment of product i that *FQL* is in on the intermediate MDP at step h in episode k. We use $\underline{y}_h^k[i]$ to denote the amount of replenishment of product i that *FQL* would order given $\underline{x}_h^k$. For any vector v, we use function $\text{sum}(v)$ to denote the sum across all coordinates of v. We use $x'_{h+1}[i]()$ to denote the function of leftover inventory for the non-discarding model and $\underline{x}'_{h+1}[i]()$ to denote the function of leftover inventory for the intermediate MDP.

Main Idea: *Mimic-FQL* solves the non-discarding model by mimicking *FQL* for the intermediate MDP, which *Mimic-FQL* simulates on the side. *FQL* always starts an episode with less inventory than *Mimic-FQL* because of discarding. For each product i, as long as *FQL* has less total summed inventory of product i (including replenishment in the pipeline) than *Mimic-FQL*, *Mimic-FQL* does not order any

ALGORITHM 4: Mimic-FQL (MimicQL)

```
Initialization: $Q_h(y) \leftarrow H, \forall (y,h) \in \mathcal{A} \times [H]$;   $A_h^0 \leftarrow \mathcal{A}, \forall h \in [H]$;
$A_{H+1}^k \leftarrow \mathcal{A}, \forall k \in [K]$;
for $k = 1, \ldots, K$ do
  Initiate empty demand list $\mathbf{D}_k[i] = [\ ]$ for each product $i$;
  for $h = 1, \ldots, H$ do
    for product $i = 1, \ldots, n$ do
      if sum($x_h^k[i]$) $\leq$sum($\underline{x}_h^k[i]$) then
        Order new replenishment $y_h^k[i] \leftarrow$ sum($\underline{x}_h^k[i]$) $-$ sum($x_h^k[i]$);
      else
        Order no new replenishment for product $i$;
      end if
    end for
    Observe realized demand $\tilde{D}_h^k[i]$ for each product $i$ and append to $\mathbf{D}_k[i]$;
    Update $\underline{x}_{h+1}^k[i] \leftarrow \underline{x}'_{h+1}[i](x_h^k[i], y_h^k[i], \tilde{D}_h^k[i]), \forall i$;
    Update $x_{h+1}^k[i] \leftarrow x'_{h+1}[i](x_h^k[i], y_h^k[i], \tilde{D}_h^k[i]), \forall i$;
  end for
  Update the $Q$ values and $V$ values using $\mathbf{D}_k[i]$ in the same way as *FQL*.
end for
```

product i. The moment the total summed inventory level of product i of *FQL* becomes greater than or equal to that of *Mimic-FQL*, *Mimic-FQL* orders an amount of replenishment such that given the same demand for product i in that time step, the total summed inventory of product i of *Mimic-FQL* and *FQL* will be the same at the beginning of the next time step.

Mimic-FQL continues ordering in this manner for all products for at most L time periods after the beginning of each episode before the inventory vector and the replenishment pipeline vector of *Mimic-FQL* and *FQL* are synced. From this point on, *Mimic-FQL* completely follows the actions of the simulated *FQL* until the beginning of the next episode, where the inventory levels of *Mimic-FQL* and *FQL* might differ again because of discarding.

Theorem 5 MimicQL *achieves* $\tilde{\mathcal{O}}\left(H^{5/2}T^{5/6}\right)$ *regret for the non-discarding multi-product backlogging model with fixed joint-ordering cost, lead time and order limits.*

We provide the detailed proof of Theorem 5 in Appendix A.7.

3.4 Conclusion and Discussions

3.4.1 Full Feedback vs. One-Sided Feedback

We observe the full feedback structure in the multi-product backlogging model, and the one-sided feedback in the lost-sales model. In Appendix A.8 we give more formal definitions of the one-sided-feedback structure and the full-feedback structure.

When full feedback is present, for the episodic model, it is "easy" for us to design an efficient reinforcement learning algorithm that removes the regret dependence on the cardinality of the state-action space. There is no trade-off between optimizing our action and the amount of feedback we can observe. We only made small modifications in Q-learning algorithms to take advantage of the full feedback. We do not need optimism bonuses such as the *UCB*bonus (see Jin et al. (2018)) for exploration either. The regret bound for the episodic model is $\mathcal{O}(H^2\sqrt{T})$ without dependence on the state-action space.

In contrast, one-sided feedback does not guarantee much benefit. If the replenishment decisions are not chosen carefully, one-sided feedback might give very little additional information. There is an additional trade-off between ordering more to use the one-sided feedback and ordering less to save on costs. The existing machinery in reinforcement learning cannot readily take advantage of the one-sided feedback structure. That is the reason why we have to devise a new algorithm *HQL* that uses stopping times and running sets to fully take advantage of the one-sided feedback. The regret bound of *HQL* for the episodic lost-sales model is $\mathcal{O}(H^3\sqrt{T})$ without dependence on the state-action space.

Assuming our goal is to eliminate the regret dependence on the state-action space, then for the episodic models, full feedback makes the problem much simpler than one-sided feedback. Therefore, we can deal with very general complicated backlogging models, while for the lost-sales model, our algorithms can only deal with the simplest lost-sales model.

To transition from the episodic models to the non-discarding models, *Meta-HQL* uses the optimal policy structure for the simplest lost-sales model to obtain a regret bound of $\mathcal{O}(H^{3.5}\sqrt{T})$. However, this structure is not optimal for the very general multi-product backlogging model that we introduced, so we gave some other ideas and techniques that obtain suboptimal regret bounds for the non-discarding multi-product backlogging model. If not for our ambition to deal with very general multi-product backlogging model, *Meta-HQL* would solve the non-discarding single-product backlogging model with the same regret bound (with a simple modification of the cost function). However, we consider it more useful to the readers to discuss the general multi-product backlogging model, because the full feedback structure present in the problem has a lot of potential for future research on complex models.

Now we provide a few example problems that are important in operations research or finance and possess either the one-sided-feedback or the full-feedback structure.

Airline Overbooking Policy: Overbooking or overselling is the sale of a good or service in excess of actual supply. This is a common and legally allowed practice in the airline, hotel, and rental car industries in many countries, in which no-shows or cancellations frequently occur, allowing for significant additional revenue (Chatwin (1998)). Substantial compensation is offered to customers who are denied due to excessive overbooking even though they have a ticket. The goal of the agent is to navigate the trade-off between the additional revenue and the penalty. Often in reality, the demands and customer behavior (no-shows or cancellations) follow a strong weekly pattern. Specifically, the distributions in these industries for the weekdays differ significantly from the distributions for the weekends.

Using the airline industry as an example, the agent must decide the overbooking ratio that the airline will allow for each time period. Suppose the airline has 10 seats available every day. If the agent sets the overbooking ratio to be 10% for Monday, then the airline will allow at most 11 customers to purchase a seat for Monday. If a 12th customer arrives and wants to purchase a seat for Monday, that customer will be unsuccessful and the agent does not observe this additional demand: the demand is therefore censored.

The overbooking problem possesses the lower-sided feedback structure: once the agent chooses an overbooking ratio for a time period, the agent observes the part of the realized demand that does not exceed the allowed number of bookings, and the reward for this chosen overbooking ratio. At the same time, using the observed part of the demand, the agent can also deduce the reward for any overbooking ratio lower than the chosen one. The assumptions needed in Appendix A.8.2 are satisfied as well.

Online Second-Price Auctions: A second-price auction is an auction mechanism where the highest bidder receives the item and pays the second-highest bid as opposed to the highest bid. This is considered to be a more "ideal" auction mechanism than first-price auctions because it is *incentive-compatible*, meaning that it is in each bidder's best interest to bid truthfully according to their true valuation of an item. The online second-price auction is used in different markets for financial products and online ads placements.

We consider repeated online second-price auctions where the auctioneer also submits a bid at each time period to prevent the item from selling below what the auctioneer thinks it is worth. This bid by the auctioneer is also called *the reserve price*. The auctioneer must decide the reserve price for the same item in each round, and each bidder draws a value from its unknown distribution.

Each bidder submits a bid only if its valuation of the item is not lower than the reserve price. The auctioneer observes the bids, gives the item to the highest bidder, if any, and collects the second-highest bid price (including the reserve price) as profit. In this case, we have the higher-sided-feedback structure. Once the auctioneer announces the reserve price and bids are submitted, the auctioneer can deduce what bids, and thus what profit, she would have received if she had set any reserve price higher than the announced reserve price. The assumptions needed for the one-sided-feedback setting in Section A.8.2 are satisfied as well. Note that the feasibility of the actions does not depend on the state in this model.

If the bidders submit their bids regardless of whether their values are lower than the reserve price, then this problem becomes a full-feedback problem because the auctioneer can now deduce the alternative profit for any reserve price.

Portfolio Management: is the classic problem of allocating a fixed sum of cash to a variety of financial instruments Markowitz (1952). In each time period, the manager collects the increase in the portfolio value as the reward and is penalized for any decrease.

We assume that the total volume of the portfolio is not sufficiently large to have a noticeable effect on the market prices of the financial instruments. This is a full-feedback problem because once the returns of all instruments become realized for that day, the manager can deduce what his reward would have been for all feasible portfolios.

The episodic MDP model is suitable when the returns of the financial instruments we consider have periodic or seasonal distributions, e.g. futures contracts.

3.4.2 Conclusion

We fill a theoretical gap in inventory theory for online inventory control with unknown cyclic stochastic demands. We design reinforcement learning algorithms that cater to the special structures of inventory models. Specifically, we construct a bandit learning algorithm on top of multiple copies of our reinforcement learning algorithms to achieve the optimal regret bound for the online lost-sales model under unknown cyclic stochastic demands.

Our algorithm works well for both synthetic data that are generated to have cyclic demand distributions and the real sales data of Rossmann drugstores. Our policy, *Meta-HQL*, drastically outperforms the best policy under the i.i.d. demand assumption and rapidly approaches the clairvoyant optimal policy.

We expect that future extensions of our work in customizing reinforcement learning and other advanced machine learning methods to more varied operations problems could be very fruitful.

Chapter 4

Online Assortment Optimization for Reusable Resources

4.1 Introduction

Assortment optimization is an important problem that arises in a broad set of applications including online advertising, recommendations and e-retailing. In these applications, the goal of the decision-maker is to select a subset of products from the available universe to offer to the user to maximize the expected revenue or reward. For any given subset S of offered products, the selection of the user depends on his or her random preference over the set of products including the no-purchase or exit option. We model this random selection using a *choice-model* that for any offer set S, specifies the probability, that the user selects product $j \in S \cup \{0\}$ (where 0 refers to the no-purchase or exit option). Several parametric choice models have been studied in the literature including multinomial logit (MNL) model Luce (1959); Plackett (1975); McFadden (1973), the nested logit model Williams (1977); McFadden et al. (1978); Davis et al. (2014); Gallego and Topaloglu (2014), Markov chain based model Blanchet et al. (2016) and the mixture of multinomial logit model McFadden and Train (2000) (see Train (2009); Kök et al. (2015); Berbeglia et al. (2018) for a detailed overview of these models).

In this chapter, we consider an online assortment problem where we are given n substitutable products with fixed capacities or inventories $c_1, \dots, c_n$. Users with different choice models arrive sequentially. For each user, the seller offers a subset S of the available products satisfying certain constraints, the user selects a random product $j \in S \cup \{0\}$ with probability given by his or her choice model and uses it for a random amount of time, $\tilde{t}_j$ and returns it to the platform, generating revenue $r_j(\tilde{t}_j)$ for the seller. The goal of the platform or the seller is to design a policy to offer assortments to the user so that the expected revenue is maximized.

Our main contribution is to show that a simple myopic policy (where we offer the myopically optimal assortment from the available products to each user) provides a good approximation for the problem. In particular, we show that the myopic policy is 1/2-competitive, i.e., the expected cumulative revenue of the myopic policy is at least half the expected revenue of the optimal policy with full information about the sequence of user preference models and the distribution of random usage times of all the products. In contrast, the myopic policy does not require any information about future arrivals or the distribution of random usage times. The analysis is based on a coupling argument that allows us to bound the expected revenue of the optimal algorithm in terms of the expected revenue of the myopic policy. We also consider the setting where usage time distributions can depend on the type of each user and show that in this more general case there is no online algorithm with a non-trivial competitive ratio guarantee.

This model fits the setting of classical online product allocation and revenue management. However, unlike traditional settings where the capacity or inventory of any product decreases permanently whenever user selects that product, the products are "reusable" in our setting, meaning that upon allocation they come back into the inventory after some period of time and may be allocated several times over the planning horizon. Such a setting arises commonly in many applications including cloud computing, physical storage, make-to-order service and other sharing econoy applications. For example, consider modern cloud platforms such as Amazon Web Services, Google Cloud, and Microsoft Azure. Among other services these platforms commonly support large scale data storage, a service that is widely used by online video platforms such as YouTube and Netflix. Given the large scale of these networks and huge volume of data, a given data file is stored in a subset servers that are part of the cloud. Thus, a typical user request for data can only be sent to the subset of servers with the required data. Additionally, each server can concurrently serve only a limited number of requests while meeting the stringent low latency requirements on such platforms. Our online assortment model captures this setting as a special case where products correspond to servers in the cloud and starting inventory corresponds to capacity of servers. Customers correspond to user requests that arrive sequentially over time and each customer must be irrevocably matched on arrival to at most one product out of a given subset that is revealed when the customer arrives. Interestingly, several recent works in the queuing theory study a similar setting with the objective of latency minimization on a bipartite networks, for instance Weng et al. (2020); Budhiraja et al. (2019); Mukherjee et al. (2018); Cruise et al. (2020). Our model ignores the queuing aspect and provides a complementary perspective from the point of view of maximizing the number of successful matches in a *loss system* (where the latency of accepted jobs is zero).

Settings similar to ours have previously been considered in the literature on online assortment planning for non-reusable products as well as reusable products. Golrezaei et al. (2014) consider the online assortment problem with fixed product capacities for the case of non-reusable products (i.e. inventory of a product decreases whenever any user selects that product). They give an inventory balancing based algorithm that is $(1 - 1/e)$-competitive for adversarial arrivals in the limit of capacities going to infinity. For the case of all capacities being equal to one, their algorithm is 1/2-competitive. Ma and Simchi-Levi (2020) consider a more general setting where the seller can make joint assortment and pricing decisions, and obtain guarantees with adversarial customer arrivals for the case of non-reusable products.

While the above results pertain to non-reusable resources, most closely related result to our setting of reusable resources is in the work of Rusmevichientong et al. (2020). They consider this problem in the setting of stochastic customer arrivals where the distribution of user types is known in advance. Using this distributional knowledge one can write the optimal algorithm in this case as a dynamic program (DP), but this DP suffers from the curse of dimensionality. Rusmevichientong et al. (2020) give an algorithm based on approximate dynamic programming and show that it is a 1/2-approximation to the optimal DP for the problem. Earlier, Dickerson et al. (2018) considered the problem of online matching (instead of assortments) when resources are reusable and the distribution of customer types is known. They proposed a simulation and LP based approach and showed that it is 1/2 competitive against an offline LP benchmark (which is a stronger guarantee).

In contrast, we assume no advance knowledge of the user type distribution and consider an adversarial model for the sequence of user types. Recall, user type refers to the choice model of the user, which is revealed to the platform when the user arrives. Let ϕ^z be the choice model for user type z and $\phi^z(i, S)$ the probability that user type z selects product i given assortment S. We make the following assumptions about choice probabilities and usage time distributions.

Assumption 1 *For any user type z, assortment $S \subseteq T \in \mathcal{S}$ and $i \in S$, we have $\phi^z(i, S) \geq \phi^z(i, T)$.*

This is a mild assumption and without much loss of generality. In fact, all random utility based choice models including multinomial logit (MNL), nested logit and mixture of MNLs satisfy Assumption 1.

Assumption 2 *For every product j, usage time distribution depends only on j and not on the user type.*

For settings where this assumption does not hold we show that it is impossible to obtain any constant factor competitive algorithm for adversarial arrivals. In general, Assumption 2 is reasonable in various settings where the choice of the product depends on the user type, while the usage time depends only on the product. In the setting of cloud computing this translates to the assumption that the time taken to fulfill a user request depends primarily on the characteristics (processing power, memory) of the server that a request is assigned to. While imperfect, this assumption is commonly employed in related literature (Weng et al., 2020; Budhiraja et al., 2019; Mukherjee et al., 2018; Cruise et al., 2020). As another example, consider a make-to-order setting where user selects a product from the offered assortment. Once the user makes the selection, a dedicated machine (resource) makes the product for the user. In such a setting, the busy time of the machine (resource) depends only on the product. Make-to-order settings also involve non-reusable resources in the form of raw materials. However, in a typical setting the machines are the bottleneck resources as they are often much more expensive than the raw materials. It is also worth noting that in many scenarios making a product may require the use of several machines either in parallel or in a given sequence. This presents new challenges that are not captured in the model we consider here and could be an interesting direction for future work.

The revenue to the seller, $\boldsymbol{r}_j(\tilde{t}_j)$ when product j is used for some random time, $\tilde{t}_j$, could be a general function of the usage time. In particular, we can model fixed revenue for every use, as well as revenue which is an affine function of usage time with fixed component and per-unit usage time component. Let r_j denote the expected revenue of product j where the expectation is taken over the random usage time of product j, i.e.,

$$r_j = \mathbb{E}_{t_j \sim F_j}[\boldsymbol{r}_j(t_j)],$$

where F_j is the cdf of usage time distribution of product j.

Mathematical Formulation We are given n substitutable products with *reusable* capacities $c_1, \ldots, c_n \in \mathbb{N}$. Each product i has a price $r_i \in \mathbb{R}^+$. In each period t over a horizon of length T, a customer, denoted by customer t, arrives to our platform. The customer's choice model $\boldsymbol{\phi}^t$ becomes known to us upon arrival. We want to offer a subset of products S_t to this customer, from the set of available products at time t. The expected revenue of any assortment S is $\sum_{i \in S} r_i \phi^t(i, S)$.

For any given subset S_t, user at time t selects a product or no-purchase according to their choice model ϕ^t. If user t selects a product j, he uses it for a random number of periods, $\tilde{t}_j$ that is distributed i.i.d. according to some distribution that depends only on product j. This purchase generates a revenue $r_j(\tilde{t}_j)$ for us before the product comes back into the inventory. Full revenue is collected regardless of the end of the horizon. At the end of the horizon no further customers arrive but the revenue is still collected from resources in use until they are returned.

We assume Assumptions 1 and 2. The arrival sequence of customers and their respective preferences can be adversarially chosen, but the adversary is oblivious. In other words, we make no assumptions (distributional or otherwise) on the arrival sequence and allow it to be completely arbitrary (periodic or aperiodic/continuous). The goal of the seller is to find a policy that maximizes the expected cumulative revenue over a finite horizon T:

$$\max\ \mathbb{E}\Big[\sum_{t=1}^{T}\sum_{j\in S_t} r_j\phi^t(j,S_t)\Big]$$

where the expectation is over product choices and realizations of usage times.

We take the perspective of competitive analysis on our policy, where we measure the worst case performance of a policy against a clairvoyant optimal policy that is computed with the complete knowledge of arrival epochs, the choice models of all customers and all the usage time distributions. Note that OPT does not know the realizations of product choices and usage times beforehand. We compare the worst case expected revenue collected by our policy with the optimal expected revenue collected by the optimal policy, where the expectation is again over product choices and realizations of usage times.

Our Contributions. Our main contribution is to show that a myopic policy provides a good approximation for the online assortment optimization problem with reusable products.

Myopic Policy. For each user, the myopic policy offers an assortment $S\in\mathcal{S}$ from the set of available products that maximizes the expected revenue from that user. More specifically, suppose user at time t has type z_t and let $\mathcal{I}_t$ be the set of products available to the myopic policy at time t. Then the myopic policy offers assortment

S_t where,

$$S_t \in \ \mathsf{argmax} \ \left\{ \sum_{i \in S} r_i \cdot \phi^{z_t}(i, S) \ \middle| \ S \subseteq \mathcal{I}_t, S \in \mathcal{S} \right\},$$

where recall, ϕ^z is the choice model for user type z, $\phi^z(i, S)$ is the probability that user type z selects product i given assortment S and r_t is the expected revenue when product i is selected where the expectation is taken over the usage time of product i. Recall that we assume the usage time distribution only depends on the product and is not dependent on user type. Therefore, the myopic policy only needs the expected revenue r_j from product j if it is selected and does not need any further information about the usage time distribution. Further, the optimal set S_t can be found using any black-box algorithm for static assortment optimization.

We show that this myopic policy is 1/2-competitive. In other words, the expected revenue of the myopic policy is at least 1/2 times the expected revenue of an optimal policy that has full information about the sequence of user types and product usage distributions (although not the choice realizations and the realization of usage times). We refer to this as the *clairvoyant benchmark*. More generally, when the (possibly constrained) static assortment optimization problem at each stage can only be solved up to within an α factor of the optimal, our myopic policy is $\alpha/(1+\alpha)$-competitive. We would also like to remark that even for case of non-reusable items, which is a special case of our setting, there are instances where the myopic policy achieves exactly 1/2 the value of clairvoyant (for examples, see Golrezaei et al. (2014); Karp et al. (1990)).

Impossibility Result for User Type Dependent Uage Distributions. We also show that if the usage time distribution depends on the user type, then there is no online algorithm that can obtain a constant factor competitive ratio as compared to our clairvoyant benchmark for the case of adversarial arrivals. This result holds even in the large capacity case. We would like to note that Rusmevichientong et al. (2020) consider the case where usage time distributions can depend on the user type. However, they consider the setting of stochastic arrivals with known distribution of user types and give a 1/2-approximation compared to the optimal dynamic programming solution as opposed to the clairvoyant benchmark.

Challenges and New Techniques. We would like to note that even with full-information about the sequence of users and the usage time distribution, computing an optimal policy is intractable due to the curse of dimensionality. Golrezaei et al. (2014) use an LP-based upper bound as a benchmark for the case of non-reusable products. One of the challenges in extending the results to the case of reusable products is the lack of a good LP-based upper bound. The LP formulation in Golrezaei et al. (2014) has an unbounded gap since it does not account for reusability and therefore, is not a useful benchmark for the problem. On the other hand, LP upper bounds of the form proposed in Dickerson et al. (2018) naturally capture reusability, however, it is not immediately clear how to perform a primal-dual type analysis with this more involved LP. Part of the reason is that due to reusability, the remaining capacity of products non-monotonic (capacity decreases when product is used but increases when used units return). But more importantly, this non-montonicity is inherently stochastic due to the uncertainty in usage durations and this makes it non-trivial to dual-fit in the way of Golrezaei et al. (2014); Devanur et al. (2013).

In order to prove the competitive ratio bound, we design a novel *queue-based coupling* that allows us to upper bound the expected revenue of an optimal algorithm with full information in terms of the expected revenue of the myopic policy, for any usage time distributions.

Further work: Since a version of this chapter appeared online, there have been several developments in these settings, especially in the large capacity regime. In case of adversarial arrivals, Feng et al. (2019) show that the inventory balancing algorithm of Golrezaei et al. (2014) is $(1-1/e)^2 \approx 0.4$ competitive for large capacity. For the special case of deterministic/fixed usage durations, they show the best possible performance guarantee of $(1-1/e)$ in the large capacity regime. In contrast, Goyal et al. (2021) (merging two earlier papers Goyal et al. (2020a) and Goyal et al. (2020b)) demonstrate that the general case of stochastic usage durations is fundamentally different and inventory balancing may not be sensitive enough to address reusability even for simple two-point usage distributions. First, they propose a new ranking based allocation scheme and show that it achieves the best possible guarantee of $(1-1/e)$ for large capacities when the usage distributions are IFR (roughly speaking). Building on this insight they develop a fluid guided algorithm that is $(1-1/e)$ competitive for arbitrary usage distributions when the capacities are large. To analyze these algorithms they introduce a new LP free analysis approach inspired by the primal-dual analysis of Devanur et al. (2013) and its path-based generalization in Goyal and Udwani (2020). Beating the performance of the myopic algorithm without the large capacity assumption remains open.

In the stochastic arrival setting, Feng et al. (2019) show a competitive ratio of 1/2 against a stronger LP benchmark and with a different simulation based policy (recall that Dickerson et al. (2018) showed such a result for the special case of online matching of reusable resources). In concurrent work, Baek and Ma (2019) gave a 1/2 competitive policy against the LP benchmark more generally for network revenue management with reusable resources. Finally, Feng et al. (2020) show a near optimal result in the stochastic arrival case for large capacities.

4.1.1 Related Work

There is a considerable amount of literature on dynamic assortment optimization problems with non-reusable products starting with Bernstein et al. (2015), which studied the problem of dynamic assortment optimization for a stochastic arrival model where users choose according to a multinomial logit choice model (Talluri and Van Ryzin (2004); Liu and van Ryzin (2008); Gallego et al. (2004); Topaloglu (2013)) and the user type is drawn i.i.d. from a stationary distribution. Chan and Farias (2009) considered a stochastic depletion framework for non-stationary environments which includes the assortment planning problem under random arrivals. They gave a 1/2-competitive myopic policy for this general framework. More recently, Stein et al. (2018); Wang et al. (2018) consider other closely related models for online product allocation with stochastic arrivals. We refer the reader to Golrezaei et al. (2014) for a more detailed review.

For revenue management with reusable products and random usage times, Levi and Radovanović (2010) first studied a product independent demand model where the users do not exhibit any choice behavior and the goal is to design a policy to maximize the average revenue in an infinite horizon setting. Owen and Simchi-Levi (2018) extend this model to include user choices and also study the infinite horizon setting. Chen et al. (2017) consider a related problem of control admission for a system with multiple units of a single product which can be reserved in advance for time intervals determined by users arriving according to a multi-class Poisson process.

Product allocation problems also closely relate to online matching problems and often generalize the classical online bipartite matching problem of Karp et al. (1990). In this seminal work, they showed that matching arriving users (the unknown vertices) based on a random RANKING over all products (vertices on the known side of the graph) gives the best possible competitive guarantee of $(1 - 1/e)$.

We refer the interested reader to Mehta et al. (2013) for a more detailed review of work on online matching and its variants/generalizations.

4.2 Competitive Ratio of Myopic Policy

In this section, we show that the myopic policy is 1/2-competitive for general usage time distributions and general revenue functions (as functions of usage times). Before proceeding, we discuss useful notation and introduce an important simplification.

Notation. Let $\mathcal{N} = \{1, \ldots, n\}$ be the set of products available to the platform with capacities $c_1, \ldots, c_n$. Let us refer to the myopic policy as ALG and the clairvoyant optimal as OPT. Recall, we let $\mathcal{S}$ denote the set of feasible assortments. Let $\mathcal{Z}$ denote the set of user types. For any $z \in \mathcal{Z}$, $S \subseteq \mathcal{N}$, $i \in S \cup \{0\}$, let $\phi^z(i, S)$ denote the probability that user type z selects i when offered assortment S. The choice probabilities satisfy Assumption 1. Also recall, r_j is the expected revenue to the seller when j is selected by any user and F_j is the cdf of usage duration for product j. $R(S, z_t) = \sum_{i \in S} r_t \cdot \phi^{z_t}(i, S)$ is the expected revenue of assortment S for user type z_t. Let ω denote the sample path that specifies the random preference realizations of all users $z_1, \ldots, z_T$ and the random usage times. Let $\mathcal{I}_t(\omega)$ denote the set of available products in ALG at time t on sample path ω. Also, let $S_t(\omega)$ ($S_t^*(\omega)$ respectively) denote the assortment offered by ALG (OPT respectively) at time t to user type z_t. Here recall $S_t(\omega) = \arg\max_{S \in \mathcal{I}_t(\omega)} R(S, z_t)$. Let $j_t(\omega) \in S_t(\omega) \cup \{0\}$ be the product selected by user z_t in ALG, and let $j_t^*(\omega) \in S_t^*(\omega) \cup \{0\}$ be the product selected by user z_t in OPT at time t on sample path ω. Note that product 0 refers to the do-nothing or exit option.

From arbitrary to unit inventory: We now argue that one can safely assume $c_j = 1$ for all j, since a guarantee for the case of unit inventory leads to a stronger result that generalizes to the case of arbitrary inventories. This allows us to perform a simpler and crisper analysis without loss of generality (w.l.o.g.). We note that such a property is commonly used in the online matching literature for non-reusable resources (Mehta et al., 2013), and we show this also for our assortment setting with reusable resources.

Given a setting with arbitrary inventories $\{c_j\}_{j\in\mathcal{N}}$. Consider a unit inventory setting where for each j we have c_j identical products (with the same usage distribution as the original product), instead of c_j units of product j in the original instance. We refer to this as the *unit setting*. In the unit setting we index the products as (j, k_j), where $k_j \in [c_j]$ for every $j \in \mathcal{N}$. For any given assortment S_u in the unit setting, we let S denote the set of products in the original instance with $j \in S$ if there exists some $k_j \in [c_j]$ such that $(j, k_j) \in S_u$. Now, given an arrival with choice model ϕ on the original product space, define the following choice function ϕ_u in the new space of products:

$$\phi_u\left((j,k_j),S_u\right) = \begin{cases} \phi(j,S) & \text{if for every } k < k_j,\ (j,k) \notin S_u, \\ 0 & \text{otherwise.} \end{cases}$$

Further, we couple the customer choice between the two settings so that when customer chooses product $j \in S$ in the original instance, the unique product $(j, k_j) \in S_u$ such that $\phi_u\left((j,k_j),S_u\right) = \phi(j,S)$, is chosen in the unit setting and vice versa.

Note that if choice model ϕ satisfies Assumption 1 then so does the unit setting choice model ϕ_u. Given an arrival sequence in the original setting, we construct an arrival sequence for the unit setting using this transformation. The following proposition shows that the expected revenue of OPT (and ALG) does not change when we perform this transformation.

Proposition 2 *Given an instance of the problem with arbitrary inventories $\{c_i\}_{i\in\mathcal{N}}$, the expected total revenue of* OPT *and* ALG *remains unchanged in the transformed instance with unit inventories.*

Proof: Consider an arbitrary arrival sequence in the original instance and its equivalent sequence (as given by the transformation) in the unit setting. Given an optimal algorithm (OPT) for the original instance, we construct an algorithm for the unit setting with the same expected revenue.

For every product j, whenever OPT includes j in an assortment for the original instance, we include exactly one available product (j, k_j), for an arbitrary $k_j \in [c_j]$, in the unit setting. This defines a policy for the unit setting but to make this definition meaningful we need to ensure that whenever j is available in the original instance, some product (j, k_j) is available in the unit instance. This is true at the first arrival. Inductively, following the defined policy while using the same realization of usage times for both settings and coupling the customer choice as described earlier, we have that if product j is available in the original instance at arrival t then we are guaranteed some k_j such that (j, k_j) is available in the unit setting at t. Therefore, we have a well defined policy for the unit setting with expected revenue exactly as much as OPT in the original setting.

The reverse is also true. Given the optimal algorithm for the unit setting, whenever a product (j, k_j) is included in the assortment, we include product j in the assortment for the original instance. Once again due to the coupling between the choice models and using the same realizations of usage durations, whenever a product

(j, k_j) is available in the unit setting at least one unit of j is available in the original setting. This leads to a policy for the original setting with the same expected revenue as the optimal policy for the unit setting.

Therefore, the optimal value of clairvoyant is the same in both instances. In fact, using the same argument the expected revenue of ALG is also identical in the two settings. □

In the rest of this section we will show that on every unit inventory instance the expected revenue of ALG is at least half of OPT. The above proposition then gives us our general result for arbitrary inventories. Note that, while the number of products in the unit setting can be much larger, this is only for the purpose of analysis and has no impact on the actual run time of the myopic policy (which is on the original space of products and choice models).

Theorem 6 *Suppose for every product j, the usage time is distributed according to a distribution that only depends on the product j itself. Then for any sequence of user types $z_1, \ldots, z_T$, the expected cumulative revenue of the myopic policy is at least $1/2$ times the expected cumulative revenue of the clairvoyant optimal that knows the full sequence, i.e.,*

$$\mathbb{E}_\omega \left[\sum_{t=1}^{T} R(S_t(\omega), z_t) \right] \geq \frac{1}{2} \cdot \mathbb{E}_\omega \left[\sum_{t=1}^{T} R(S_t^*(\omega), z_t) \right].$$

Proof: The proof proceeds by decomposing the revenue $\mathbb{E}_\omega[R(S_t^*(\omega), z_t)]$ of OPT into two parts, one corresponding to the products in $S_t^*(\omega)$ that are available in ALG at time t and the other corresponding to products in $S_t^*(\omega)$ that are unavailable in ALG at time t. We upper bound the former by the revenue $\mathbb{E}_\omega[R(S_t(\omega), z_t)]$ of ALG for the same arrival. The key challenge is to bound the total expected revenue in

OPT from products that are unavailable in ALG at the time they are offered in OPT. We do this subsequently in Lemma 6. Formally,

$$\begin{aligned}
\mathbb{E}_\omega\left[\sum_{t=1}^{T} R(S_t^*(\omega), z_t)\right] &= \mathbb{E}_\omega\left[\sum_{t=1}^{T}\sum_{j\in S_t^*(\omega)} r_j\cdot\phi^{z_t}(j, S_t^*(\omega))\right]\\
&= \mathbb{E}_\omega\left[\sum_{t=1}^{T}\left(\sum_{j\in S_t^*(\omega)\cap\mathcal{I}_t(\omega)} r_j\cdot\phi^{z_t}(j, S_t^*(\omega)) + \sum_{j\in S_t^*(\omega)\setminus\mathcal{I}_t(\omega)} r_j\cdot\phi^{z_t}(j, S_t^*(\omega))\right)\right]\\
&\le \mathbb{E}_\omega\left[\sum_{t=1}^{T}\sum_{j\in S_t^*(\omega)\cap\mathcal{I}_t(\omega)} r_j\cdot\phi^{z_t}(j, S_t^*(\omega))\right] + \mathbb{E}_\omega\left[\sum_{t=1}^{T} R(S_t(\omega), z_t)\right]\\
&\le \mathbb{E}_\omega\left[\sum_{t=1}^{T}\sum_{j\in S_t^*(\omega)\cap\mathcal{I}_t(\omega)} r_j\cdot\phi^{z_t}(j, S_t^*(\omega)\cap\mathcal{I}_t(\omega))\right] + \mathbb{E}_\omega\left[\sum_{t=1}^{T} R(S_t(\omega), z_t)\right]\\
&\le 2\cdot\mathbb{E}_\omega\left[\sum_{t=1}^{T} R(S_t(\omega), z_t)\right],
\end{aligned}$$

The first inequality follows from the claim, $\mathbb{E}_\omega\left[\sum_{t=1}^{T}\sum_{j\in S_t^*(\omega)\setminus\mathcal{I}_t(\omega)} r_j\cdot\phi^{z_t}(j, S_t^*(\omega))\right] \le \mathbb{E}_\omega\left[\sum_{t=1}^{T} R(S_t(\omega), z_t)\right]$, which we show in Lemma 6. The second inequality follows from Assumption 1. The final inequality follows from the definition of $S_t(\omega)$, which is a subset of $\mathcal{I}_t(\omega)$ that maximizes the single period revenue for user type z_t. $\square$

Queue Coupling Technique. In order to bound the total expected revenue in OPT from products that are unavailable in ALG at the time they are offered in OPT, we introduce a new coupling between the usage times in ALG and OPT. In particular, we introduce *coupling queues* to specify the coupling of usage times between sample paths in ALG and OPT. For each product j, we maintain a queue, $\mathcal{Q}_j$. Initially $\mathcal{Q}_j$ is empty.

Whenever product j is selected in ALG by any user, we generate an i.i.d. sample from the usage time distribution F_j and insert the sample at the rear of the queue, $\mathcal{Q}_j$.

Whenever product j is selected in OPT by any user, we get the first element of queue, $\mathcal{Q}_j$ and use it as a usage time sample for product j in OPT (we also remove this element from the queue). So we use the samples in $\mathcal{Q}_j$ in a FIFO order. This couples the usage distributions of ALG and OPT. In case $\mathcal{Q}_j$ is empty, we generate an i.i.d. sample from F_j.

Lemma 5 *For any time $t = 1, \ldots, T$ and any product $j = 1, \ldots, n$, whenever a user selects product j in* OPT *the usage time distribution given by the above coupling is i.i.d. according to the usage time distribution for product j.*

Proof: The interesting case is when $\mathcal{Q}_j$ is not empty. Then the sample that is used by OPT and removed from the queue, denoted by $\tilde{L}_j$, was originally picked independently from all previous samples and added to the queue unconditionally. Any other samples that might have been added to the queue subsequent to adding $\tilde{L}_j$, do not affect $\tilde{L}_j$. All samples in a queue have the same probability of being the front of the queue. Therefore, the samples obtained from the queues by selection of the product in OPT are i.i.d. according to F_j. $\square$

We are now ready to bound the total expected revenue in OPT from products that are unavailable in ALG at the time they are offered in OPT. In particular, we have the following lemma.

Lemma 6 *For any usage time distributions, and sequence of user types $z_1, \ldots, z_T$,*

$$\mathbb{E}_\omega \left[\sum_{t=1}^{T} \sum_{j \in S_t^*(\omega) \setminus \mathcal{I}_t(\omega)} r_j \cdot \phi^{z_t}(j, S_t^*(\omega)) \right] \leq \mathbb{E}_\omega \left[\sum_{t=1}^{T} R(S_t(\omega), z_t) \right].$$

Proof: Consider any ω such that $j_t^*(\omega) \in S_t^*(\omega) \setminus \mathcal{I}_t(\omega)$. Let us refer to $j_t^*(\omega)$ as j_t^* for brevity. At time t, j_t^* is not available in ALG on sample path ω. Therefore, it is in use at time t and must have been selected in ALG at some previous time period, say $t - \tau$ for some $\tau \geq 1$. Let $\tilde{L}$ be the random usage time that ALG sampled for j_t^* at time $(t-\tau)$ and inserted in the queue $\mathcal{Q}_{j_t^*}$ corresponding to product j_t^*. Since j_t^* is still in use by ALG by our coupling, we get $\tilde{L} \geq \tau$. Using this and the fact that OPT is able to select j_t^* at time t, we have that the sample $\tilde{L}$ must exist on the queue up to time t (but may be popped at t). Therefore, $\mathcal{Q}_{j_t^*}$ is non empty before user arrives at t.

Hence, when OPT selects j_t^* at time t, we get a sample from $\mathcal{Q}_{j_t^*}$. Suppose the sample used by OPT was generated for ALG at time $t' \leq t - \tau$. We charge the revenue earned by OPT for this selection to the revenue earned by ALG for using j_t^* at time t'. Observe that the charging is unique since each sample on the queue is used at most once by OPT, and we only charge to ALG when the corresponding sample is used by OPT. Therefore,

$$\mathbb{E}_\omega\left[\sum_{t=1}^{T} \sum_{j \in S_t^*(\omega)\setminus\mathcal{I}_t(\omega)} r_j \cdot \mathbb{1}(j = j_t^*(\omega))\right] \leq \mathbb{E}_\omega\left[\sum_{t=1}^{T} r_{j_t(\omega)}\right] = \mathbb{E}_\omega\left[\sum_{t=1}^{T} R(S_t(\omega), z_t)\right]. \tag{4.1}$$

We would also like to note that the revenue from a product can now even depend on the usage time duration of the product. Simplifying as before, we get

$$\begin{aligned}
\mathbb{E}_\omega\left[\sum_{t=1}^{T}\sum_{j\in S_t^*(\omega)\setminus\mathcal{I}_t(\omega)} r_j\cdot\phi^{z_t}(j,S_t^*(\omega))\right] &\le \mathbb{E}_\omega\left[\sum_{t=1}^{T}\sum_{j\in S_t^*(\omega)\setminus\mathcal{I}_t(\omega)} r_j\cdot\phi^{z_t}(j,S_t^*(\omega)\setminus\mathcal{I}_t(\omega))\right]\\
&= \mathbb{E}_\omega\left[\sum_{t=1}^{T}\sum_{j\in S_t^*(\omega)\setminus\mathcal{I}_t(\omega)} r_j\cdot\mathbb{1}(j=j_t^*(\omega))\right]\\
&\le \mathbb{E}_\omega\left[\sum_{t=1}^{T} R(S_t(\omega),z_t)\right],
\end{aligned}$$

where the first inequality follows from Assumption 1 and the last inequality follows from (4.1). □

Generalizations and Extensions

The following results follow as a direct consequence of out main result.

Corollary 2 *Given an α-approximation algorithm for solving the (possibly constrained) static assortment optimization problem at each stage, our myopic policy is $\alpha/(1+\alpha)$-competitive.*

Proof: Recall the proof of Theorem 6 where we split the revenue of OPT into two parts, one corresponding to the products in $S_t^*(\omega)$ that are available in ALG at time t and the other corresponding to products in $S_t^*(\omega)$ that are unavailable in ALG at time t. The latter term is still bounded as before since Lemma 6 holds even if the static assortment optimization problem at each stage can only be solved approximately. However, the former is now bounded as follows,

$$\mathbb{E}_\omega \left[\sum_{t=1}^{T} \sum_{j \in S_t^*(\omega) \cap \mathcal{I}_t(\omega)} r_j \cdot \phi^{z_t}(j, S_t^*(\omega) \cap \mathcal{I}_t(\omega)) \right] \leq \frac{1}{\alpha} \cdot \mathbb{E}_\omega \left[\sum_{t=1}^{T} R(S_t(\omega), z_t) \right].$$

Resulting in a competitive ratio $\alpha/(1+\alpha)$. □

The following corollary addresses situations where the revenue from each product is non-stationary and varies across time.

Corollary 3 *Letting γ denote the ratio of minimum to maximum revenue of any product across all arrivals, the myopic policy is $\frac{\gamma}{2}$- competitive.*

The proof follows directly by considering the worst case scenario where the revenue collected by our myopic policy is always with maximum markdown, while none of the revenue collected by OPT is discounted. As an example application of the corollary: if the maximum markdown on products over the entire planning horizon is 10%, then $\gamma = 0.9$ and the myopic policy guarantees a total revenue at least 0.45 times that of the clairvoyant.

Tighter result using booking limits: This result above is arbitrarily bad when $\gamma \to 0$ i.e., when products may be sold to customers at a steep discount. In this case we can do better by setting random booking limits. This idea is inspired by a similar notion in Ball and Queyranne (2009) for a setting with non-reusable resources. It generalizes naturally to the reusable case and leads to a worst case guarantee of $\frac{1}{4}$.

Formally, for any product j, let r_j denote the normal price of the item and let r_{jd} denote the discounted price. We treat each product as two separate products and assume that the choice model for every arriving customer dictates (possibly) different probabilities based on prices.

Recall that $\gamma = \min_j \frac{r_{jd}}{r_j}$. The new myopic policy works as described next. In the beginning, we randomly decide whether to offer products at a discount. Specifically, with probability 0.5 we consider both the discounted and normal versions all products when finding optimal assortment for customers and with remaining probability of 0.5, we do not include the discounted version of products in any assortment. This protects against the possibility of selling products at steep discounts when future arrivals would have chosen the same product at higher prices. After randomly pruning the discounted products at the start of the planning horizon, to each customer we offer the revenue maximizing assortment.

Lemma 7 *The myopic policy with random booking limit is $\frac{1}{4}$ competitive even for $\gamma \to 0$.*

Proof: On each sample path ω, let $\mathcal{I}_t(\omega, d)$ denote the set of products available at t in ALG when discounted products are included. We write $\mathcal{I}_t(\omega)$ to denote the set of available products when discounted products are blocked. Let $S_t^*(\omega)$ and $S_t^*(\omega, d)$ denote the subset of normal price products and the subset of discounted products offered to arrival t in OPT. Let $S_t(\omega)$ denote the assortment offered to t in ALG when discounted products are excluded and let $S_t(\omega, d)$ denote the overall assortment when discounted products are included in ALG. Finally, let ALG_d denote the expected revenue of ALG given that discounted products are included and similarly, ALG_n denotes the expected revenue given discounted products are excluded. Overall, $\text{ALG} = 0.5(\text{ALG}_d + \text{ALG}_n)$. Now,

$$\mathbb{E}_\omega\left[\sum_{t=1}^{T} R(S_t^*(\omega) \cup S_t^*(\omega, d), z_t)\right] \leq \mathbb{E}_\omega\left[\sum_{t=1}^{T} R(S_t^*(\omega), z_t)\right] + \mathbb{E}_\omega\left[\sum_{t=1}^{T} R(S_t^*(\omega, d), z_t)\right].$$

Applying the decomposition used in proving Theorem 6 and using the coupling argument from Lemma 6 it follows that,

$$\begin{aligned}\mathbb{E}_\omega\left[\sum_{t=1}^{T} R(S_t^*(\omega), z_t)\right] &\leq \mathbb{E}_\omega\left[\sum_{t=1}^{T} R\left(S_t^*(\omega)\cap \mathcal{I}_t(\omega), z_t\right)\right] + \mathbb{E}_\omega\left[\sum_{t=1}^{T} R(S_t(\omega), z_t)\right]\\ &= 2\mathbb{E}_\omega\left[\sum_{t=1}^{T} R(S_t(\omega), z_t)\right] = 2\mathrm{ALG}_n.\end{aligned}$$

Similarly,

$$\begin{aligned}\mathbb{E}_\omega\left[\sum_{t=1}^{T} R(S_t^*(\omega, d)), z_t)\right] \leq &\mathbb{E}_\omega\left[\sum_{t=1}^{T} R\left(S_t^*(\omega, d)\cap \mathcal{I}_t(\omega, d), z_t\right)\right]\\ &+ \mathbb{E}_\omega\left[\sum_{t=1}^{T} R(S_t(\omega, d), z_t)\right]\\ \leq &2\mathbb{E}_\omega\left[\sum_{t=1}^{T} R(S_t(\omega, d), z_t)\right] = 2\mathrm{ALG}_d.\end{aligned}$$

Combining the inequalities, we have OPT $\leq$ 4ALG, as desired. $\square$

4.3 User Type Dependent Usage Times: Family of Bad Examples

We now consider the case where the product usage time distributions could depend on the user type and show there is no online algorithm with a constant competitive ratio in the adversarial arrival model. The result holds even in the high capacity regime where the capacities of all products goes to $+\infty$. It will suffice for us to consider a single product. For a user arriving at time t, let d_t denote the random usage duration. Even for this special case with a single reusable product we have the following upper bound on the competitive ratio of any online algorithm.

Theorem 7 *For online matching with a single reusable product and arbitrary product capacity, if the random usage durations depend on the user, no online (randomized) algorithm can have competitive ratio better than $O\left(\frac{\log T}{T}\right)$, where T is the number of users.*

The proof is provided in Appendix B.1.

Remark: Subsequent work Goyal et al. (2021) shows that no meaningful competitive ratio result is possible even in the case of *deterministic* user type dependent usage durations.

4.4 Computational Experiments

We compare the performance of our myopic policy against the approximate dynamic program based algorithm (*the DP-based policy*) in Rusmevichientong et al. (2020), and the inventory-balancing policy (*the IB policy*) in Golrezaei et al. (2014).

Experimental Setup.

We consider $N = 5$ products indexed by $\mathcal{N} = \{1, \ldots, 5\}$ and $M = 5$ customer types. We consider a selling horizon of $T = 300$ periods. In each period, a random customer from a known distribution over M types arrives. We offer an assortment to each customer when they arrive, who in turn either purchases a product in the offered assortment, or leaves the platform without making any purchase. We experiment with three different levels of starting inventory for all products: i) scarce inventory: 1 unit per product, ii) moderate inventory: 5 units per product and iii) abundant inventory: 20 units per product, to demonstrate the performance of the algorithms at different levels of abundance of products.

We first experiment when the price and usage time distribution for each product is fixed. In the later part of this section, we change the setting parameters to further investigate the performance of the myopic policy when these assumptions do not hold. We select prices for the products to be evenly spaced in $[15, 30]$. Revenue r_i is collected whenever a product of type i is chosen by a customer. The usage time distribution for each product i is a geometric distribution with parameter $p_i \in [0.05, 0.07]$ and expected usage time between 14 and 20 days (until we remove this assumption later). In particular, for product type i, the parameter is $p_i = \frac{1}{20-i}$. Therefore, type 1 product is the most expensive and has the longest expected usage time, and type 5 product is the least expensive and has the shortest expected usage time.

We follow the MNL model with consideration sets in the computational experiments as in Golrezaei et al. (2014) and Rusmevichientong et al. (2020). For any $j \in [M]$, customers of type j have the consideration set $\mathcal{C}_j = \{1, \dots, j\}$. Each customer makes a choice among the assortment they are provided according to the multinomial logit model. A customer of type j associates the preference weight w_i^j with product i and the preference weight w_0^j with the no purchase option. When offered assortment S, a customer of type j chooses product $i \in S$ with probability $\phi_i^j(S) = \frac{w_i^j}{w_0^j + \sum_{\ell \in S} w_\ell^j}$. The weight w_i^j over product i of type j customers is generated uniformly randomly from $[0, 1]$ for all j and for any product i in the customer's consideration set. We calibrate the preference weight of the no-purchase option so that for any customer type, if we offer all the products to the customer, the probability of the customer leaving without making a purchase is 0.1, i.e. $w_0^j / (w_0^j + \sum_{\ell \in [N]} w_\ell^j) = 0.1$.

We experiment in a scenario that is less favorable to our myopic policy. More specifically, our setting generates customer arrivals so that the pickier customers are more concentrated in the later part of the selling horizon. Therefore, being myopic can be harmful in this setting. We choose equally-spaced time periods $\tau^N \leq \tau^{N-1} \cdots \leq \tau^2 \leq \tau^1$ over the selling horizon as in Rusmevichientong et al. (2020). The probability $p^{t,j}$ that a customer of type j arrives at time period t is proportional to $e^{-\kappa|t-\tau^j|}$. So the arrival probability for a customer of type j peaks at around time period τ^j. Because $\tau^N \leq \tau^{N-1} \cdots \leq \tau^2 \leq \tau^1$, as $\kappa \to \infty$, we obtain an arrival process where customers of type 5 arrive first, followed by customers of type 4. As $\kappa \to 0$, we have $p^{t,j} \to 1/M$, in which case, different customer types arrive with equal probability at each time period. Thus, we can control how much the arrival order for the customer types deviates from the equal probability distribution through the parameter κ. For our experiments, we use $\kappa = 0.5$.

Algorithms and Benchmark.

We evaluate the performance of our myopic policy as well as the DP policy and the IB policy. Note that we compute the DP policy assuming the knowledge of the distribution of arrival types, while our policy and the IB policy are agnostic to the arrival distribution. We also compare the performance of our myopic policy with the DP-based policy when the realized distribution of arrivals is slightly perturbed. In particular, we consider the following notion of "noise":

We use a scalar λ to control the noise or perturbation from the assumed distribution as follows: at any time t, with probability $1 - \lambda$, the arrival customer type is chosen according to the original distribution (where $p^{t,j} \propto e^{-\kappa|t-\tau^j|}, \forall j$); and with probability λ, the arrival customer type is chosen from the uniform probability distribution where $p^{t,j} = 1/M$. Therefore, when $\lambda = 0$, the *DP-based policy* has fully accurate distributional knowledge. We evaluate and compare the performances of our myopic algorithm, the DP-based algorithm, and the IB algorithm for different noise levels.

LP Upper Bound: We use a natural adaptation of the LP upper bound in Dickerson et al. (2018) as a benchmark. This LP imposes inventory constraints *in expectation* over randomness in rental times:

$$\begin{aligned} \underset{y_{S,t}}{\text{maximize}} \quad & \sum_{t=1}^{T} \sum_{S \in \mathcal{S}} \sum_{i=1}^{n} r_i \phi_i^{z_t}(S) y_{S,t} \\ \text{subject to} \quad & \sum_{\tau=1}^{t} \sum_{S \in \mathcal{S}} \big(1 - F_i(t-\tau)\big) \phi_i^{z_\tau}(S) y_{S,\tau} \leq c_i, i \in [n], t \in [T], \\ & \sum_{S \in \mathcal{S}} y_{S,t} \leq 1\,, t \in [T] \end{aligned} \tag{4.2}$$

The decision variables $\{y_{S,t}\}$ correspond to the probability that assortment S is offered to the customer arriving at time t.

Results. For each experiment setting described above or later in this section, we ran the LP benchmark and each algorithm 1000 times over randomly generated customer preference weights, customer arrivals, usage times and purchase choices for statistical significance.

Table 4.1 shows our computational results in the basic setting under different levels of noise (λ) added to the customer arrivals. The columns corresponding to each algorithm give the ratios of the average revenue of the algorithm (over 1000 instances) divided by the average revenue of the LP upper bound (over 1000 instances) under low, moderate and high inventory scenarios. The standard deviation of the algorithms are very similar and therefore omitted.

Table 4.1 shows that our myopic algorithm achieves comparable optimality with the other two benchmark algorithms in Rusmevichientong et al. (2020) and in Golrezaei et al. (2014) in practice when the customer arrivals are stochastic and not adversarial, even though our policy is myopic. The myopic policy slightly outperforms the other algorithms when inventory is abundant.

	Myopic Alg			*IB Alg*			*DP-based Alg*		
λ	C=1	C=5	C=20	C=1	C=5	C=20	C=1	C=5	C=20
0.0	0.949	0.927	0.996	0.950	0.927	0.994	0.948	0.937	0.975
0.2	0.908	0.913	0.995	0.909	0.911	0.996	0.913	0.931	0.973
0.5	0.909	0.895	0.994	0.908	0.893	0.992	0.926	0.917	0.970
1	0.910	0.855	0.993	0.914	0.857	0.994	0.919	0.884	0.970

Table 4.1: Comparison under different levels of noise (higher λ implies more noise) added to the customer arrivals, with inventory levels $C = 1, 5, 20$, and arrival distribution parameter $\kappa = 0.5$. An entry represents the average performance of the algorithm divided by the average value of the LP upper bound. The standard deviations of revenue for the three algorithms in the 1000 repeated experiments for each inventory level are very similar and therefore omitted.

Next, we examine the performance of the myopic policy in more general settings where the analytical guarantee does not hold. Table 4.2 compares the cumulative revenue of the three algorithms and the LP upper bound when the usage time depends on the users. In particular, every time we run the experiment, for each product-user type pair, with probability 0.5 the usage time distribution is a geometric distribution as before and with remaining probability 0.5 this user type never returns this product type. The DP and LP benchmarks incorporate the knowledge of the user-type-dependent usage time distribution in rewards while the myopic policy does not.

	Myopic Alg			*IB Alg*			*DP-based Alg*		
λ	C=1	C=5	C=20	C=1	C=5	C=20	C=1	C=5	C=20
0.0	0.522	0.480	0.666	0.514	0.479	0.664	0.521	0.487	0.709
0.2	0.376	0.436	0.694	0.377	0.435	0.694	0.382	0.443	0.737
0.5	0.318	0.405	0.707	0.313	0.404	0.708	0.328	0.413	0.743
1	0.315	0.397	0.750	0.314	0.396	0.750	0.317	0.408	0.780

Table 4.2: Comparison when usage time depends on the customer type, with $C = 1, 5, 20$. The standard deviations of revenue for the three algorithms in the 1000 repeated experiments are very similar.

Table 4.2 shows that the DP-based algorithm noticeably outperforms the other two algorithms when the usage time depends on the user type. The ratio between any of the three algorithms and the LP upper bound drops significantly. As we can see, removing the assumption that the usage time for each product is independent of customer type leaves the myopic policy at a disadvantage compared with the DP-based algorithm. However, the difference is not enormous.

Table 4.3 compares the cumulative revenue of the three algorithms and the LP upper bound when the revenue collected for each product depends on the users. In particular, every time we run the experiment we generate a uniformly random discount factor for each user type from $[0, 1]$, and the revenue we collect from each user is discounted by the user's discount factor.

	Myopic Alg			*IB Alg*			*DP-based Alg*		
λ	C=1	C=5	C=20	C=1	C=5	C=20	C=1	C=5	C=20
0.0	0.936	0.924	1.000	0.941	0.924	1.000	0.929	0.931	0.982
0.2	0.794	0.893	0.989	0.790	0.896	0.991	0.801	0.909	0.970
0.5	0.769	0.868	0.974	0.774	0.867	0.972	0.780	0.892	0.951
1	0.752	0.840	0.970	0.748	0.840	0.969	0.770	0.868	0.947

Table 4.3: Comparison when revenue depends on the customer type, with $C = 1, 5, 20$. The standard deviations of revenue for the three algorithms in the 1000 repeated experiments are very similar.

The performances of the three algorithms stay comparable even with customer dependent product revenues. Overall, the DP-based algorithm has an advantage over the other two algorithms. However, this advantage diminishes when the inventory level is abundant. We remark that other types of revenue dependence on user types could potentially lead to varied results.

4.5 Conclusions

In this chapter, we consider an online assortment optimization problem with reusable resources or products under an adversarial arrival model. Under the assumption, that product usage time distributions do not depend on the user type, we show that the policy that offers a myopically optimal assortment to every user from the set of available products achieves an expected revenue that is at least $1/2$ times the expected revenue of a clairvoyant algorithm that has full information about the sequence of user types. For the case of reusable capacities, we do not have a good upper bound (LP based or otherwise) for the clairvoyant optimal which makes the comparison with the benchmark challenging. The main contribution of this chapter is a queue-based coupling technique that allows us to relate the expected revenue of the clairvoyant optimal to the expected revenue of the myopic policy. This coupling is algorithmic and might be of independent interest. The assumption that product usage time distribution does not depend on user type is fairly reasonable and satisfied in many settings. We also show that if the assumption is not satisfied, there is no online algorithm that can be constant-factor competitive as compared to our clairvoyant benchmark. Therefore, the assumption is necessary to get any non-trivial performance guarantee for the case of adversarial arrivals.

Our myopic online algorithm is easy to implement, and achieves comparable optimality with the DP-based algorithm in Rusmevichientong et al. (2020) and the inventory-balancing algorithm in Golrezaei et al. (2014) in synthetic experiments, even when some of the assumptions we make in this chapter do not hold.

An interesting open question is to study whether we can obtain stronger results analogous to the online assortment problem with non-reusable and large capacities. In particular, a $(1 - 1/e)$-competitive algorithm in the adversarial arrivals model and a near optimal result in the stochastic arrivals model. The first of these open questions was subsequently resolved in Goyal et al. (2021) (which merges Goyal et al. (2020a) and Goyal et al. (2020b)), and the second in Feng et al. (2020).

Chapter 5

Fast and Exact Cloud Server Deployment Under Demand Uncertainty

5.1 Introduction

The boom of the cloud computing industry in the last decade (Statista, 2021) is nowhere near to stopping, as digitization continues to take place globablly and remote work becomes more of a norm. A Gartner, Inc. report from April 2021 (Gartner, 2021) estimates that world-wide end-user spending on public cloud services will reach 397 billion U.S. dollars in 2022, up 47% from 270 billion U.S. dollars in 2020. Part of the recent acceleration in public cloud adoption has been attributed to the COVID-19 pandemic (Luxner, 2021) as more activities move online, and many of these changes are expected to become permanent even after the pandemic impact diminishes (LaFleur, 2020).

Publicly available cloud services such as Microsoft Azure and Amazon Web Services enable businesses to scale up without incurring large capital and operational expenses. Cloud customers may reserve cloud capacity ahead of time or rent it on demand, shifting the risks of demand volatility to the cloud providers.

To satisfy these resource requests from customers in a timely manner, a main challenge is to deploy new cloud server hardware under demand uncertainty, without incurring unnecessarily large operational costs. Towards that end, cloud providers needs to frequently make hardware deployment decisions, taking into account many cost considerations (shipping, depreciation, building, etc.) and operational constraints (compatibility, capacity, inventory, throughputs, etc.). The minimal hardware unit used to satisfy each demand is a *cluster* - a set of servers (usually in the order of tens) that is jointly installed in the data center. Each cluster is compatible with one of the service genres offered by the cloud provider (e.g., storage, database as a service, analytics and machine learning services, and other SaaS applications). Given the heterogeneity of their operations, demand characteristics vary substantially across different services.

Deploying a new piece of hardware into production is a complex process that takes weeks to finish. In order to successfully deploy a cluster, the cloud provider must first prepare the row in the data center to receive the cluster. This process is referred to as *building a row*, which includes setting up power supply, networking equipment and necessary cabling among other preparation work at the location. The provider must then select the supplier and the date from which the cluster will be acquired.

In production, there are usually dedicated teams to create demand forecasts that are used as an input to the deployment decision process. While some demands can be estimated with high accuracy (hence regarded as *deterministic* for practical purposes), demands that would be materialized farther into the future are prone to estimation errors. Existing methods might consider a myopic approach where the cloud provider optimizes deployment periodically only for deterministic demands. In this work, we study how we can improve the provider's hardware deployment process, fully acknowledging the stochasticity of the demands.

5.1.1 Contributions

We formulate the underlying optimization problem as a two-stage stochastic mixed-integer optimization problem, which is notoriously hard to solve (Birge and Louveaux (2011)). To tackle this problem, we identify the network flow representation of the second-stage problem to solve it quickly via Linear Programming relaxation. Then we design a cutting-plane scheme that leverages this relaxation and Benders decomposition (Benders (1962)). We further accelerate the scheme with ideas from the Level method in Lemaréchal et al. (1995), in order to efficiently generate an initial set of cuts, which addresses the bottleneck for the Benders decomposition scheme for our problem. We prove that our algorithm is *exact*, i.e., it terminates with an *optimal* solution, under common assumptions.

5.2 Motivation and Literature

Now we describe the cloud service deployment operations in more detail and survey the literature that share similarities with our problem.

5.2.1 Overview of Cloud Deployment Operations

A cloud provider operates a fleet of data centers around the world, which are partitioned into sets of co-located data centers, called *regions*.

At a high level, the process of deploying new hardware in a data center consists of two main steps: (i) *building* the row in the data center, which is the preparation work that includes setting up the required infrastructure, such as networking equipment and cabling[1], and then (ii) shipping and placing the cluster on top of a (ready) row.

Data center architecture. Each data center consists of rows (physical locations) on which clusters (computing hardware) can be deployed. Building a row is a time-consuming operation, which takes longer than the shipping of a cluster. As a result, the preparation of a row should begin before the cluster arrives in the data center, and often before a particular cluster is assigned to the row.

Suppliers. The cluster inventory is available from multiple suppliers that are spread in different geo-regions. Without loss of generality, we assume that each supplier holds an inventory of a single cluster type (a supplier that has multiple cluster types can be split into multiple sub-suppliers, each corresponding to a different cluster type); we refer to the amount of available clusters as the supplier *inventory*. Shipping a cluster incurs a fixed *shipping cost*, which depends on the locations of both the supplier and the target region. For simplicity, we assume that the shipping lead times are deterministic.

Demands. A demand is a request for a cluster that has to be deployed in a particular region. Each demand can be deployed in any of the data centers of its target region, provided that there is sufficient capacity. The demand specification includes the cluster type and an ideal dock date. Docking outside the ideal dock date incurs a

[1]The shipping and handling of infrastructure equipment is handled separately, and typically not a bottleneck. Hence, this aspect is not included in our problem description.

penalty. In particular, we assume a per-day *delay penalty* for docking the cluster after the ideal date. We further assume a per-day *idle penalty* for docking a cluster ahead of time; this penalty may reflect operational overheads (e.g., electricity, cooling) as well as depreciation of the hardware. Typically, the delay penalty is larger than the idle penalty for a given demand. The magnitude of the penalties can be used to model prioritization across different demands.

Operational throughput. The number of clusters that can be deployed in each day is upper-bounded by the *throughput* of the data centers. The throughput reflects limitations imposed by personnel availability and physical constraints in the data centers (e.g., number of unloading docks for trucks that transfer the clusters). A throughput constraint may apply to a single data center or a set of co-located data centers within a region. For instance, each of two nearby data centers may have its own personnel constraints, but the two may also share unloading infrastructure with fixed capacity. More generally, the throughput constraints form a *hierarchical* structure – namely, the sets of data centers that are involved in any two throughput constraints cannot have partial overlaps; see Section 5.3 for more details.

Execution plan. Building rows and shipping clusters are operations that have rather long duration. Accordingly, cloud providers typically make execution decisions at relatively slow time scales (say, every day or week). We refer to every decision point of the provider as a decision instance, or in short, an *instance*. Our goal in this chapter is to solve the optimization problem for an instance, whose output is a set of decisions that are carried out until the next instance. We refer to this set as an *execution plan*. An execution plan consists of two classes of decisions:

1. *Cluster assignment.* The planner makes the following decisions for each demand: (i) supplier; (ii) dock date (which determines the shipping date); (iii)

target data center.

2. *Row building.* For every data center, the planner decides how many rows should be built.

These decisions use as input both deterministic demands as well as projections of future demands. As we elaborate in Section 5.3, this would give rise to a stochastic optimization approach. The goal of the provider is to minimize the total cost, consisting of delay, idle, shipping, and row-building costs, as well as penalties for unfulfilled demands. A precise formulation is given in Section 5.3.

5.2.2 Related Work

Cloud resource management. Cloud resource management entails numerous challenges; see Armbrust et al. (2009) for a general overview, and Chen et al. (2020) for a survey from an Operations Management perspective. One way to classify the different works in this area is by referring to the time scale in which the resource management operates.

Management decisions that take place at a fast time scale (e.g., milliseconds) include cloud network routing, and container or Virtual Machine (VM) scheduling Maguluri et al. (2012); Stolyar and Zhong (2013); Buchbinder et al. (2021); Hadary et al. (2020). The VM allocation problem is online in nature and has similarities with the online dynamic bin packing problem.

In this work, we focus on a set of decisions that take place at a much slower time scale – planning for capacity expansion to accommodate increasing demand. Capacity planning includes a variety of operations, ranging from procurement and sourcing of hardware to the actual hardware deployment Chen et al. (2020). These operations have not drawn as much attention in the literature as the previously discussed topics. Arbabian et al. (2021) consider the expansion of CPU and RAM capacity in data centers via the deployment of servers in preconfigured “bundles”, each with a fixed ratio of CPU and RAM. The authors focus on the setting with two available preconfigurations and study capacity expansion policies assuming deterministic demand over a finite time horizon. Our work differs from these prior works in cloud resource management. First, in the cloud service deployment problem, for each demand request, we have a set of compatible suppliers, each of which can be used to offer the required capacity under supply availability constraints. Second, our goal is to optimize the broader deployment process: we decide in which data center each demand will be placed taking into account constraints such as data center throughput, and also account for the preparation of the data center rows. Finally, we study the server deployment process under demand uncertainty in order to account for demand variability, which contrasts with the deterministic nature of Arbabian et al. (2021).

Capacity Expansion. The main focus in the capacity expansion problems is to decide the right size, type, timing, and location of additional capacity that has to be acquired for satisfying future demands (see Manne (1967) and Luss (1982) for surveys on this topic). For instance, the owner of a manufacturing facility may wish to expand the size of a plant in order to increase its production capabilities; depending on the setting, these decisions may include multi-location, multi-type, and multi-period elements. Applications of this problem can be found in many industries (e.g., the automobile industry Eppen et al. (1989), telecommunications Chao et al. (2009)).

Similar to our problem, demand is often not fully known during planning, so different approaches have been proposed to deal with this uncertainty. Bean et al. (1992) consider a capacity expansion problem under certain assumptions on the stochastic demand process, and show that it can be formulated as a deterministic dynamic program. Eppen et al. (1989) propose a Mixed Integer Programming formulation based on a set of scenarios that capture the demand uncertainty, and discuss how risk can be taken into consideration. In Ahmed et al. (2003), the authors also use scenarios and formulate their problem as a multi-stage stochastic integer program. They then exploit a useful substructure which allows them to extent a well known tight reformulation that leads to fast convergence as part of a branch and bound algorithm. A survey of capacity expansion problems under uncertainty is provided by Van Mieghem (2003).

Our cloud server deployment problem resembles a capacity expansion problem at first glance. However, due to the long-lasting nature of cluster deployments, we consider each unit of "capacity", which is an available row, to be consumed by a demand, instead of to be merely "utilized" temporarily by a demand and reusable in the future by another demand. This is also one of the main differences between capacity expansion and inventory management problems, which we discuss next.

Inventory Management. A part of our cloud server deployment problem contains a multi-source inventory management problem, as we need to choose from the available cluster supply to satisfy the demand. Multi-source inventory management is the problem of deciding how to replenish inventory using multiple suppliers to minimize the operational costs, see for examples Song and Zipkin (2009); Song et al. (2021); Xin (2022). A better-understood special case of multi-source inventory management is the dual-sourcing inventory problem, where there are two possible sources of supply that may differ in their lead times and purchasing prices. Sheopuri et al. (2010) propose two new policy structures for the periodic-review dual-sourcing problem with stochastic demands; Xiong et al. (2022) study the dual sourcing problem with uncertainties in the demand and also in the purchasing price; finally, Song et al. (2017) identify the optimal policy structure for a dual-sourcing problem with Poisson demand and stochastic lead times.

To manage inventory from a robust optimization perspective, Bertsimas and Thiele (2006) show that the optimal policy for a robust optimization formulation of the inventory problem has base-stock policy structures. These base-stock structures were further characterized by Bienstock and Özbay (2008). Other works that take a robust optimization approach at inventory problems include Ben-Tal et al. (2005); Solyalı et al. (2016); Gorissen and den Hertog (2011); Ardestani-Jaafari and Delage (2016); Sun and Van Mieghem (2019); Dillon et al. (2017). Among these, Sun and Van Mieghem (2019) study a robust dual sourcing inventory problem.

Our problem shares similarities with the inventory management literature. However, the problem we study is more complex as we have multiple data centers for which we need to also make infrastructure building decisions in the first stage, while satisfying additional constraints such as the daily throughput limits.

Stochastic Programming. Stochastic programming is a mathematical optimization model for decision making when the uncertainty is characterized by random scenarios (or events), where the scenarios are assumed to be generated according to a probability distribution. Stochastic programs aim to find the best decision for a given preference to the scenarios. The extensive literature has considered both risk-neutral and risk-averse settings, see, e.g., Artzner et al. (1999); Shapiro et al. (2014) and references therein. Of particular relevance to our work is the class of two-stage stochastic programs, where the decision maker makes an initial set of decisions before any scenario is realized, and later on makes a new set of decisions tied to the observed scenario Birge and Louveaux (2011).

Our cloud server deployment problem is a two-stage stochastic programming problem, where we show in Section 5.4.1 that we can construct a minimum cost flow representation for the second stage problem. Bertsimas and Sim (2003) were the

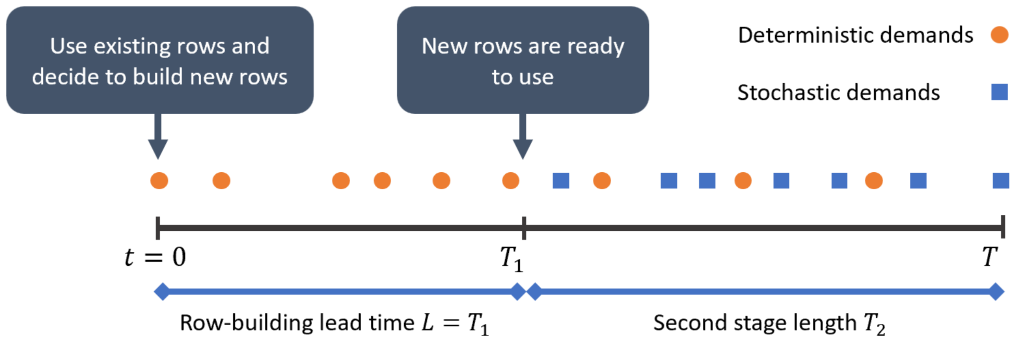

Use existing rows and decide to build new rows
New rows are ready to use
Deterministic demands ●
Stochastic demands ■
$t = 0$
T_1
T
Row-building lead time $L = T_1$
Second stage length T_2

Figure 5-1: Basic two-stage setting for the problem.

first to study static two-stage robust network flow problems. Atamtürk and Zhang (2007) study the two-stage robust network flow and design problem with uncertain demand. Bertsimas et al. (2013) develop approximate solutions to the two-stage robust maximum flow and minimum cut problems where the nodes and arcs might fail. Finally, Simchi-Levi et al. (2019) propose exact algorithms for the two-stage minimum cost flow problem with general polyhedral uncertainty set.

5.3 Problem Formulation

We first outline the temporal aspects of the problem below. See Figure 5-1 for an illustration.

The time horizon. Let T denote the horizon considered by the optimization of an instance. We refer to T as *the length of the instance*. The time at which the optimization takes place is defined as $t = 0$. At time $t = 0$, the provider is given a set $\mathcal{D}^1$ of deterministic demands that needs to be satisfied at different times in the horizon. Throughout the horizon, the provider may receive additional demands, which are stochastic demands from a known distribution. A demand that needs to be docked very soon is typically a realized demand. Therefore, we assume that there exists a time $T_1 < T$ such that all the demands with ideal dock dates prior to T_1 are deterministic. The demands with ideal dock dates after T_1 may be stochastic demands that arrive after $t = 0$, or as outlined above, they may be from the deterministic set $\mathcal{D}^1$ given at time $t = 0$. We use $\mathcal{D}^2$ to denote the set of demands that arrive after $t = 0$.

Building lead time. Recall that the cloud provider has to make two classes of decisions: (i) row-building decisions, and (ii) cluster assignment decisions. Building a row is a process that requires weeks to complete. We refer to the corresponding lead time as the *row-building lead time* and denote it by L. Due to this non-negligible lag, when making row-building decisions, the provider has to account for both the known, deterministic demands in the near future, as well as for the farther away stochastic demands.

For simplicity, we assume for now that the row-building lead time

$$L = T_1. \tag{5.1}$$

This implies that the demands that need to be docked before the new rows are ready are all from the deterministic demand set. We relax this assumption on the lead time in Appendix C.6.1.

The lag between the decision to build a row until it is ready, together with the stochastic nature of demands, motivates the formulation of the single instance optimization as a two stage stochastic optimization problem that we describe in Section 5.3.1.

Costs. We use $c_{d,t}$ to denote the cost of deploying demand d on day t; this captures the idle/delay costs associated with that deployment based on the ideal dock-date of the demand. If a demand d is not fulfilled, then it incurs a penalty u_d. The unit cost of shipping supply s to fulfill demand d is denoted by $h_{d,s}$, and the cost of building a new row in data center ℓ by b_ℓ.

Capacity and Throughput Constraints. For each data center $\ell \in \mathcal{L}$, capacity ζ_ℓ denotes the maximum number of rows that can exist in that data center.

In addition to the capacity of a data center, in cloud service operations, data center has a maximum throughput for docking for each day. Let $\mathcal{H}$ denote the set of throughput constraints. Instead of thinking of a throughput constraint as a limit on an individual data center, for practical operations of cloud services, this chapter considers a more general form of throughput constraints where we can place additional total throughput constraints on collections of data centers and vary over different days. In particular, each throughput constraint is denoted by (p, t), where p is a set of data centers p and t denotes the day for the constraint. We assume the following two properties of the throughput constraints:

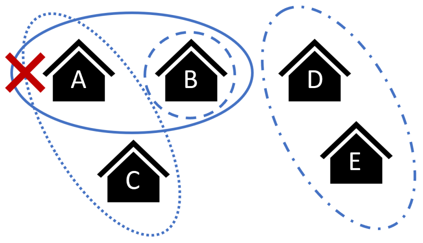
Property 1 *The throughput constraints satisfy the following* tree structure*:*

$$\textit{For any } (p_1, t), (p_2, t) \in \mathcal{H}, \textit{ either } p_1 \cap p_2 = \varnothing \textit{ or } p_1 \subseteq p_2 \textit{ or } p_1 \supseteq p_2.$$

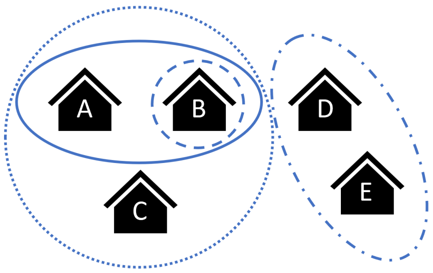
Property 2 *For any throughput constraint $(p, t) \in \mathcal{H}$, if a demand d is compatible with one of the data centers in p, then demand d is compatible with all the data centers in p, i.e.,*

$$\textit{For any } d \in \mathcal{D} \textit{ and any } (p, t) \in \mathcal{H}, \textit{ either } \mathcal{L}(d) \cap p = \varnothing \textit{ or } p \subseteq \mathcal{L}(d).$$

Property 1 and 2 are consistent with the fact that the throughput constraints on collections of data centers are often related to limited availability of professional



(a) Hierarchy not valid for $p_1 = \{A, B\}$ and $p_2 = \{A, C\}$.



(b) Valid throughput hierarchy.

Figure 5-2: Illustrative example of the throughput hierarchy on five data centers; each oval shape reflects a throughput constraint at some given time t.

labor and tools in the same region. Later we discuss possibilities of further relaxing our throughput constraints to even more diverse structures in Section 5.4.2.

An illustration of this property on an example of five data centers is provided in Figure 5-2.

5.3.1 A Two-Stage Stochastic Programming Formulation

In this section, we present a two-stage stochastic programming formulation for the problem. In the first stage, we make docking decisions for the near-future deterministic demands and decide how many new rows to build. In the second stage, we make docking decisions for the additional demands that arrive after $t = 0$.

Per standard stochastic programming terminology, the variables are split into two categories : state and stage variables.

State variables. The *state* variables capture the state of the system at the beginning of each stage. Initially, the state of the system is reflected in the currently available rows in each data center ℓ (denoted by ρ_ℓ), the available inventory of each supplier s (denoted by σ_s), and the throughput in each set p of data centers (denoted by $\delta_{p,t}$) which expresses the total number of demands the set p of data centers can deploy at time t. To distinguish between the two stages, we use subscripts 1 and 2 to indicate the stage of the state variables. Therefore, the system state at the first stage is given by $\rho_{\ell,1}, \sigma_{s,1}, \delta_{p,t,1}$ and fixed, while the state at the beginning of the second stage is represented by the state variables $\rho_{\ell,2}, \sigma_{s,2}, \delta_{p,t,2}$, which depend on the first-stage actions. Let ρ_2, σ_2, δ_2 denote the corresponding vectors of these state variables. Table 5.2 summarizes all the input parameters of the problem.

Stage variables. The *stage* variables capture decisions within a stage. We have three stage variable sets. For building new rows, each decision variable $x_\ell \in \mathbb{Z}_+$ indicates the number of new rows to build in data center ℓ. Row building decisions only happen in the first stage, so variables x_ℓ only appear in the first stage optimization. For assigning demands to clusters, each decision variable $z_{d,\ell,t} \in \{0,1\}$ indicates whether demand d is docked in data center ℓ on day t, and $w_{d,s} \in \{0,1\}$ indicates whether demand d is fulfilled using supplier s. Cluster assignment decisions happen in both the first and the second stages, so these variables appear in both stages. See Table 5.1 for a summary of decision variables used in the formulation.

Variable	Scope	Interpretation
$z_{d,\ell,t}$	stage	Demand d docks at data center ℓ on day t or not
$w_{d,s}$	stage	Demand d uses a cluster from supplier s or not
x_ℓ	stage	Number of rows to build in data center ℓ during the first stage
$\sigma_{s,2}$	state	Available units of supply s at the start of stage 2
$\rho_{\ell,2}$	state	Available number of rows in data center ℓ at the start of stage 2
$\delta_{p,t,2}$	state	Available number of demands the set of data centers p can deploy on day t in stage 2

Table 5.1: Decision variables in the formulation

Formulation. Let $\mathcal{D}^1$ denote the set of deterministic demand requests, $\mathcal{S}$ the set of suppliers, and $\mathcal{L}$ the set of data centers. For a demand d, let $\mathcal{S}(d)$ denote the set of its compatible suppliers; and let $\mathcal{L}(d)$ denote the set of its compatible data centers.

Parameter	Interpretation
$\mathcal{D}^1$	Set of demand requests
$\mathcal{L}$	Set of data centers
$\mathcal{L}(d)$	Set of data centers compatible with demand d
$\mathcal{S}$	Set of suppliers
$\mathcal{S}(d)$	Set of suppliers compatible with demand d
$\mathcal{T}_1$	Set of time periods of the first stage
$\mathcal{T}_2$	Set of time periods of the second stage
$\mathcal{T}$	Set of time periods of the full planning horizon, $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$
$\mathcal{H}$	Set of throughput constraints
$c_{d,t}$	Cost of docking demand d on day t due to delay or idling
u_d	Cost of not fulfilling demand d
$h_{d,s}$	Cost of shipping a unit of supply s to fulfill demand d
b_ℓ	Cost of building one new row at data center ℓ
ζ_ℓ	Capacity of data center ℓ
$\sigma_{s,1}$	The supplier inventory from supplier s in the first stage
$\rho_{\ell,1}$	The number of available rows at data center ℓ in the first stage
$\delta_{p,t,1}$	The total throughput limit on the set of data centers p can deploy on day t

Table 5.2: Input parameters used in the formulation

We use $Q(\cdot, \xi)$ to denote the optimal value of the second stage problem as a function of the state variable vectors δ_2, ρ_2, σ_2, where ξ is the random vector associated with the stochastic demands and is explained below.

Let $\mathcal{R}$ denote a risk measure that the cloud service provider adopts to evaluate the cost of the second stage program in relation to the first stage costs. In this work, we allow some amount of flexibility over the choice of the risk measure, under the assumption that it is well-defined for any possible realized δ_2, ρ_2 and σ_2. Some example risk measures in our consideration are *Expectation*, *Conditional Value-at-Risk*, and *Mean-Deviation.* Our optimization problem is of the following form:

$$
\begin{aligned}
\min \sum_{d \in \mathcal{D}} & \left[\overbrace{\sum_{\ell \in \mathcal{L}(d)} \sum_{t \in \mathcal{T}} c_{d,t} z_{d,\ell,t}}^{\text{idle cost}} + \overbrace{u_d \left(1 - \sum_{\ell \in \mathcal{L}(d)} \sum_{t \in \mathcal{T}} z_{d,\ell,t} \right)}^{\text{cost of missed demand}} + \overbrace{\sum_{s \in \mathcal{S}(d)} h_{d,s} w_{d,s}}^{\text{shipping cost}} \right] \\
& + \overbrace{\sum_{\ell \in \mathcal{L}} b_\ell x_\ell}^{\text{building cost}} + \overbrace{\mathcal{R}[Q(\delta_2, \rho_2, \sigma_2, \xi)]}^{\text{cost of the second stage}} && (5.2) \\
\text{s.t.} & \sum_{\ell \in \mathcal{L}(d)} \sum_{t \in \mathcal{T}} z_{d,\ell,t} \leq 1 && \forall d \in \mathcal{D}^1 \quad (5.3) \\
& \sum_{\ell \in \mathcal{L}(d)} \sum_{t \in \mathcal{T}} z_{d,\ell,t} = \sum_{s \in \mathcal{S}(d)} w_{d,s} && \forall d \in \mathcal{D}^1 \quad (5.4) \\
& \delta_{p,t,1} \geq \sum_{d \in \mathcal{D}^1} \sum_{\ell \in p \cap \mathcal{L}(d)} z_{d,\ell,t} && \forall (p,t) \in \mathcal{H} : t \in \mathcal{T}_1 \quad (5.5) \\
& \delta_{p,t,2} = \delta_{p,t,1} - \sum_{d \in \mathcal{D}^1} \sum_{\ell \in p \cap \mathcal{L}(d)} z_{d,\ell,t} && \forall (p,t) \in \mathcal{H} : t \in \mathcal{T}_2 \quad (5.6) \\
& \rho_{\ell,1} \geq \sum_{d \in \mathcal{D}^1 : \ell \in \mathcal{L}(d)} \sum_{t \in \mathcal{T}_1} z_{d,\ell,t} && \forall \ell \in \mathcal{L} \quad (5.7) \\
& \rho_{\ell,2} = \rho_{\ell,1} + x_\ell - \sum_{d \in \mathcal{D}^1 : \ell \in \mathcal{L}(d)} \sum_{t \in \mathcal{T}} z_{d,\ell,t} && \forall \ell \in \mathcal{L} \quad (5.8) \\
& \zeta_\ell \geq \rho_{\ell,1} + x_\ell && \forall \ell \in \mathcal{L} \quad (5.9) \\
& \sigma_{s,2} = \sigma_{s,1} - \sum_{d \in \mathcal{D}^1} w_{d,s} && \forall s \in \mathcal{S} \quad (5.10) \\
& \rho_{\ell,2}, \delta_{p,t,2}, \sigma_{s,2} \geq 0, && (5.11) \\
& z_{d,\ell,t}, w_{d,s} \in \{0,1\}, x_\ell \in \mathbb{Z}_+. && (5.12)
\end{aligned}
$$

In constraints (5.3), at most one data center and day is selected for each demand, while (5.4) ensures that a demand docks if and only if a compatible supplier has been selected. The throughput constraints for each set of data centers p during $\mathcal{T}_1$ are enforced in (5.5), while (5.6) makes any remaining throughput capacity during $\mathcal{T}_2$ available in the second stage. According to (5.7), only the existing rows can be used to dock demands during the first stage; any remaining rows and the newly built ones become available in the second stage to deploy the unseen demands (5.8). The total amount of rows in each data center is limited by the data center capacity in (5.9). Finally, (5.10) enforces the supplier inventory constraints, and makes the remaining supplier inventory available to the second stage. The objective function (5.2) consists of the delay/idle costs of the demands, penalties if demands are not fulfilled, shipping costs, and the cost of building new rows.

$$Q(\delta_2, \rho_2, \sigma_2, \xi) = \min \sum_{d \in \mathcal{D}^2} \left[\overbrace{\sum_{\ell \in \mathcal{L}(d)} \sum_{t \in \mathcal{T}_2} c_{d,t} z_{d,\ell,t}}^{\text{idle cost}} + \overbrace{u_d \left(1 - \sum_{\ell \in \mathcal{L}(d)} \sum_{t \in \mathcal{T}_2} z_{d,\ell,t} \right)}^{\text{cost of missed demand}} + \overbrace{\sum_{s \in \mathcal{S}(d)} h_{d,s} w_{d,s}}^{\text{shipping cost}} \right] \tag{5.13}$$

$$\text{s.t.} \quad \sum_{\ell \in \mathcal{L}(d)} \sum_{t \in \mathcal{T}_2} z_{d,\ell,t} \leq 1 \qquad \forall d \in \mathcal{D}^2 \tag{5.14}$$

$$\sum_{\ell \in \mathcal{L}(d)} \sum_{t \in \mathcal{T}_2} z_{d,\ell,t} = \sum_{s \in \mathcal{S}(d)} w_{d,s} \qquad \forall d \in \mathcal{D}^2 \tag{5.15}$$

$$\delta_{p,t,2} \geq \sum_{d \in \mathcal{D}^2} \sum_{\ell \in p \cap \mathcal{L}(d)} z_{d,\ell,t} \qquad \forall (p,t) \in \mathcal{H} : t \in \mathcal{T}_2 \tag{5.16}$$

$$\rho_{\ell,2} \geq \sum_{d \in \mathcal{D}^2 : \ell \in \mathcal{L}(d)} \sum_{t \in \mathcal{T}_2} z_{d,\ell,t} \qquad \forall \ell \in \mathcal{L} \tag{5.17}$$

$$\sigma_{s,2} \geq \sum_{d \in \mathcal{D}^2} w_{d,s} \qquad \forall s \in \mathcal{S} \tag{5.18}$$

$$z_{d,\ell,t}, w_{d,s} \in \{0,1\}. \tag{5.19}$$

In the second stage, the cloud service provider can no longer make decisions to build more rows. The randomness associated with ξ is the set of stochastic demands $\mathcal{D}^2$ and parameters $c_{d,t}, u_d, h_{d,s}$ of each demand $d \in \mathcal{D}^2$. Deviating from the conventional modeling, the second stage problem is a mixed integer linear program with no fixed dimension, because we do not cap the number of demands that may appear. Note that $Q(\cdot, \xi)$ has a practical meaning only on the nonnegative integral vectors.

Due to (5.14) and (5.15), a single data center, day, and supplier are chosen for the demands that dock successfully. The throughput constraints are enforced in (5.16), the row availability constraints in (5.17), and the supply availability in (5.18).

Since $\sum_{d\in\mathcal{D}^2} u_d$ is constant, for any translation-invariant risk measure $\mathcal{R}$, we can subtract $\sum_{d\in\mathcal{D}^2} u_d$ from $Q(\delta_2, \rho_2, \sigma_2, \xi)$ without affecting the solution for our entire problem. For risk measures that are not translation-invariant, we omit subtracting this constant from $Q(\delta_2, \rho_2, \sigma_2, \xi)$ above, and all the theoretical results still hold with slight modifications. Since all the risk measures discussed in Section 5.5.2 are translation-invariant, we go ahead with the subtraction for simplicity:

$$Q(\delta_2, \rho_2, \sigma_2, \xi) = \min \sum_{d\in\mathcal{D}^2} \left[\overbrace{\sum_{\ell\in\mathcal{L}(d)} \sum_{t\in\mathcal{T}_2} c_{d,t} z_{d,\ell,t}}^{\text{idle cost}} + \overbrace{u_d \left(- \sum_{\ell\in\mathcal{L}(d)} \sum_{t\in\mathcal{T}_2} z_{d,\ell,t} \right)}^{\text{cost of missed demand}} + \overbrace{\sum_{s\in\mathcal{S}(d)} h_{d,s} w_{d,s}}^{\text{shipping cost}} \right] \tag{5.20}$$

$$\text{s.t. constraints (5.14)-(5.19) hold.} \tag{5.21}$$

When the risk measure $\mathcal{R}$ is taken to be the expectation operator $\mathbb{E}$, the formulation is risk neutral; that is, the outcomes of $Q(\cdot, \xi)$ are summed/integrated according to the distribution of ξ. When $\mathcal{R}$ is some other risk measure, it may shift more weight to bad outcomes, i.e., outcomes having relatively high values. These formulations are risk averse, as they in some sense penalize solutions with more bad outcomes. The monotonicity and convexity of risk measures plays an important role in computation, since the composition $\mathcal{R}[Q(\cdot, \xi)]$ is convex whenever $Q(\cdot, \xi)$ is convex for each outcome of ξ. We present some examples of popular monotone and convex risk measures in Section 5.5 and demonstrate their effectiveness in numerical experiments in Section 5.6.

5.4 A Convex Approximation

Cutting plane methods as a family of algorithms has proven to be efficient for solving large scale stochastic optimization problems (Birge and Louveaux, 2011). In our case however, the function $Q(\cdot, \xi)$ is in general nonconvex and discontinuous, which does not allow applying cutting plane methods directly. A possible workaround is to find a good convex approximation of Q. For our problem setting, we show in Section 5.4.1 that the LP relaxation of the second stage problem associated with Q yields the same optimal value when the state variables $\delta_{p,t,2}, \rho_{\ell,2}, \sigma_{s,2}$ are integral. Let us denote $\tilde{Q}(\cdot, \xi)$ to be the optimal value of the LP relaxation of the second stage problem, then, for each outcome of ξ, the function $\tilde{Q}(\cdot, \xi)$ is convex and it coincides with $Q(\cdot, \xi)$ on the integral vectors. In Section 5.4.2, we study the generalization of our problem where either Property 1 or Property 2 does not hold; we prove two hardness results in this case showing NP-hardness and the existence of an integrality gap respectively.

5.4.1 Tightness of LP Relaxation

Before we show the tightness of the LP relaxation, let us first recall the settings of minimum cost flow problems which will play an important role in obtaining our results.

Definition 1 (Korte and Vygen (2018)) *Given a digraph G, capacities $\kappa : E(G) \to \mathbb{R}_+$, and numbers $b : V(G) \to \mathbb{R}$ with $\sum_{v \in V(G)} b(v) = 0$, a b-flow in (G, κ) is a function $f : E(G) \to \mathbb{R}_+$ with $f(e) \leq \kappa(e)$ for all $e \in E(G)$ and $\sum_{e \in \delta^+(v)} f(e) - \sum_{e \in \delta^-(v)} f(e) = b(v)$ for all $v \in V(G)$. A b-flow with $b \equiv 0$ is a circulation.*

Given weights $c : E(G) \to \mathbb{R}$, the minimum cost flow problem is to find a b-flow

f that minimizes $\sum_{e \in E(G)} f(e)c(e)$ or decides that none exists.

Theorem 8 (Korte and Vygen (2018)) *Let (G, κ, b, c) be an instance of the minimum cost flow problem, where κ and b are integral. If there exists a b-flow in (G, κ), then there exists a minimum cost b-flow which is integral.*

Representation Network: For any program in the form of $Q(\delta_2, \rho_2, \sigma_2, \xi)$, we create a representation network G as follows: the node set $V(G)$ contains a node for each supplier s, a node for each data center ℓ, a source node, a sink node, a pseudo-node for each demand d, and a pseudo-node for each throughput constraint (p, t). The pseudo-node for demand d consists of two nodes that represent the head and tail of the demand d, connected by an edge of capacity 1 and with a unit cost of $-u_d$. The pseudo-node for throughput constraint (p, t) consists of two nodes that represent the head and tail of the throughput constraint (p, t), connected by an edge of capacity of $\delta_{p,t,2}$.

The arc set $E(G)$ of the network contains directed arcs from:

- the source to each supplier s, with a capacity of $\sigma_{s,2}$;
- each supplier s to each compatible demand d, with a unit cost of $h_{d,s}$;
- each demand d to each throughput (p, t) if demand d is compatible with all data centers in p and $\nexists (p_1, t) \in \mathcal{H}$ such that $p_1 \supset p$. The unit cost of the arc is $c_{d,t}$;
- each throughput (p, t) to each throughput (p', t), if $p \supset p'$ and $\nexists (p'', t) \in \mathcal{H}$ such that $p \supset p'' \supset p'$;
- each throughput (p, t) to each data center ℓ if $\ell \in p$ and $\nexists (p', t) \in \mathcal{H}$ such that $\ell \in p' \subset p$;

- each data center ℓ to the sink, with a capacity of $\rho_{\ell,2}$;

- the sink to the source.

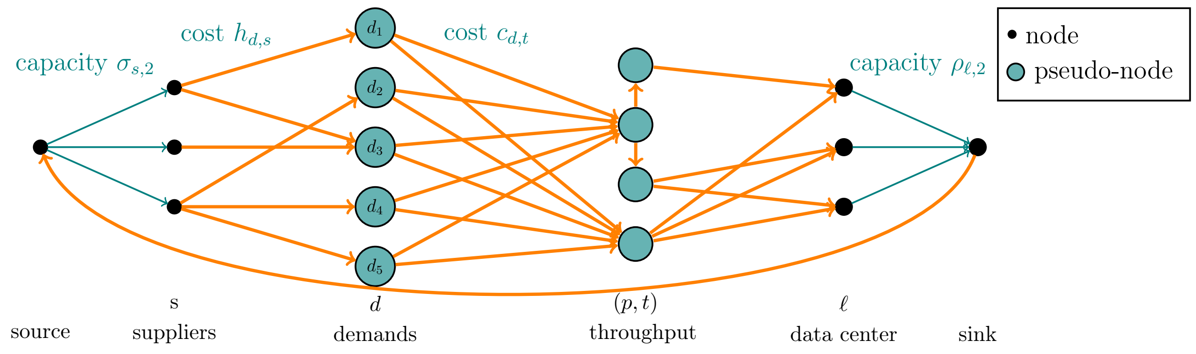


Figure 5-3: An example of a representation network for the second stage program.

Note that our construction of the representation network assumes Properties 1 and 2. The construction of the demand pseudo-nodes makes sure that at most 1 unit of flow can pass through any demand node, and there is a cost associated with any flow that passes through the demand node. The construction of the throughput pseudo-nodes makes sure that at most $\delta_{p,t,2}$ unit of flow passes through the throughput node for (p,t).

Lemma 8 *Consider the above network (see Figure 5-3). For any $C \geq 0$, any feasible solution of the LP relaxation of cost C can be mapped to a feasible flow of the same cost C. Reversely, any feasible flow of cost C can be mapped to a feasible solution of the LP relaxation that has the same cost C.*

See proof in Appendix C.2. Now we are ready to show that the LP relaxation $\tilde{Q}(\cdot,\xi)$ of the second stage problem produces a good convex approximation of the function $Q(\cdot,\xi)$.

Theorem 9 *For any given outcome ξ, the LP relaxation $\tilde{Q}(\cdot,\xi)$ is a convex function that agrees with the second stage problem $Q(\cdot,\xi)$ on the integral vectors.*

Proof: The LP relaxation of the second stage problem is a partial linear minimization given the state variables $\delta_{p,t,2}, \rho_{\ell,2}$, and $\sigma_{s,2}$, thus $\tilde{Q}(\cdot,\xi)$ is convex in $(\delta_{p,t,2}, \rho_{\ell,2}, \sigma_{s,2})$. Since $\delta_{p,t,2}, \rho_{\ell,2}$, and $\sigma_{s,2}$ are integers for our application, we invoke Theorem 8 to obtain an optimal integral flow of the corresponding minimum cost circular flow problem. Note that the arcs with infinite capacity can be modified to have capacity of some large integers so that it would not alter the optimal solution. By Lemma 8, an optimal integral flow can be mapped to an optimal integral solution of the LP relaxation, which implies that $\tilde{Q}(\cdot,\xi)$ agrees with $Q(\cdot,\xi)$ on the integral vectors. $\square$

Remark 3 *Note that a sufficient condition that is often used to show tightness of an. LP relaxation is the constraint matrix of the formulation being totally unimodular (Korte and Vygen (2018)). However, that property does not necessarily hold in our problem. We show this in Appendix C.1 by constructing an objective function where the optimal solution can only be fractional.*

Since the parameters can only take integral values in our problem, the function $Q(\cdot,\xi)$ can be safely replaced by its convex approximation $\tilde{Q}$ for computational purposes. This enables solving the second stage efficiently by either using a polynomial time algorithm for the corresponding minimum cost flow problem (see Schrijver (2003); Chen et al. (2022)), or solving the LP relaxation directly. In our implementation, we choose to solve the dual of the LP relaxation , because the dual optimal solution is useful for generating cutting planes (see Appendix C.5). The overall formulation now becomes a mixed integer convex program, which is in general computationally easier and allows us to employ Benders decomposition (Benders (1962)) for the large-scale problem.

5.4.2 Two Hardness Results

As stated in Section 5.3.1, this chapter considers more general throughput constraints than the traditional throughput constraints on individual facility locations. In particular, we allow any time-varying throughput constraints on collections of data centers as long as they satisfy Properties 1 and 2, which are compatible with the typical local/regional professional labor and tooling limits that a cloud service provider might work with. Now, one may ask, can we further generalize the throughput constraints in our model formulation?

In our investigation, we find out that the LP relaxation approach is not effective in the absence of either Property 1 or Property 2. In this section, we establish two hardness results regarding the second stage problem. We provide the proofs for the following two lemmas in the Appendix.

Lemma 9 *With only Property 2, the second stage problem is both NP-hard and hard to approximate (i.e., there exists no efficient constant-factor approximation algorithm unless $P = NP$).*

Basically, Lemma 9 says that solving the second stage problem alone is very difficult when the hierarchical structure of the throughput constraints is missing. The stochastic optimization setting further aggravates the situation, since we need to solve at least one instance of the second stage problem for each outcome ξ. Nevertheless, such structure alone does not guarantee the tightness of LP relaxation.

Lemma 10 *With only Property 1, the integrality gap of the second stage problem (without removing the constant in the objective) is at least* $\frac{4}{3}$.

The proofs for the above two lemmas are provided in Appendix C.3 and C.4.

5.5 Algorithm

Substituting Q with $\tilde{Q}$ enables us to design an efficient cutting-plane algorithm (Section 5.5.1) that exploits methods from convex optimization to solve the problem efficiently for a variety of risk measures. We conclude this section by describing three concrete risk measures (in Section 5.5.2) that will be used in our evaluation in Section 5.6.

5.5.1 Hybrid Level-Benders Algorithm

Main ideas. The core of our algorithm is a cutting plane scheme that is guaranteed to obtain the optimal solution within a finite number of iterations. To design the scheme, we consider the following problem MasterIP which is derived from the stochastic formulation (5.2)-(5.12) by replacing $\mathcal{R}[\tilde{Q}(\delta_2, \rho_2, \sigma_2, \xi)]$ with a variable θ.

$$\begin{aligned}
(\textsc{MasterIP}) \quad \min \quad & \sum_{d \in \mathcal{D}^1} \Bigg[\sum_{\ell \in \mathcal{L}(d)} \sum_{t \in \mathcal{T}} c_{d,t} z_{d,\ell,t} + u_d \left(1 - \sum_{\ell \in \mathcal{L}(d)} \sum_{t \in \mathcal{T}} z_{d,\ell,t} \right) \\
& + \sum_{s \in \mathcal{S}(d)} h_{d,s} w_{d,s} \Bigg] + \sum_{\ell \in \mathcal{L}} b_\ell x_\ell + \theta && (5.22) \\
\text{s.t.} \quad & \text{constraints } (5.3) - (5.12). && (5.23)
\end{aligned}$$

Our algorithm is an iterative algorithm that starts with the MasterIP and proceeds by generating cuts, i.e., additional constraints that reduce the size of the feasible solution set without excluding the optimal solution of the problem. In our particular case, the cut generation process will ensure that, in the optimal solution, θ is always a lower approximation of the function $\mathcal{R}[\tilde{Q}(\cdot,\xi)]$. The algorithm keeps refining that lower convex approximation and eventually finds a (near) optimal solution when the approximation is good enough.

To generate the cuts, in each iteration i, we select a trial point $(\delta_2^i, \rho_2^i, \sigma_2^i)$ corresponding to a set of decisions for the state variables. By the definition of convexity of $\mathcal{R}[\tilde{Q}(\cdot,\xi)]$, we have

$$\mathcal{R}[\tilde{Q}(\delta_2,\rho_2,\sigma_2,\xi)] \geq \mathcal{R}[\tilde{Q}(\delta_2^i,\rho_2^i,\sigma_2^i,\xi)] + (\nabla^i)^\intercal(\delta_2-\delta_2^i, \rho_2-\rho_2^i, \sigma_2-\sigma_2^i) \quad \forall \delta_2,\rho_2,\sigma_2, \tag{5.24}$$

where ∇^i is a subgradient of $\mathcal{R}[\tilde{Q}(\cdot,\xi)]$ at $(\delta_2^i, \rho_2^i, \sigma_2^i)$ and $\mathcal{R}[\tilde{Q}(\delta_2^i,\rho_2^i,\sigma_2^i,\xi)]$ is the value of the function $\mathcal{R}[\tilde{Q}(\cdot,\xi)]$ evaluated at $(\delta_2^i, \rho_2^i, \sigma_2^i)$. Hence, our algorithm proceeds iteratively by generating a cut in each iteration i, given by

$$\theta \geq \mathcal{R}[\tilde{Q}(\delta_2^i,\rho_2^i,\sigma_2^i,\xi)] + (\nabla^i)^\intercal(\delta_2-\delta_2^i, \rho_2-\rho_2^i, \sigma_2-\sigma_2^i), \tag{5.25}$$

and adding the cut to the MasterIP. Note that at any point of the algorithm, the lower approximation of $\mathcal{R}[\tilde{Q}(\cdot,\xi)]$ is given by

$$\max_i \left\{ \mathcal{R}[\tilde{Q}(\delta_2^i,\rho_2^i,\sigma_2^i,\xi)] + (\nabla^i)^\intercal(\delta_2-\delta_2^i, \rho_2-\rho_2^i, \sigma_2-\sigma_2^i) \right\} \quad \forall \delta_2,\rho_2,\sigma_2. \tag{5.26}$$

The sequence of trial points $(\delta_2^i, \rho_2^i, \sigma_2^i)$ based on which the cuts are generated can have great impact on the quality of the cuts, which directly affects the number of iterations required for the algorithm's convergence. At the same time, the speed of generating cuts is important, as slow cut generation makes each iteration of the algorithm more time consuming, affecting the overall run-time. To address this quality-performance tradeoff, we design a *hybrid* strategy to obtain the sequence of trial points. Although fractional solutions are not feasible for our original problem, we initially allow the trial points to be fractional. This enables generating an adequate set of cuts quickly, without solving a time-consuming MIP. Once this initial set of cuts has been generated, we switch to enforcing the integrality constraints and obtain additional cuts until the optimal solution for our original problem has been found.

Algorithm details. As described above, our algorithm has two main steps, corresponding to different methodologies for obtaining trial points. The first step follows the Level method (Lemaréchal et al. (1995)), hence termed LevelStep; the second step follows the Benders decomposition (Benders (1962)), hence termed BendersStep. Accordingly, we refer to our entire algorithm as the *Hybrid Level-Benders Algorithm.* We provide below the details for the two steps (note that each of the steps is an iterative process).

ALGORITHM 5: Hybrid Level-Benders Algorithm

```
1 Initialize MasterIP (5.22)-(5.23).
2 LevelStep: Relax the integrality constraints in MasterIP and use an
    auxiliary problem based on the Level method (Lemaréchal et al. (1995)) to
    generate cuts until the LP relaxation of our problem is solved to
    $\epsilon$-optimality. Add the cuts to MasterIP.
3 BendersStep: Continue generating cuts for MasterIP focusing on solely
    integral trial points based on Benders decomposition. Terminate when the
    optimal solution for our original problem has been found.
```

LevelStep: We consider the LP relaxation of our problem and define the MasterLP by relaxing the integrality constraints of MASTERIP as follows:

$$
\begin{aligned}
(\textsc{MasterLP}) \quad \min \quad & \sum_{d \in \mathcal{D}^1} \left[\sum_{\ell \in \mathcal{L}(d)} \sum_{t \in \mathcal{T}} c_{d,t} z_{d,\ell,t} + u_d \left(1 - \sum_{\ell \in \mathcal{L}(d)} \sum_{t \in \mathcal{T}} z_{d,\ell,t} \right) \right. \\
& \left. + \sum_{s \in \mathcal{S}(d)} h_{d,s} w_{d,s} \right] + \sum_{\ell \in \mathcal{L}} b_\ell x_\ell + \theta \qquad (5.27) \\
\text{s.t.} \quad & \text{constraints } (5.3) - (5.11) \\
& z_{d,\ell,t}, w_{d,s}, x_\ell \geq 0.
\end{aligned}
$$

Note that an upper bound of 1 is not required in the relaxation for the binary variables as it is implicitly enforced via the remaining constraints. Starting with MASTERLP, we utilize ideas from the Level method (Lemaréchal et al. (1995)) to generate a set of cuts that eventually solve the LP relaxation of our problem to (near) optimality. At each of its iterations, the method tries to keep the next trial point close to the current trial point, to maintain adequate lower approximation for the function $\mathcal{R}[\tilde{Q}(\cdot, \xi)]$. This is achieved through the use of an auxiliary Level problem that penalizes trial points that are further away. This step terminates once an ϵ-optimal solution has been found for the LP relaxation of our problem; at this point, the set of cuts that has been generated for MASTERLP is being added to MASTERIP.

BendersStep: This step is based on Benders decomposition Benders (1962). Now, we focus on obtaining solely integral trial points. In each iteration, we obtain the next trial point by solving MASTERIP to optimality. In contrast to classic Benders decomposition, which typically starts with a problem of the form (5.22)-(5.23), our MASTERIP has already been enhanced by a set of cuts during the LevelStep when this step begins. This step terminates with an optimal solution for our original problem.

Subgradients: To complete the description of our algorithm we need to describe how the subgradients ∇^i in (5.25) are derived for the different risk measures; see Appendix C.5 for the details.

Optimality. We have the following guarantees.

Theorem 10 *For convex risk measures $\mathcal{R}$, the Hybrid Level-Benders Algorithm converges to the optimal solution in a finite number of iterations.*

Proof: Here we provide a proof sketch: since $\tilde{Q}(\cdot,\xi)$ is convex, for convex risk measures $\mathcal{R}$ the composition $\mathcal{R}[\tilde{Q}(\cdot,\xi)]$ is also convex. As a result, the subgradient exists and, by definition of convexity, adding cuts of the form (5.25) in MasterIP results in a lower approximation of $\mathcal{R}[\tilde{Q}(\cdot,\xi)]$ (5.26). By construction, these cuts are valid on both fractional and integral trial points and in both cases provide a lower approximation of $\mathcal{R}[\tilde{Q}(\cdot,\xi)]$. As a result, the cuts that are generated for the LP relaxation during the LevelStep are valid cuts when we restrict $\mathcal{R}[\tilde{Q}(\cdot,\xi)]$ to only integral values of the state variables.

We now need to show that the algorithm converges to the optimal solution in a finite number of iterations. The convergence of the LevelStep is obtained directly by the convergence guarantee of the Level method. Similarly, for the BendersStep, it can be shown that the Benders decomposition never visits a suboptimal solution twice. Note that the first stage decision variables in our problem are bounded and integral, hence the procedure finds an optimal solution in a finite number of iterations. Finally, recall that in Section 5.4.1 it was shown that $\tilde{Q}(\cdot,\xi)$ has the same value as $Q(\cdot,\xi)$ when the state variables $\delta_{p,t,2}, \rho_{\ell,2}, \sigma_{s,2}$ are integral. As a result, the optimal solution obtained through this algorithm that considers the relaxation $\tilde{Q}(\cdot,\xi)$ is also optimal for our original problem with $Q(\cdot,\xi)$. $\square$

5.5.2 Risk Measures

We now present some concrete risk measures that will be used in Section 5.6 for the numerical experiments. We consider three monotone and convex risk measures $\mathcal{R}$: expectation, conditional Value-at-Risk, and mean-deviation risk measures of order 1. Let $(\xi_1, \ldots, \xi_N)$ denote the N demand scenarios, and let $f_i(x) = \tilde{Q}(x, \xi_i)$ for each $i = 1, \ldots, N$.

Expectation. The expectation is given by $\mathbb{E}[\tilde{Q}(x,\xi)] = N^{-1}\sum_{i=1}^{N} f_i(x)$. Expectation is a common measure when users are risk neutral.

Conditional Value-at-Risk (CVaR). The CVaR metric stands for the expected loss (or cost) at the tail, namely the expected loss of the "worst" scenarios Rockafellar and Uryasev (2000) (i.e., the α-percentile of scenarios with the highest cost). Formally, for a threshold $\alpha \in (0,1)$, the Conditional Value-at-Risk is defined as

$$\mathcal{R}_\alpha(Z) = \alpha^{-1} \int_{1-\alpha}^{1} VaR_{1-t}(Z)\, dt,$$

where

$$VaR_\alpha(Z) = \inf\{t : Pr(Z \le t) \ge 1 - \alpha\}$$

is the Value-at-Risk. Then, for the composition $\rho_\alpha(x) = \mathcal{R}_\alpha[\tilde{Q}(x,\xi)]$ we have:

$$\rho_\alpha(x) = \mathcal{R}_\alpha[\tilde{Q}(x,\xi)] = \frac{1}{\alpha N}\sum_{i=1}^{\lfloor \alpha N \rfloor} f_{(i)}(x) + \left(1 - \frac{\lfloor \alpha N \rfloor}{\alpha N}\right) f_{(\lfloor \alpha N \rfloor + 1)}(x),$$

where (i) is an ordering of $1, \ldots, N$ (depending on x) such that $f_{(i_1)}(x) \ge f_{(i_2)}(x)$ whenever $i_1 < i_2$.

Mean-deviation risk measure of order 1 **(MeanDev)**. Balancing expectation and deviation, the mean-deviation risk measure allows users to optimize the average and the average absolute deviation at the same time (Shapiro et al., 2021, Chapter 6). For a threshold $c \in [0, 1/2]$, consider

$$\mathcal{R}_c(Z) = \mathbb{E}[Z] + c\mathbb{E}|Z - \mathbb{E}[Z]|.$$

The composition $\rho_c(x) = \mathcal{R}_c[\tilde{Q}(x, \xi)]$ is then

$$\begin{aligned}
&\rho_c(x) \\
&= N^{-1} \sum_{i=1}^{N} f_i(x) + cN^{-1} \sum_{i=1}^{N} \left| f_i(x) - N^{-1} \sum_{i=1}^{N} f_i(x) \right| \\
&= N^{-1} \sum_{i=1}^{N} f_i(x) + cN^{-1} \sum_{i=1}^{N} \max \left(f_i(x) - N^{-1} \sum_{i=1}^{N} f_i(x), N^{-1} \sum_{i=1}^{N} f_i(x) - f_i(x) \right) \\
&= \max_{b \in \{\pm 1\}^N} N^{-1} \sum_{i=1}^{N} \left(f_i(x) + cb_i \left(f_i(x) - N^{-1} \sum_{i=1}^{N} f_i(x) \right) \right) \\
&=: \max_{b \in \{\pm 1\}^N} g_b(x).
\end{aligned}$$

It can be verified that each g_b is a convex combination of $f_i's$, so g_b is also convex. A lower c value results in more emphasis in minimizing expectation, and a higher c value results in stronger preference over minimizing deviations.

5.6 Numerical Experiments

We conduct numerical experiments to evaluate our algorithms under a variety of risk measures, using real data from a large cloud provider, Microsoft Azure.

5.6.1 Dataset

Our raw data consists of historical demands and projections of future demands over time from Microsoft Azure. Projections take place at regular time intervals (e.g. every hour), and capture the expected demand over the planning time-horizon (e.g. three months). How these future demand projections are generated will not be revealed; however, it is known that a good number of methods, such as Sample Average Approximation (SAA), could obtain a reasonably good projection under some conditions (Kleywegt et al. (2002)). Naturally, for a given time, we use the realized demands by that time as the deterministic demands, and use the projected demands to form our knowledge of the stochastic future demand.

We think of each time series of projected demands as a *scenario*, i.e., a possible demand realization for the next few months. We gather all time series of projected demands to form an empirical distribution of scenarios. We are working towards releasing this dataset to be available to the research community.

5.6.2 Setup

All algorithms were implemented in Python, and all experiments were run using Gurobi 9.5.1 on a Desktop with 2.6GHz CPU. In our implementation of the Hybrid Level-Benders Algorithm, the LevelStep terminates when an ϵ-optimal solution has been found for the LP relaxation with $\epsilon = 1$. Unless otherwise specified, we use the default threshold values of $\alpha = 0.5$ for the CVaR and $c = 0.5$ for the MeanDev.

5.6.3 Benchmarks

We compare the performance of our algorithm against two benchmark algorithms. The first benchmark algorithm *Deterministic Baseline Algorithm* is closely related to the current approach in production by Microsoft Azure. The second one *Scenario Based Algorithm* is an adapted version of the former that take into consideration the stochastic nature of the demands.

Deterministic Baseline Algorithm (DBA). DBA is a popular approach that obtains the optimal deployment plan for each instance assuming that the demand is fully deterministic. In our experiments, DBA selects each scenario as its belief about the future demands, and optimize the two-stage mixed integer program based on this deterministic belief. We evaluate the quality of the first stage decisions on all of the other possible scenarios, and then aggregate the resulting costs according to the different risk measures under consideration. For statistical significance, we repeat the above procedure where DBA takes each of the scenarios as its belief, and output the average cost over all these scenarios.

Scenario Based Algorithm (SBA). In contrast to DBA, which assumes a single deterministic scenario for future demands, SBA is an adaptation that assumes access to the full set of scenarios. SBA selects the scenario that leads to the minimum overall cost as its belief on the future demands. In short, SBA is the top-performing DBA.

We provide the pseudo-code for both DBA and SBA in Algorithm 6.

ALGORITHM 6: Benchmark algorithms DBA and SBA

```
1 for each scenario $i$ do
2     Set $i$ as the ground truth scenario; $\xi^G := i$
3     Solve (5.2)-(5.19) with the single scenario $\xi^G$ in the second stage; let
        $(\delta_2^*, \rho_2^*, \sigma_2^*)$ denote the optimal values of the state variable vectors.
4     Calculate the total cost (5.2) with the risk measure now applied to the
        full set of scenarios $\mathcal{R}[Q(\delta_2^*, \rho_2^*, \sigma_2^*, \xi)]$; let $V_i$ denote this total cost.
5 end
6 return $\mathbb{E}[\{V_i\}_i]$ for DBA    or    $\min_i V_i$ for SBA
```

Performance Measure. For each experiment point, we compute the percentage improvement of our algorithm over DBA and SBA, defined by

$$\text{improvement} := \frac{\text{benchmark algorithm's cost} - \text{our algorithm's cost}}{\text{benchmark algorithm's cost}} \cdot 100\%.$$

Performance Under Uncertainty and Scarcity.

In the first set of experiments, each experiment point that we evaluate our algorithm on varies in degree of uncertainty and the degree of supply availability.

In terms of the degree of uncertainty, we vary the percentage of the demands that are in fact not deterministic. We consider three levels of uncertainty: low, medium and high with 45%, 60% and 72% of stochastic demands, respectively.

In terms of the degree of supply availability, we conduct all experiments on two levels of supply availability: standard and scarce. The standard supply are the typical amount as shown from our production traces. The scarce supply mimics inventory crunch situations, where we only have half of the standard supplier inventory available.

The results of our numerical experiments are shown in Table 5.3 and 5.4.

Instance		Improvement Over DBA		
Uncertainty	Supply	Expectation	CVaR	MeanDev
Low	Standard	72.74%	72.82%	61.54 %
Low	Scarce	18.33%	19.47%	14.09%
Medium	Standard	70.89%	68.65%	59.68%
Medium	Scarce	26.77%	27.63%	16.34%
High	Standard	64.86%	62.02%	53.81%
High	Scarce	18.97%	19.82%	12.76%

Table 5.3: Cost improvement of our algorithm over the two benchmark algorithms DBA.

Instance		Improvement Over SBA		
Uncertainty	Supply	Expectation	CVaR	MeanDev
Low	Standard	57.04%	58.75%	44.91%
Low	Scarce	9.28%	9.30%	5.96%
Medium	Standard	56.34%	54.54%	43.63%
Medium	Scarce	14.13%	15.36%	3.12%
High	Standard	50.40%	48.58%	37.52%
High	Scarce	11.72%	13.12%	4.46%

Table 5.4: Cost improvement of our algorithm over the benchmark algorithm SBA.

Results. We observe that our algorithm significantly outperforms both baselines for all settings and risk measures. In particular, we see the greatest gains in the instances with standard supply ranging from 53% to 73% for DBA and from 37% to 59% for SBA; as expected SBA performs better than DBA, but still far from our algorithm.

For the scarce supply experiments, we observe smaller, yet still noticeable gains (12% to 28% for DBA, 3% to 16% for SBA). This is attributed to the fact that a large number of demands are inevitably unfulfilled due to supply scarcity for any algorithm. These unmet demands incur a large penalty that dominates the smaller differences between the different algorithms, thus making the improvement seemingly smaller.

Robustness in the Number of Scenarios.

In practice, the empirical distribution may contain a large number of scenarios. To save on time and computing resources, the cloud service provider may sometimes prefer to consider only a subset of the scenarios for deployment optimization. We

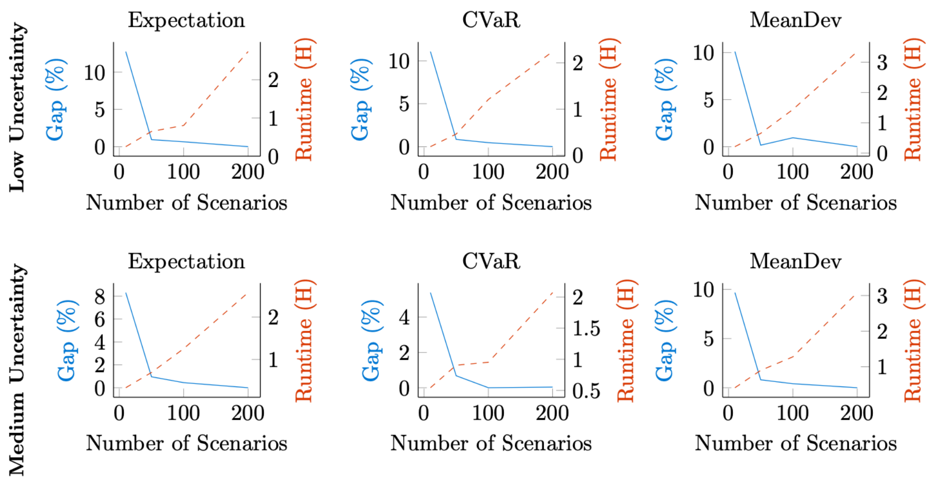


Figure 5-4: Trade-off of total cost and running time as the number of scenarios varies.

thus investigate how this behavior affects the quality of our deployment solution. In each run, we obtain a subset of the scenarios (of size 1, 5, 10, 30, 50, 100, or 200 scenarios) by sampling uniformly at random. We then run our algorithm using the subset of scenarios, and evaluate the resulting decisions on the full scenario set.

Results. We observe that for most risk measures, 50 scenarios are sufficient to obtain good quality solutions (with gap $< 5\%$ to the optimal solution). Reducing the number of scenarios leads to faster total running times, as expected. In particular, we can save 50% - 70% of the original running time by randomly selecting only 50 scenarios out of the 200 scenarios.

5.7 Conclusion

In this chapter, we introduce the server deployment problem which is critical for cloud supply chains. We propose a new class of algorithms based on stochastic optimization, and show their merits using real production traces. In particular, our algorithms clearly outperform other heuristic approaches (including the one used in production) with respect to the actual cost savings. Furthermore, by combining a variety of techniques, we are able to scale to large problem instances in adequate running times. There are several interesting directions that are left for future work. First, we are exploring additional techniques to further improve the running times of our algorithm, such as including only the most "meaningful" scenarios in the subgradient generation. On the modeling side, there are several intriguing directions, such as multiple demand sizes, and modern data center architectures that embed additional combinatorial constraints (Zhang et al. (2021)).

Appendix A

Appendix for Chapters 2 and 3

A.1 Regret Analysis for Elimination-Based Half-Q-Learning

Recall the coefficients $\alpha_t := \frac{H+1}{H+t}$ used in Algorithm *HQL*. We define related weights $\alpha_t^0 := \prod_{j=1}^{t}(1-\alpha_j)$, and $\alpha_t^i := \alpha_i \prod_{j=i+1}^{t}(1-\alpha_j)$ as in Jin et al. (2018) and in Dong et al. (2019). Below are useful properties of these weights:

Lemma 11 *The following properties hold:*

1. $\sum_{i=1}^{t} \alpha_t^i = 1$ *and* $\alpha_t^0 = 0, \forall t \geq 1$;
2. $\sum_{i=1}^{t} \alpha_t^i = 0$ *and* $\alpha_t^0 = 1$ *when* $t = 0$
3. $\max_{i\in[t]} \alpha_t^i \leq \frac{2H}{t}$ *and* $\sum_{i=1}^{t} \left(\alpha_t^i\right)^2 \leq \frac{2H}{t}, \forall t \geq 1$
4. $\sum_{t=i}^{\infty} \alpha_t^i = 1 + \frac{1}{H}$ *for every* $i \geq 1$
5. $\frac{1}{\sqrt{t}} \leq \sum_{i=1}^{t} \frac{\alpha_t^i}{\sqrt{i}} \leq \frac{1+\frac{1}{H}}{\sqrt{t}}$ *for every* $t \geq 1$

Remark: The last property is tighter than the corresponding bound in Lemma 4.1 in Jin et al. (2018). See Appendix A.1.1.

We state the fact that base-stock policies are optimal for the episodic lost-sales model with zero lead time in the following Lemma 12. Lemma 12 can be obtained by applying classical results in Porteus (2002). For completeness, we provide a proof in Appendix A.1.2.

Lemma 12 *Base-stock policies are optimal for the episodic lost-sales model with zero lead time.*

For any base-stock policy, the reward and the leftover inventory level only depend on the base-stock level and do not depend on the state, even though the feasible action set depends on the state. Therefore, in this setting, we can simplify the Q-value functions: $Q(x, y) = Q(y), \forall x \in \mathcal{S}$.

Recall for any $(x, h, k) \in \mathcal{S} \times [H] \times [K]$, and for any base-stock level $y \in A_h^k$, $\tau_h^k(x, y)$ is the next time step after time step h in episode k that our policy lands on a simulated inventory level ${x_{\tau_h^k(x,y)}^k}'$ that allows us to take an action in the running set $A_{\tau_h^k(x,y)}^k$. Therefore, $\tau_h^k(x, y)$ is a stopping time. The time steps in between are "skipped" in the sense that the Q, V-values for those time steps never appear on the right hand side of Equation (A.2) when we update value functions. If no skipping happened, then $\tau_h^k(x, y) = h + 1$, and we have the original Bellman equation (2.2.1). Using the general property of optional stopping that $\mathbf{E}[M_\tau] = M_0$ for any stopping time τ and discrete-time martingale M_τ, our Bellman optimality equation becomes the following *delayed form* of the Bellman equation:

$$Q_h^*(x, y) = Q_h^*(y) = \mathbf{E}_{\tau_h^k, \tilde{r}^*_{h,\tau_h^k}, x'_{\tau_h^k}}[\tilde{r}^*_{h,\tau_h^k} + V^*_{\tau_h^k}(x'_{\tau_h^k})] \tag{A.1}$$

where we simplify the notation $\tau_h^k(x,y)$ to τ_h^k, and recall $\tilde{r}_{h,h'}$ denotes the cumulative reward from step h to h'.

Using the stopping times and simulated trajectories, *HQL* updates the Q-values backward $h = H, \ldots, 1$ as follows:

$$Q_h^{k+1}(y) \leftarrow (1-\alpha_k) Q_h^k(y) + \alpha_k [\tilde{r}^{k+1}_{h,\tau_h^{k+1}} + V^{k+1}_{\tau_h^{k+1}}(x'_{\tau_h^{k+1}})]. \tag{A.2}$$

where Q_h^k, V_h^k denotes the Q_h, V_h functions at the beginning of episode k respectively.

Then by Equation (A.2) and the definition of the weights α_k^i's,

$$Q_h^k(y) = \alpha_{k-1}^0 H + \sum_{i=1}^{k-1} \alpha_{k-1}^i \left[\tilde{r}^i_{h,\tau_h^k} + V^{i+1}_{\tau_h^k}\left(x^i_{\tau_h^k}\right)\right]. \tag{A.3}$$

which naturally gives us Lemma 13, where we bound the difference between the optimal Q-value of a state-action pair and our estimated Q-value. The proof of Lemma 13 is provided in Appendix A.1.3.

Lemma 13 *For any* $(x,h,k) \in \mathcal{S} \times [H] \times [K]$, *and for any* $y \in A_h^k$, *we have*

$$\begin{aligned}\left(Q_h^k - Q_h^*\right)(y) =& \alpha_{k-1}^0 \left(H - Q_h^\star(y)\right) + \sum_{i=1}^{k-1} \alpha_{k-1}^i \Big[\left(V^{i+1}_{\tau_h^i} - V^*_{\tau_h^i}\right)(x^i_{\tau_h^i}) + \tilde{r}^i_{h,\tau_h^i} - \tilde{r}^*_{h,\tau_h^i} \\ &+ \Big(V^*_{\tau_h^i(x,y)}(x^i_{\tau_h^i}) + \tilde{r}^*_{h,\tau_h^i} - \mathbf{E}_{\tilde{r}^*,x',\tau_h^i}\left[\tilde{r}^*_{h,\tau_h^i} + V^*_{\tau_h^i}(x'_{\tau_h^i})\right]\Big)\Big].\end{aligned}$$

Then by identifying the martingales in the right-hand side of Lemma 13, we bound the difference between our Q-value estimates and the optimal Q-values in the following lemma:

Lemma 14 *For any* $(x,h,k) \in \mathcal{S} \times [H] \times [K]$, *and any* $y \in A_h^k$, *let* $\iota = 9\log(AT)$, *we have:*

$$\left|(Q_h^k - Q_h^*)(y)\right| \leq \alpha_{k-1}^0 H + \sum_{i=1}^{k-1} \alpha_{k-1}^i \left| \left(V_{\tau_h^i}^{i+1} - V_{\tau_h^i}^*\right)(x_{\tau_h^i}^i) + \tilde{r}_{h,\tau_h^i}^i - \tilde{r}_{h,\tau_h^i}^* \right| + c\sqrt{\frac{H^3 \iota}{k-1}} \tag{A.4}$$

with probability at least $1 - 1/(AT)^8$, *for some* $c \geq 2\sqrt{2}$.

The proof of Lemma 14 is provided in Appendix A.1.4.

We review *shortfall decomposition* below. For proof and reference, see Appendix A.1.7.

Lemma 15 *(shortfall decomposition) For any policy* π *and any episode* k*, the per-episode regret is:*

$$\left(V_1^* - V_1^{\pi_k}\right)(x_1^k) = \mathbf{E}_\pi \Big[\sum_{h=1}^{H} \big(\max_{y \in \mathcal{A}} Q_h^*(x_h^k, y) - Q_h^*(x_h^k, y_h^k) \big) \Big].$$

Shortfall decomposition allows us to calculate the regret of our policy by summing up the difference between the optimal Q-values of our action and those of the optimal action from the same state. We then find high-probability upper-bounds on the sum.

Proof: Proof for Theorem 1

Recall that we partition the time steps $h = 1, \ldots, H$ in each episode k into two sets, Γ_A^k and Γ_B^k, where Γ_A^k contains all the steps h where we are able to choose from the running set, and Γ_B^k contains all the steps h where we are unable to choose from the running set.

Then by shortfall decomposition, we have that the per-episode regret is

$$\begin{aligned}\big(V_1^* - V_1^{\pi_k}\big)(x_1^k) =& \mathbf{E}\Big[\sum_{h=1}^{H}\Big(\max_{y\ge x_h^k} Q_h^*(y) - Q_h^*(y_h^k)\Big)\Big]\\ \le& \mathbf{E}\Big[\sum_{h\in\Gamma_A^k}\max_{y\ge x_h^k}\Big(Q_h^*(y) - Q_h^*(y_h^k)\Big)\Big] + \mathbf{E}\Big[\sum_{h\in\Gamma_B^k}\max_{y\ge x_h^k}\Big(Q_h^*(y) - Q_h^*(y_h^k)\Big)\Big].\end{aligned}$$

Recall Lemma 1 and that we define $\{\delta_h\}_{h=1}^{H+1}$ to be a list of values that satisfy the following recursive relationship:

$$\begin{aligned}&\delta_h = H + (1+1/H)\delta_{h+1} + c\sqrt{H^3\iota}, \forall h\in[H],\\ &\delta_{H+1} = 0\end{aligned}$$

where c is the same constant as in Lemma 14.

By Lemma 1.3, we can bound the first term on the right-hand side:

$$\begin{aligned}&\mathbf{E}\Big[\sum_{h\in\Gamma_A^k}\max_{y\ge x_h^k}\Big(Q_h^*(y) - Q_h^*(y_h^k)\Big)\Big]\\ &\le \mathbf{E}\big[\sum_{h\in\Gamma_A^k}\frac{3\delta_h}{\sqrt{k-1}}\big]\mathbf{P}\Big(\max_{y\ge x_h^k}\big(Q_h^*(y) - Q_h^*(y_h^k)\big) \le \frac{3\delta_h}{\sqrt{k-1}}\Big)\\ &+ \sum_{h\in\Gamma_A^k} H\cdot\mathbf{P}\Big(\max_{y\ge x_h^k}\Big(Q_h^*(y) - Q_h^*(y_h^k)\Big) > \frac{3\delta_h}{\sqrt{k-1}}\Big)\\ &\le \mathcal{O}\Big(\sum_{h\in\Gamma_A^k}\frac{\delta_h}{\sqrt{k-1}}\Big) + \mathcal{O}\Big(\sum_{h\in\Gamma_A^k}\frac{H}{A^5T^5}\Big).\end{aligned}$$

By Lemma 1.4, we can bound the last term

$$\mathbf{E}\Big[\sum_{h\in\Gamma_B^k}\max_{y\ge x_h^k}\Big(Q_h^*(y) - Q_h^*(y_h^k)\Big)\Big] \le 0\cdot(1-\frac{1}{A^5T^5}) + \sum_{h\in\Gamma_B^k} H\cdot\frac{1}{A^5T^5} \le \sum_{h\in\Gamma_B^k} H\cdot\frac{1}{A^5T^5}.$$

Then the difference between the expected total reward of HQL and of the optimal policy π^* is

$$\begin{aligned}\text{Regret}_{MDP}(K) &= \left(V_1^* - V_1^{\pi_1}\right)(x_1^1) + \sum_{k=2}^{K} \left(V_1^* - V_1^{\pi_k}\right)(x_1^k) \\ &\leq H + \sum_{k=2}^{K} \Big(\sum_{h \in \Gamma_B^k}^{H} \frac{H}{A^5 T^5} + \sum_{h \in \Gamma_A^k} \frac{\delta_h}{\sqrt{k-1}} + \sum_{h \in \Gamma_A^k} \frac{H}{A^5 T^5} \Big) \\ &\leq \sum_{k=2}^{K} \frac{\mathcal{O}(\sqrt{H^7 \iota})}{\sqrt{k-1}} \leq \mathcal{O}(H^3 \sqrt{T \iota}).\end{aligned}$$

It follows that the total expected regret of HQL against OPT is

$$\begin{aligned}\text{Regret}_{total}(K) = \text{Regret}_{MDP}(K) + \text{Regret}_{gap}(K) &= \mathcal{O}\Big(H^3 \sqrt{T\iota} + (M-m)/K\Big) \\ &= \mathcal{O}\Big(H^3 \sqrt{T \log T}\Big)\end{aligned}$$

Finally, recall the scaling we performed on the reward, so we multiply by the factor $\mathcal{O}\big(M \cdot \max(|o_h|, |p_h|)\big)$. This implies an $\mathcal{O}\big(H^3 M \cdot \max(|o_h|, |p_h|)\sqrt{T \log T}\big)$ total dependence on all setting parameters.

$\square$

A.1.1 Properties of weights α_t^i

We obtain the last property in Lemma 11 by a more careful algebraic analysis, so that we obtain a tighter bound on $\sum_{i=1}^{t} \frac{\alpha_t^i}{\sqrt{i}}$ than the corresponding bound in Jin et al. (2018). For the remaining properties in Lemma 11, see Lemma 4.1 in Jin et al. (2018).

Proof: Proof of Lemma 11, part 5: We prove the last property in Lemma 11 by induction. For the base case $t = 1$, we have $\sum_{i=1}^{t} \frac{\alpha_t^i}{\sqrt{i}} = \alpha_1^1 = 1$ so the statement holds. For $t \geq 2$, by the relationship $\alpha_t^i = (1-\alpha_t)\alpha_{t-1}^i$ for $i = 1, \ldots, t-1$ we have

$$\sum_{i=1}^{t} \frac{\alpha_t^i}{\sqrt{i}} = \frac{\alpha_t}{\sqrt{t}} + (1-\alpha_t)\sum_{i=1}^{t-1} \frac{\alpha_{t-1}^i}{\sqrt{i}} \tag{A.5}$$

Assuming the inductive hypothesis holds, on the one hand,

$$\frac{\alpha_t}{\sqrt{t}} + (1-\alpha_t)\sum_{i=1}^{t-1} \frac{\alpha_{t-1}^i}{\sqrt{i}} \geq \frac{\alpha_t}{\sqrt{t}} + \frac{1-\alpha_t}{\sqrt{t-1}} \geq \frac{\alpha_t}{\sqrt{t}} + \frac{1-\alpha_t}{\sqrt{t}} = \frac{1}{\sqrt{t}}$$

where the first inequality holds by the inductive hypothesis. On the other hand,

$$\begin{aligned} \frac{\alpha_t}{\sqrt{t}} + (1-\alpha_t)\sum_{i=1}^{t-1} \frac{\alpha_{t-1}^i}{\sqrt{i}} &\leq \frac{\alpha_t}{\sqrt{t}} + \frac{(1+1/H)(1-\alpha_t)}{\sqrt{t-1}} = \frac{H+1}{\sqrt{t}(H+t)} + \frac{(1+1/H)\sqrt{t-1}}{H+t} \\ &\leq \frac{H+1}{\sqrt{t}(H+t)} + \frac{(1+1/H)\sqrt{t}}{H+t} \leq \frac{(1+1/H)}{\sqrt{t}} \end{aligned} \tag{A.6}$$

where the first inequality holds by the inductive hypothesis. □

A.1.2 Optimality of base-stock policies

Proof: Proof of Lemma 12: Since the optimal value functions $V_h^*(\cdot)$ and $Q_h^*(\cdot)$ evaluate all possible ways of ordering inventory at each time step throughout each episode, the fact that they turn out to be concave (Lemma 16) implies that there is one single quantity that we should order up to for each time period h to obtain the maximum expected reward. □

A.1.3 Proof of Lemma 13

Proof: Proof of Lemma 13: From the Bellman optimality equation (A.1), and the fact that $\sum_{i=0}^{k-1} \alpha_{k-1}^i = 1$, we have

$$Q_h^*(y) = \alpha_{k-1}^0 Q_h^*(y) + \sum_{i=1}^{k-1} \alpha_{k-1}^i \left[\mathbb{E}_{x', \tau_h^i} [\tilde{r}_{\tau_h^i}^* + V_{\tau_h^i}^*(x'_{\tau_h^i})] \right]$$

Subtracting Equation (A.3) from this equation, and adding some of the middle terms that cancel with themselves gives us Lemma 13. $\square$

A.1.4 Proof of Lemma 14

Proof: Proof of Lemma 14: Since we assume that given a fixed value D_h, the next state $x_{h+1}(y_h)$ is increasing in y_h, and $a_h(x_h)$ is increasing in x_h for the lower one-sided-feedback problem, we conclude that the (deterministic given D_h) dynamics are monotone with respect to any simulation starting point x_h. Since the algorithm chooses at least the maximal action in A_h^k at all times, this implies it can observe the simulated trajectory started from any $x_h \in A_h^k$ for any $k, h \in [K] \times [H]$.

Let $\mathcal{F}_h^i$ be the σ-field generated by all the random variables until episode i, stage h. Then for any $\tau \in [K]$,

$$\left(V_{\tau_h^i}^*(x_{\tau_h^i}^i) + \tilde{r}_{\tau_h^i}^* - \mathbb{E}_{\tilde{r}^*, x', \tau_h^i} \left[\tilde{r}_{\tau_h^i}^* + V_{\tau_h^i}^*(x'_{\tau_h^i}) \right] \right)_{i=1}^{\tau}$$

is a martingale difference sequence with respect to the filtration $\{\mathcal{F}_h^i\}_{i \geq 0}$. Then by

Azuma-Hoeffding Theorem, we have that with probability at least $1-(1/AT)^9$:

$$\begin{aligned}
&\left|\sum_{i=1}^{k-1}\alpha_{k-1}^i\cdot\left(V^*_{\tau_h^i}(x^i_{\tau_h^i})+\tilde{r}^*_{\tau_h^i}-\mathbb{E}_{\tilde{r}^*,x',\tau_h^i}\left[\tilde{r}^*_{\tau_h^i}+V^*_{\tau_h^i}(x'_{\tau_h^i})\right]\right)\right| \\
&\leq \frac{cH}{2}\sqrt{\sum_{i=1}^{k-1}\left(\alpha_{k-1}^i\right)^2\cdot\iota}\leq c\sqrt{\frac{H^3\iota}{k-1}}
\end{aligned} \tag{A.7}$$

for any constant $c\geq 2\sqrt{2}$. By union bound, we have with probability at least $1-(1/AT)^8$ that for any $x,h,k,y\in A_h^k$,

$$\left|\sum_{i=1}^{k-1}\alpha_{k-1}^i\left(V^*_{\tau_h^i}(x^i_{\tau_h^i})+\tilde{r}^*_{\tau_h^i}-\mathbb{E}_{\tilde{r}^*,x',\tau_h^i}\left[\tilde{r}^*_{\tau_h^i}+V^*_{\tau_h^i}(x'_{\tau_h^i})\right]\right)\right|\leq c\sqrt{\frac{H^3\iota}{k-1}}$$

By this equation and Lemma 13, Lemma 14 follows. □

A.1.5 Upper bound on sequence $\delta_h, h=1,\ldots,H$

Proof: We set $d_h=(\delta_h)\cdot\left(1+\frac{1}{H}\right)^h$ and observe that the recurrence implies

$$d_h=d_{h+1}+H+2\sqrt{2}\sqrt{H^3\iota} \tag{A.8}$$

Then from this recursion we see $d_h\leq H^2+2\sqrt{2H^5\iota}$ for all h. Since d_h,δ_h differ by a constant factor $\left(1+\frac{1}{H}\right)^h$, we have $\delta_h=\frac{H^2+2\sqrt{2H^5\iota}}{\left(1+\frac{1}{H}\right)^h}\leq 4\sqrt{H^5\iota}$. □

A.1.6 Concavity of the Optimal Value Functions

Below we prove the concavity of the Q,V value functions of the lost-sales model with zero lead time. The same proof and result applies to the single-product backlogged model.

Lemma 16 *For the lost-sales model, for any $h \in [H]$, the optimal V-value function $V_h^*(x)$ is concave in x, and the optimal Q-value function $Q_h^*(y)$ is concave in y.*

Proof: We proceed by backward induction on h, starting from the base case $h = H$. The base case is the value functions for the last step of each episode: $Q_H^*(y)$ and $V_H^*(x)$. Since $Q_H^*(y)$ is just the expectation of a one time reward for the last period, we know $Q_H^*(y) = -[o_H(y - D_H)^+ + p_H \min(y, D_H)]$. This function is concave in y. Since $V_H^*(x) = \max_{y \geq x} Q_H^*(y)$, the graph of $V_H^*(x)$ is constant on the left of $x = \arg\max_{y\geq x} Q_H^*(y)$, and then goes down with a slope of $-o_H$ on the right of $x = \arg\max_{y \geq x} Q_H^*(y)$. So $V_H^*(x)$ is also concave.

Now suppose $Q_{h+1}^*(y)$ and $V_{h+1}^*(x)$ are concave. It remains to show concavity of $Q_h^*(y)$ and $V_h^*(x)$.

Since $Q_h^*(y) = \mathbb{E}[V_{h+1}^*(y - D_h) + r_h(y, D_h)]$, and we know $r_h(y, D_h)$ is concave in y just like $Q_H^*(y)$, and that $V_{h+1}^*(x)$ is concave in x from the induction hypothesis, which means $V_{h+1}^*(y - D_h)$ is concave in y for any value of D_h. Therefore, $\mathbb{E}[V_{h+1}^*(y - D_h) + r_h]$ is also concave, as a weighted average of concave functions. Thus, $Q_h^*(y)$ is concave, and $V_h^*(x) = \max_{y \geq x} Q_h^*(y)$ is concave. □

A.1.7 Shortfall decomposition

The following proof of shortfall decomposition is adapted from Benjamin Van Roy's reinforcement learning notes for the class MS 338 at Stanford University.

Proof: Proof of Lemma 15: For any policy π, let y_h^k denote the action the policy π_k takes at stage h of episode k. Let R_h denote the expected reward of y_h^k.

Define

$$Z_{h+1} = \begin{cases} R_h + \max_y Q^*_{h+1}\left(x^k_{h+1}, y\right) & \text{if } h < H \\ R_h & \text{if } h = H \end{cases}$$

Then

$$\mathbb{E}_\pi\left[Q^*_h\left(x^k_h, y^k_h\right)\right] = \mathbb{E}_\pi\left[Z_{h+1}\right]$$

Therefore,

$$\begin{aligned} V^*_1 - V^{\pi_k}_1 =& \mathbb{E}_\pi\Big[\max_{a\in\mathcal{A}} Q^*_1(x^k_1, a) - \sum_{h=1}^{H} R_h\Big] \\ =& \mathbb{E}_\pi\Big[\max_{a\in\mathcal{A}} Q^*_1\left(x^k_1, a\right) - \sum_{h=1}^{H}\left(R_h - Z_{h+1} + Q^*_h\left(x^k_h, y^k_h\right)\right)\Big] \\ =& \mathbb{E}_\pi\Big[\sum_{h=1}^{H}\big(\max_{a\in\mathcal{A}} Q^*_h(x^k_h, a) - Q^*_h(x_h, y^k_h)\big)\Big] \end{aligned}$$

□

A.1.8 Proof for Lemma 1

Proof: We prove by backward induction. Note that all of our statements below hold with high probability. In particular, we will use Azuma-Hoeffding no more than AT times in the below, with each use holding with probability at least $1/(AT)^9$. Under the assumption that each use of Azuma-Hoeffding holds we will obtain the statements of the Lemma. Our proof goes by induction; for the base case $\delta_{H+1} = 0$ satisfies the Inequality in Lemma 1.1 (actually equality here) with probability 1 based on Bellman equations.

Now suppose the inequality in Lemma 1.1 is true for any $k \in [K]$, $x \in \mathcal{S}$, then for any $h' = \tau_h^k(x,a)$ that has $a \in A_h^k$:

$$\max_{y \in A^k_{\tau_h^k(x,a)}} \left|(Q^k_{\tau_h^k(x,a)} - Q^*_{\tau_h^k(x,a)})(y)\right| \le \frac{\delta_{\tau_h^k(x,a)}}{\sqrt{k-1}} \tag{A.9}$$

for all $a \in A_h^k$ with high probability. Then the statement of Lemma 1.2 is true for $k, \tau_h^k(x,a)$: recall for any (x,h,k), $y_h^{k*} = \arg\max_{y \in A_h^k} Q_h^k(y)$. Suppose $y_h^* \notin A_h^k$, then $Q_h^k(y_h^*) < Q_h^k(y_h^{k*}) - \frac{8\sqrt{H^5\iota}}{\sqrt{k-1}} = Q_h^k(x, y_h^{k*}) - \frac{2\delta_h}{\sqrt{k-1}}$. Then we need either $Q_h^k(y_h^*) < Q_h^*(y_h^*) - \frac{\delta_h}{\sqrt{k-1}}$ or $Q_h^k(y_h^{k*}) > Q_h^*(y_h^{k*}) + \frac{\delta_h}{\sqrt{k-1}}$. Therefore by Equation (A.9), Prob($y_h^* \notin A_h^k(x)$)$\le \frac{1}{(AT)^5}$. Therefore, the optimal action $y^*_{\tau_h^k(x,a)}$ is in the running set $A^k_{\tau_h^k(x,a)}$ with high probability.

Lemma 1.3 is also true: by Equation A.9, the optimal Q-value of the optimal policy's action $Q^*_{\tau_h^k(x,a)}(y^*_{\tau_h^k(x,a)})$ is with high probability at most $\frac{\delta_h}{\sqrt{k-1}}$ more than the estimated Q-value of our estimated best arm $Q^k_{\tau_h^k(x,a)}(y^{k*}_{\tau_h^k(x,a)})$. Any action we take in $A^k_{\tau_h^k(x,a)}$ has an estimated Q-value no more than $\frac{8\sqrt{H^5\iota}}{\sqrt{k-1}} = \frac{2\delta_{\tau_h^k(x,a)}}{\sqrt{k-1}}$ lower than $Q^k_{\tau_h^k(x,a)}(y^{k*}_{\tau_h^k(x,a)})$ base on our algorithm. Therefore, the optimal Q-value of the optimal policy's action $Q^*_{\tau_h^k(x,a)}(y^*_{\tau_h^k(x,a)})$ is with high probability at most $\frac{3\delta_{\tau_h^k(x,a)}}{\sqrt{k-1}}$ more than the estimated Q-value of any action $y \in A^k_{\tau_h^k(x,a)}(x)$. Then again, by Equation A.9, we know that the optimal Q-value of the optimal policy's action $Q^*_{\tau_h^k(x,a)}(y^*_{\tau_h^k(x,a)})$ is with high probability at most $\frac{4\delta_{\tau_h^k(x,a)}}{\sqrt{k-1}}$ more than the optimal Q-value of any action in $A^k_{\tau_h^k(x,a)}$.

Then the statement of Lemma 1.4 is true: from Lemma 1.2, we know that with high probability, the optimal action is in the running set. When the running set is not feasible to choose from, then recall the assumptions that the value functions are concave and that the feasible action set at any time is an interval of the form $\mathcal{A} \cap [a, \infty)$ for some a dependent on the state. So if we cannot play in the running set, then the running set, and hence w.h.p. the true optimal action, is contained in $(-\infty, a)$. By concavity, this implies that the closest feasible action to the running set is optimal in this case with high probability.

Now we induct on the previous stage $h' = h$. By Lemma 14, with probability at least $1 - 1/(AT)^8$

$$\max_{y \in A_h^k} \left|(Q_h^k - Q_h^*)(y)\right| \leq \max_{a \in A_h^k} \Bigg\{ \alpha_{k-1}^0 H + \sum_{i=1}^{k-1} \alpha_{k-1}^i \Bigg[\left(V_{\tau_h^i(x,a)}^{i+1} - V_{\tau_h^i(x,a)}^*\right) \left({x_{\tau_h^i(x,a)}^i}'\right) + \tilde{r}_{h,\tau_h^i(x,a)}^i - \tilde{r}_{h,\tau_h^i(x,a)}^* \Bigg] + c\sqrt{\frac{H^3 \iota}{k-1}} \Bigg\}.$$

Based on our inductive hypothesis, we have

$$\begin{aligned} \max_{a \in A_h^k} \Big[\left(V_{\tau_h^i(x,a)}^{i+1} - V_{\tau_h^i(x,a)}^*\right) \left({x_{\tau_h^i(x,a)}^i}'\right) + \tilde{r}_{h,\tau_h^i(x,a)}^i - \tilde{r}_{h,\tau_h^i(x,a)}^* \Big] \\ \leq \max_{y \in A_{\tau_h^i(x,a)}^i} |(Q_{\tau_h^i(x,a)}^{i+1} - Q_{\tau_h^i(x,a)}^*)(y)| \leq \frac{\delta_{\tau_h^i(x,a)}}{\sqrt{i}} \end{aligned}$$

where the first inequality is because $\tilde{r}_{h,\tau_h^i(x,a)}^i - \tilde{r}_{h,\tau_h^i(x,a)}^*$ is with high probability zero because of the Lemma 1.4 part of the inductive hypothesis. Then

$$\max_{y \in A_h^k} \left|(Q_h^k - Q_h^*)(y)\right| \leq \max_{a \in A_h^k} \left\{ \alpha_{k-1}^0 H + \left(\sum_{i=1}^{k-1} \alpha_{k-1}^i \cdot \frac{\delta_{\tau_h^i(x,a)}}{\sqrt{i}}\right) + c\sqrt{\frac{H^3 \iota}{k-1}} \right\}. \tag{A.10}$$

We can bound α_{k-1}^0 by $\frac{1}{\sqrt{k}}$, and bound $\sum_{i=1}^{k-1} \alpha_{k-1}^i \cdot \frac{\delta_{\tau_h^i(x,a)}}{\sqrt{i}}$ by $\frac{1+1/H}{\sqrt{k-1}} \delta_{\tau_h^i(x,a)}$ using

Lemma 11:

$$\begin{aligned}\max_{y\in A_h^k}\left|(Q_h^k - Q_h^*)(y)\right| &\le \frac{1}{\sqrt{k}}H + \frac{1+1/H}{\sqrt{k-1}}\delta_{\tau_h^i(x,a)} + c\sqrt{\frac{H^3\iota}{k}} \\ &\le \frac{1}{\sqrt{k-1}}H + \frac{1+1/H}{\sqrt{k-1}}\delta_{h+1} + c\sqrt{\frac{H^3\iota}{k-1}} = \frac{\delta_h}{\sqrt{k-1}}\end{aligned} \tag{A.11}$$

where the second inequality is because $\tau_h^i(x,a) \ge h+1$ and δ_h's is a decreasing sequence. The last equality is true based on the recursive definition of δ_h. □

A.1.9 Regret caused by discretization.

Proof: Proof of Lemma 2: If we discretize $[m, M]$ with step-size $\frac{M-m}{T^2}$, for example, then $A = \Theta(T^2)$. Discretization incurs additional regret: $\text{Regret}_{gap} = \mathcal{O}(\frac{M-m}{T^2} \cdot HT) = o(1)$ by Lipschitzness of the reward function. □

A.2 Applying existing Q-learning algorithms on the inventory control problems

Here we show that existing Q-learning results in general MDPs give suboptimal guarantees when specialized to our setting, as discussed in Section 2.3 of the main text.

For Jin et al. (2018), suppose we discretize the state and action space optimally with step-size ϵ_1 to apply Jin et al. (2018) to the backlogged/lost-sales episodic inventory control problem with continuous action and state space. Then the Regret_{gap} we get is $\epsilon_1 T$. Applying the results of Jin et al. (2018), their Regret_{MDP} is $\mathcal{O}(\sqrt{H^3SAT\iota}) = \mathcal{O}(\sqrt{\frac{1}{\epsilon_1}\cdot\frac{1}{\epsilon_1}T\iota})$. To minimize Regret_{total}, we balance the Regret_{MDP} and Regret_{gap} by setting $\sqrt{\frac{1}{\epsilon_1} \cdot \frac{1}{\epsilon_1} T} = \epsilon_1 T$, which gives $\epsilon_1 = \frac{1}{T^{1/4}}$, giving us an optimized regret bound of $\mathcal{O}(T^{\frac{3}{4}} \sqrt{H^3 \log T})$.

For Dong et al. (2019), suppose we discretize the state and action space optimally with step-size ϵ_2 to apply Dong et al. (2019) to the backlogged/lost-sales episodic inventory control problem. We also optimize aggregation using the special property of these inventory control problems that the Q-values only depend on the action not the state, so we aggregate all the state-action pairs $(x_1, y), (x_2, y)$ into one aggregated state-action pair. This 0-error aggregation helps reduce the aggregated state-action space. Then the optimized regret bound in Dong et al. (2019) is $\mathcal{O}(\sqrt{H^4 \frac{1}{\epsilon} T \log T} + \epsilon T)$. We minimize Regret_{total} by balancing the two terms and take $\epsilon = \frac{1}{T^{1/3}}$, obtaining an optimized regret bound of $\mathcal{O}(T^{\frac{2}{3}} \sqrt{H^4 \log T})$.

A.3 The Non-Discarding Lost-Sales Model

In the infinite-horizon version of the non-discarding lost-sales model with cyclic demands: to have finite V-values, a long-time average reward $\overline{r}$ is subtracted from the right-hand side of the Bellman equations:

$$V_t = \mathbb{E}[V_{t+1} + r_t] - \overline{r}.$$

(Puterman, 2014, Theorem 8.4.7) guarantees the existence of an optimal average-reward policy. By taking limits of finite-horizon optimal policies, it can be proved that cyclic base-stock policies are optimal for the infinite-horizon problem.

Proof: Proof of Proposition 1: For inventory problems with known cyclic stochastic demands, (Zipkin, 1989, Proposition 1c) shows the existence of a time $h \in [H]$ such that for the episodic problem with demand distributions $D_{h+1}, \ldots, D_H, D_1, \ldots, D_h$, the optimal base-stock level is maximized at the first round with demand D_{h+1} (referred to therein as the "maximal property"). For this choice of h, it readily follows that the base-stock levels for the episodic problem are equal to those of any repeated version of length $T = KH$ again started from D_{h+1}. Indeed, because the base-stock level B_{h+1} for the episodic problem is maximal, using the episodic base-stock policy repeatedly on the T-horizon problem is equivalent to solving K separate episodic problems - we are always able to order back up to exactly B_{h+1}. As a result, this algorithm solves each episode optimally while achieving a best-case initialization for each episode. This implies that it solves the T-horizon problem optimally for any $T = KH$. (However note that this T-horizon problem is shifted from the original.)

□

We prove in the following proposition that the optimal policy for the infinite-horizon problem is also near optimal for the finite-horizon problem.

Proposition 3 *For any h and sequence $(D_1, \ldots, D_h)$, the infinite-horizon optimal policy, denoted by π^*_∞, when applied to the finite-horizon problem, achieves expected regret $O(M\gamma)$ independent of the time horizon length T from any starting state x and time $h \in [H]$, with respect to the optimal finite-horizon policy, denoted by π^*_T.*

Proof: Suppose not, which means that the infinite-horizon optimal policy has some amount of regret C' that is larger than $O(M\gamma)$ after time T. We will construct a candidate infinite-horizon policy π' with superior performance to the optimal policy π^*_∞, which would be a contradiction to the definition.

We construct this candidate policy by the following 3 phases

1. Run the optimal T-horizon policy until time T.

2. Order nothing until all inventory is depleted.

3. Copy the infinite-horizon policy from the best possible starting point for the rest of time.

Since all states are reachable from a 0 inventory state, after all inventory is depleted by phase 2, all states are reachable in phase 3. Hence the above policy is feasible.

By assumption, phase 1 above achieves reward C' greater than the optimal policy on average. Meanwhile, phase 2 requires time $O(M\gamma)$ in expectation. Therefore, the candidate policy above eventually matches the trajectory of the infinite horizon policy, but its reward is larger by a positive constant $C - O(M\gamma) > 0$. Moreover, it has the same starting point. This is a contradiction because π^*_∞ is by definition the optimal policy for the infinite horizon problem. $\square$

Proof: Proof of Lemma 3: To handle switches between arms, we simply wait for inventory to go below the base-stock level we want to choose for the beginning step of the next remaining arm h', and then start pulling arm h' once possible. By Assumption 3 for the non-episodic model, we know that each switch from an arm h to an arm h' will take $\mathcal{O}(M\gamma)$ time periods in expectation. By Markov Inequality, we know that the probability that the switch takes more than $\mathcal{O}(M\gamma)$ time periods is less than 1/2. Then the probability that the switch takes more than $\mathcal{O}(M\gamma \cdot 3\log T)$ time periods is less than $(\frac{1}{2})^{3\log T} = \frac{1}{T^3}$.

Each phase j contains $\mathcal{O}(2^j H)$ time periods, so there are no more than $\log T$ phases. Since $|W_j| \leq H$ for any j, there are only $\mathcal{O}(H \log T)$ arm switches in the whole horizon. Each switch takes time $\mathcal{O}(\gamma \log T)$ time periods with probability $1 - T^{-3}$, so switching between arms contribute negligible $\mathcal{O}(H\gamma \log^2 T)$ regret in the whole horizon. □

To analyze the regret bound for *Meta-HQL*, we need a tighter analysis than what is used in shortfall decomposition in *HQL*. Recall V_1^* denotes the expected optimal per-episode reward for the optimal policy. We use $V_1^{(w)}$ to denote the realized per-episode reward of the *w-shifted HQL* that arm w represents.

Proof: Proof of Lemma 4: First we want to show that for each arm, our estimated per-episode reward is very close to the true optimal per-episode reward for that arm. Let R_h denote the realized reward of y_h^k. For each episode $k \in [K]$ and time step $h \in [H]$, define

$$Z_{h+1}^k = \begin{cases} R_h^k + \max_y Q_{h+1}^*\left(x_{h+1}^k, y\right) & \text{if } h < H \\ R_h^k & \text{if } h = H \end{cases}$$

Then we have that for episode k, the difference between the optimal expected per-episode reward for the best arm and our realized per-episode reward for arm w is

$$\begin{aligned} V_1^* - V_1^{(w)} &= \max_a Q_1^*(x_1, a) - \sum_{h=1}^H R_h \\ &= \max_a Q_1^*(x_1, a) - \sum_{h=1}^H \Big(R_h - Z_{h+1}^k + Q_h^*(x_h, y_h)\Big) \\ &+ \sum_{h=1}^H \Big(Q_h^*(x_h, y_h) - Z_{h+1}\Big) \end{aligned} \tag{A.12}$$

Consider the last term in Equation (A.12) for rounds $k = 1, \ldots, K_j$. Each of these terms is bounded between $[-H, H]$ and has mean 0 conditioned on the past. Therefore, the partial sums over $k = 1, \ldots, K_j$ are martingales, with the difference between consecutive martingales bounded by $[-H, H]$. For notation, we use superscript k to denote round k. By Azuma-Hoeffding Inequality,

$$\mathbb{P}\Big[\left|\sum_{k=1}^{K_j}\sum_{h=1}^{H}\Big(Q_h^*(x_h^k, y_h^k) - Z_{h+1}^k\Big)\right| \geq \epsilon\Big] \leq 2\exp\big(-\frac{2\epsilon^2}{\sum_1^{K_j H} H^2}\big) \tag{A.13}$$

Then we have that the difference between the total expected reward and our realized reward for any arm if that arm is pulled for K_j rounds (meaning K_j cycles) is

$$\begin{aligned}\Big|\sum_{k=1}^{K_j}(V_1^* - V_1^{\pi_k})\Big| \leq & \Big|\sum_{k=1}^{K_j}\max_a Q_1^*(x_1^k, a) - \sum_{k=1}^{K_j}\sum_{h=1}^{H}\Big(R_h - Z_{h+1}^k + Q_h^*(x_h^k, y_h^k)\Big)\Big| \\ & + \Big|\sum_{k=1}^{K_j}\sum_{h=1}^{H}\Big(Q_h^*(x_h^k, y_h^k) - Z_{h+1}^k\Big)\Big| \\ \leq & \Big|\sum_{k=1}^{K_j}\max_a Q_1^*(x_1^k, a) - \sum_{k=1}^{K_j}\sum_{h=1}^{H}\Big(R_h - Z_{h+1}^k + Q_h^*(x_h^k, y_h^k)\Big)\Big| + \epsilon \\ \leq & \sum_{k=1}^{K_j}\sum_{h=1}^{H}\Big|\max_{a\in\mathcal{A}} Q_h^*(x_h^k, a) - Q_h^*(x_h^k, y_h^k)\Big| + \epsilon\end{aligned} \tag{A.14}$$

with probability at least $1 - 2\exp\big(-\frac{2\epsilon^2}{K_j H^3}\big)$. We take $\epsilon = 10\sqrt{H^3 K_j \log T}$; then the probability is at least $1 - 2\exp\big(-\frac{200 H^3 K_j \log T}{K_j H^3}\big) = 1 - 2e^{(-200\log T)} = 1 - \frac{2}{T^{200}}$.

On the other hand, let a^* denote the action that achieves $\max_{a\in\mathcal{A}} Q_h^*(x_h^k, a)$, then the first term inside the sum in the right hand side of the Inequality (A.12) is

$$\begin{aligned}
Q_h^*(x_h^k, a^*) - Q_h^*(x_h^k, y_h^k) \leq & Q_h^*(x_h^k, a^*) - Q_h^k(x_h^k, a^*) + Q_h^k(x_h^k, a^*) - Q_h^k(x_h^k, y_h^k) \\
& + Q_h^k(x_h^k, y_h^k) - Q_h^*(x_h^k, y_h^k) \\
\leq & \Big| Q_h^*(x_h^k, a^*) - Q_h^k(x_h^k, a^*) \Big| + \Big(Q_h^k(x_h^k, a^*) - Q_h^k(x_h^k, y_h^k) \Big) \\
& + \Big| Q_h^k(x_h^k, y_h^k) - Q_h^*(x_h^k, y_h^k) \Big| \\
\leq & \Big| Q_h^*(x_h^k, a^*) - Q_h^k(x_h^k, a^*) \Big| + \mathrm{CB}_1 \\
& + \Big| Q_h^k(x_h^k, y_h^k) - Q_h^*(x_h^k, y_h^k) \Big|
\end{aligned} \tag{A.15}$$

where the last inequality is due to the fact that the second term on the right-hand side is upper-bounded by the confidence interval CB_1 by definition of the running set in Algorithm 1. Recall that $\mathrm{CB}_1 \leq \mathcal{O}\Big(\sqrt{H^5 \iota} / \sqrt{K_j - 1}\Big)$.

On the other hand, by definition of a^*, the left-hand side is non-negative. Therefore, the right-hand side of Equation (A.15) is also an upper bound on the absolute value of the left-hand side. Therefore, we get that the first term on the right-hand side of Inequality (A.14) is upper-bounded by:

$$\begin{aligned}
\sum_{k=1}^{K_j} \sum_{h=1}^{H} \Big| Q_h^*(x_h^k, a^*) - Q_h^*(x_h^k, y_h^k) \Big| \leq & \sum_{k=1}^{K_j} \sum_{h=1}^{H} \Big| Q_h^*(x_h^k, a^*) - Q_h^k(x_h^k, a^*) \Big| \\
& + \sum_{k=1}^{K_j} \sum_{h=1}^{H} \Big| Q_h^k(x_h^k, y_h^k) - Q_h^*(x_h^k, y_h^k) \Big| + \mathrm{CB}_1
\end{aligned}$$

where the first term and the third term are both upper-bounded by

$$\sum_{k=1}^{K_j} \sum_{h=1}^{H} \max_a \Big| Q^*(x_h^k, a) - Q_h^k(x_h^k, a) \Big|.$$

Let w^* denote the best arm, that is, the arm that correctly chooses (one of) the time steps with the highest optimal base-stock level as the beginning of the cycles. By definition, the best arm has the highest optimal value function for the beginning of its cycles $V_1^{*(w^*)} \equiv \max_w V_1^{*(w)}$, which corresponds to having the highest expected per-episode reward among the arms.

When $w = w^*$, by part 1 of Lemma 1, we know that $\sum_{k=1}^{K_j} \sum_{h=1}^{H} \max_a \Big| Q^*(x_h^k, a) - Q_h^k(x_h^k, a) \Big|$ is bounded by $\frac{HK_j\delta_h}{\sqrt{K_j - 1}}$ with probability at least $1 - \frac{1}{A^5T^5}$. Therefore, let $E_{K_j}^{w^*}$ be the (random) total reward for arm w^* after pulling it for K_j cycles, then using the fact that $\delta_h \leq 4\sqrt{H^5\iota}$ again, the difference between the expected optimal reward for any arm w and our estimated reward after K_j samples of the arm w is

$$\big|K_jV_1^{*(w^*)} - E_{K_j}^{w^*}\big| \leq \mathcal{O}\big(\sqrt{H^7K_j \log T}\big) \tag{A.16}$$

with probability at least $1 - \frac{1}{T^5}$.

Let w_2 denote any suboptimal arm that has not been eliminated before being pulled K_j times. When $w = w_2$, then because of trimming, our estimated reward after K_j could be further lowered:

$$K_jV_1^{*(w_2)} \geq E_{K_j}^{w_2} - \mathcal{O}\big(\sqrt{H^7K_j \log T}\big) \tag{A.17}$$

with probability at least $1 - \frac{1}{T^5}$.

By definition, its optimal value is $V_1^{*(w_2)} \leq V_1^{*(w^*)}$. Let $E_{K_j}^{w_2}$ be the total reward for arm w_2 after pulling it for K_j cycles. Then by Equations (A.16) and (A.17), after

K_j samples, with probability $1-\frac{1}{T^5}$,

$$\begin{aligned} & E_{K_j}^{w^*} + C_2\sqrt{H^7 K_j \iota} \geq K_j V_1^{*(w^*)} \geq K_j V_1^{*(w_2)} \geq E_{K_j}^{w_2} - C_2\sqrt{H^7 K_j \iota} \\ \implies & E_{K_j}^{w^*} \geq E_{K_j}^{w_2} - 2C_2\sqrt{H^7 K_j \iota} \end{aligned} \tag{A.18}$$

for the same C_2 we used in the confidence bound CB in Algorithm 2.

Since this holds for all suboptimal arms w_2 and all no more than $\log T$ different values of K_j, by union bound, the probability of *Meta-HQL* never eliminating the best arm is at least $1-\frac{H\log T}{T^5} \leq \frac{1}{T^4}$.

□

Proof: Proof of Theorem 3: Suppose arm w_2 was eliminated after $K_j = K_{j(w_2)}$ samples of arm w_2. Then, arm w_2 was not eliminated when it was pulled $\frac{K_j}{2}$ times. To analyze the regret accumulated from pulling arms, observe that in fact each arm has total regret $\mathcal{O}\Big(\sqrt{\frac{H^7 K_j \iota}{2}}\Big)$ with probability at least $1-T^{-5}$ not only on its first $\frac{K_j}{2}$ samples, but also on its $\frac{K_j}{2}+1$ through K_j-th sample, as detailed below.

From the proof of Lemma 4, we know with probability at least $1-\frac{1}{T^4}$,

$$\begin{aligned} & E_{\frac{K_j}{2}}^{w_2} \geq E_{\frac{K_j}{2}}^{w^*} - 2C_2\sqrt{\frac{H^7 K_j \iota}{2}} \geq \frac{K_j V_1^{*(w^*)}}{2} - \mathcal{O}\Big(\sqrt{\frac{H^7 K_j \iota}{2}}\Big) \\ \implies & \frac{K_j V_1^{*(w_2)}}{2} \geq \frac{K_j V_1^{*(w^*)}}{2} - \mathcal{O}\Big(\sqrt{H^7 K_j \iota}\Big) \end{aligned} \tag{A.19}$$

Since $E_{K_j}^{w_2} \geq K_j V_1^{*(w_2)} - \mathcal{O}\Big(\sqrt{H^7 K_j \iota}\Big)$ by Inequality (2.5) of Lemma 1, we know that

$$E_{K_j}^{w_2} \geq K_j V_1^{*(w^*)} - \mathcal{O}\Big(\sqrt{H^7 K_j \iota}\Big) \tag{A.20}$$

with probability at least $1-\frac{1}{T^4}$. Therefore, the total regret from playing arm w_2 is

$$K_j V_1^{*(w^*)} - E_{K_j}^{w_2} \leq \mathcal{O}(\sqrt{H^7 K_j \iota}) \tag{A.21}$$

Summing over all suboptimal arms to find the total regret incurred when pulling arms,

$$Regret_{arms} \leq \sum_{w' \in [H]} \mathbf{E}\Big[K_{j(w')} V_1^{*(w^*)} - E_{K_{j(w')}}^{w'}\Big]$$

Since $\sum_{w' \in [H]} K_{j(w')} \leq K$, Jensen's Inequality implies $\sum_{w' \in [H]} \sqrt{K_{j(w')}} \leq H \cdot \sqrt{\frac{K}{H}} = \sqrt{KH}$. Therefore, the total regret incurred by pulling arms is upper-bounded by $\mathcal{O}(\sqrt{H^7 K_j \iota})$.

The Regret$_{gap}$ term in the total regret caused by discretization contributes $\mathcal{O}(1/T)$ to the regret. By Lemma 3, switching between arms contributes Regret$_{switching}$ = $\mathcal{O}(H\gamma \log^2 T)$ to the total regret. The low probability T^{-4} of failure in applying Azuma-Hoeffding has negligible regret contribution. Therefore the main regret term is given by the regret accumulated while pulling arms Regret$_{arms}$

$$\begin{aligned} \text{Regret}_{total} =& \text{Regret}_{gap} + \text{Regret}_{arms} + \text{Regret}_{switching} \\ \leq& \mathcal{O}(1/T) + \left(\mathcal{O}(\sqrt{H^7 T \iota}) \times 1 + \mathcal{O}(T^2) \times \frac{1}{T^4}\right) + \mathcal{O}(H\gamma \log^2 T) \\ =& \tilde{\mathcal{O}}\big(\sqrt{H^7 T}\big). \end{aligned} \tag{A.22}$$

□

A.4 Assumption of 0 Purchasing Costs

We want to show that for our episodic lost-sales model, we can amortize the unit purchasing costs c_h into unit holding costs o_h and unit lost-sales penalty p_h. First we know that for any $h \geq 2$

$$\begin{aligned} y_h - x_h = y_h - D_h + D_h - x_h &= (y_h - D_h)^+ - (D_h - y_h)^+ + D_h - x_h \\ &= (y_h - D_h)^+ - (D_h - y_h)^+ + D_h - (y_{t-1} - D_{t-1})^+ \end{aligned} \tag{A.23}$$

Then the total sum of costs starting from time step 2 is

$$\begin{aligned} &\sum_{h=2}^{H} \Big(c_h(y_h - x_h) + o_h(y_h - D_h)^+ + p_h(D_h - y_h)^+ \Big) \\ =&\sum_{h=2}^{H} \Big(c_h(y_h - D_h)^+ - c_h(D_h - y_h)^+ + c_h D_h - c_h(y_{t-1} - D_{t-1})^+ \\ &+ o_h(y_h - D_h)^+ + p_h(D_h - y_h)^+ \Big) \\ =&\sum_{h=2}^{H} \Big(c_h D_h - c_h(y_{h-1} - D_{h-1})^+ + (o_h + c_h)(y_h - D_h)^+ + (p_h - c_h)(D_h - y_h)^+ \Big) \end{aligned}$$

And the cost of stage 1 is equal to $o_1(y_1 - D_1)^+ + p_1(D_1 - y_1)^+ + c_1\big((y_1 - D_1)^+ - (D_1 - y_1)^+ + D_1 - x_1\big)$.

Let $c_{H+1} \geq 0$ denote the salvage price at which we sell the remaining inventory $(y_H - D_H)^+$ at the end of each episode. Then the total sum of costs from stage 1 to

H is

$$\sum_{h=2}^{H}\Big(c_hD_h - c_h(y_{h-1}-D_{h-1})^+ + (o_h+c_h)(y_h-D_h)^+ + (p_h-c_h)(D_h-y_h)^+\Big)$$
$$+ c_1(y_1-D_1)^+ - c_1(D_1-y_1)^+ + c_1D_1 - c_1x_1 + o_1(y_1-D_1)^+ + p_1(D_1-y_1)^+$$
$$- c_{H+1}(y_H-D_H)^+$$
$$= \sum_{h=2}^{H}\Big(c_hD_h - c_h(y_{h-1}-D_{h-1})^+ + (o_h+c_h)(y_h-D_h)^+ + (p_h-c_h)(D_h-y_h)^+\Big)$$
$$+ c_1(y_1-D_1)^+ - c_1(D_1-y_1)^+ + c_1D_1 - c_1x_1 + o_1(y_1-D_1)^+ + p_1(D_1-y_1)^+$$
$$- c_{H+1}(y_H-D_H)^+$$
$$= \sum_{h=1}^{H} c_hD_h + \sum_{h=1}^{H}\Big((o_h+c_h-c_{h+1})(y_h-D_h)^+ + (p_h-c_h)(D_h-y_h)^+\Big) - c_1x_1$$

Since $\sum_{h=1}^{H} c_hD_h$ and $-c_1x_1$ are fixed costs independent of our action, we can take them out of our consideration. Then the cost of each stage h is just $o'_h(y_h - D_h)^+ + p'_h(D_h - y_h)^+$, where $o'_h = o_h + c_h - c_{h+1}$ is the adjusted holding cost, and $p'_h = p_h - c_h$ is the adjusted lost-sales penalty.

Similar amortizing works for the single-product backlogged model with zero lead time.

A.5 Preliminaries for the episodic multi-product backlogging model

We describe the MDP$(\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$ for the multi-product backlogging model in this section. The current state $\mathbf{x}_h \in \mathbb{R}^{n\times L}$ is the concatenation of the current on-hand inventory $\mathbf{I}_h$ and the list of inventories ordered in the pipeline that are still in transit

$\mathbf{y}_{h-L+1}, \mathbf{y}_{h-L+2}, \dots, \mathbf{y}_{h-1}$.

For the multi-product backlogging model, we do a similar transformation[1] on the costs so that the per-period reward of any policy over an episode is bounded by $[0, 1]$.

We discretize both the state and action spaces to consist of multiples of $\varepsilon = \frac{M-m}{T^2}$. Rounding all the demands and orders to an adjacent multiple of ε (using a fixed but arbitrary rule) transforms any continuous policy to a discrete policy with at most $\mathcal{O}(H\varepsilon)$ additive error per time-step (due to accumulation over the episode) and hence $\mathcal{O}(H\varepsilon T \times n) = \mathcal{O}(\frac{n(M-m)}{K}) = o(1)$ total additive error in the cost. Note that technically, we might round a tiny order to $\mathbf{0}$, where the reward function is not Lipschitz. However, this only helps as the reward is upper semi-continuous. Therefore solving the discretized problem with regret Regret_{MDP} solves the continuous problem with regret $\text{Regret}_{MDP} + \text{Regret}_{gap} = \text{Regret}_{MDP} + \mathcal{O}(\frac{n(M-m)}{K}) = \text{Regret}_{MDP}$, since $K = \Theta(T)$.

Since the action set for the multi-product backlogging model includes any feasible replenishment amount within the order limits, the reward and leftover inventory depend on both the state and the action. Therefore, we do not simplify the notation $Q(x, y)$ to $Q(x)$.

A.6 Regret analysis for *FQL*

For *FQL*, we are able to adopt similar notations and analysis in Jin et al. (2018) (but adapted to our full-feedback setting).

We use $[\mathbb{P}_h V_{h+1}](x, y) := \mathbb{E}_{x' \sim \mathbb{P}(\cdot|x,y)} V_{h+1}(x')$. Then the Bellman optimality equation becomes $Q^*_h(x, y) = \big(r_h + \mathbb{P}_h V^*_{h+1}\big)(x, y)$.

[1]We scale the negated costs down by a factor of $\Theta\big(n \cdot \max(F_h, M|o_h|, M|b_h|)\big)$ and then shift to the right.

FQL updates the Q values in the following way for any $(x, y) \in \mathcal{A}$ at any time step:

$$Q_h^{k+1}(x, y) \leftarrow (1 - \alpha_k) Q_h^k(x, y) + \alpha_k [r_h^{k+1}(x, y) + V_{h+1}^k(x_{h+1})] \tag{A.24}$$

Then by the definition of weights α_t^k, we have

$$Q_h^k(x, y) = \alpha_{k-1}^0 H + \sum_{j=1}^{k-1} \alpha_{k-1}^j \left[r_h^j(x, y) + V_{h+1}^j \left(x_{h+1}^j \right) \right] \tag{A.25}$$

The following two lemmas are variations of Lemma 13 and Lemma 14.

Lemma 17 *For any* $(x, y, h, k) \in \mathcal{S} \times \mathcal{A} \times [H] \times [K]$, *we have*

$$\begin{aligned}\left(Q_h^k - Q_h^*\right)(x, y) = &\alpha_{k-1}^0 \left(H - Q_h^*(x, y)\right) + \sum_{i=1}^{k-1} \alpha_{k-1}^i \Bigg[\left(V_{h+1}^i - V_{h+1}^*\right)\left(x_{h+1}^i\right) + r_h^i \\ &- \mathbb{E}[r_h^i] + \left[\left(\hat{\mathbb{P}}_h^i - \mathbb{P}_h \right) V_{h+1}^* \right](x, y) \Bigg]\end{aligned}$$

Proof: From the Bellman optimality equation $Q_h^*(x, y) = \mathbb{E}[r_h(x, y)] + \mathbb{P}_h V_{h+1}^*(x, y)$, our notation $\left[\hat{\mathbb{P}}_h^i V_{h+1}\right](x, y) := V_{h+1}\left(x_{h+1}^i\right)$, and the fact that $\sum_{i=0}^{k-1} \alpha_{k-1}^i = 1$, we have

$$Q_h^*(x, y) = \alpha_{k-1}^0 Q_h^*(x, y) + \sum_{i=1}^{k-1} \alpha_{k-1}^i \Big[\mathbb{E}[r_h^i(x, y)] + \left(\mathbb{P}_h - \hat{\mathbb{P}}_h^i \right) V_{h+1}^*(x, y) + V_{h+1}^*\left(x_{h+1}^i\right) \Big]$$

Subtracting Equation A.25 from this equation gives us Lemma 17. $\square$

Lemma 18 *For any* $p \in (0, 1)$, *with probability at least* $1 - p$, *for any* $(x, y, h, k) \in$

$\mathcal{S} \times \mathcal{A} \times [H] \times [K]$, let $\iota = \log(SAT/p)$, we have for some absolute constant c:

$$0 \leq \left(Q_h^k - Q_h^*\right)(x,y) \leq \alpha_{k-1}^0 H + \sum_{i=1}^{k-1} \alpha_{k-1}^i \left(V_{h+1}^i - V_{h+1}^*\right)\left(x_{h+1}^i\right) + c\sqrt{\frac{H^3\iota}{k-1}} \tag{A.26}$$

Proof: Proof For any $i \in [k]$, recall that episode i is the episode where the state-action pair (x,y) was updated at stage h for the ith time. Let $\mathcal{F}_h^i$ be the σ-field generated by all the random variables until episode i, stage h. Then for any $\tau \in [K]$, $\left([(\hat{\mathbb{P}}_h^i - \mathbb{P}_h)V_{h+1}^*](x,y) + r_h^i - \mathbb{E}[r_h^i]\right)_{i=1}^{\tau}$ is a martingale difference sequence with respect to the filtration $\{\mathcal{F}_h^i\}_{i\geq 0}$. Then by Azuma-Hoeffding Theorem, we have that with probability at least $1 - p/SAT$:

$$\left|\sum_{i=1}^{k-1} \alpha_k^i \cdot \left[\left(\hat{\mathbb{P}}_h^i - \mathbb{P}_h\right) V_{h+1}^*\right](x,y) + r_h^i - \mathbb{E}[r_h^i]\right| \leq \frac{cH}{2}\sqrt{\sum_{i=1}^{k-1}\left(\alpha_{k-1}^i\right)^2 \cdot \iota} \leq c\sqrt{\frac{H^3\iota}{k-1}} \tag{A.27}$$

for some constant c.

Now we union bound over states, actions and times, we see that with probability at least $1-p$, we have

$$\left|\sum_{i=1}^{k-1} \alpha_{k=1}^i \left[\left(\hat{\mathbb{P}}_h^{k_i} - \mathbb{P}_h\right) V_{h+1}^*\right](x,y) + r_h^i - \mathbb{E}[r_h^i]\right| \leq c\sqrt{\frac{H^3\iota}{k-1}} \tag{A.28}$$

Then the right-hand side of Lemma 18 follows from Lemma 17 and Inequality (A.28). The left-hand side also follows from Lemma 17 and Inequality (A.28) using induction on $h = H, H-1, \ldots, 1$. □

Proof: **Proof of Theorem 4:** Define $\Delta_h^k := \left(V_h^k - V_h^{\pi_k}\right)\left(x_h^k\right)$ and $\phi_h^k :=$

$\left(V_h^k - V_h^*\right)\left(x_h^k\right)$.

By Lemma A.27, with $1-p$ probability, $Q_h^k \geq Q_h^*$ and thus $V_h^k \geq V_h^*$. Thus the total regret can be upper bounded:

$$\text{Regret}(K) = \sum_{k=1}^{K}\left(V_1^* - V_1^{\pi_k}\right)(x_1^k) \leq \sum_{k=1}^{K}\left(V_1^k - V_1^{\pi_k}\right)\left(x_1^k\right) = \sum_{k=1}^{K}\Delta_1^k$$

The main idea of the rest of the proof is to upper bound $\sum_{k=1}^{K}\Delta_h^k$ by the next step $\sum_{k=1}^{K}\Delta_{h+1}^k$, which gives a recursive formula to obtain the total regret. Here y_h^k denotes the base-stock levels taken at stage h of episode k, which means $y_h^k = \arg\max Q_h^k(y')$.

$$\begin{aligned}
\Delta_h^k &= \left(V_h^k - V_h^{\pi_k}\right)\left(x_h^k\right) \overset{(1)}{\leq} \left(Q_h^k - Q_h^{\pi_k}\right)\left(x_h^k, y_h^k\right) \\
&= \left(Q_h^k - Q_h^*\right)\left(x_h^k, y_h^k\right) + \left(Q_h^* - Q_h^{\pi_k}\right)\left(x_h^k, y_h^k\right) \\
&\overset{(2)}{\leq} \alpha_{k-1}^0 H + \sum_{i=1}^{k-1}\alpha_{k-1}^i\phi_{h+1}^i + c\sqrt{\frac{H^3\iota}{k-1}} + \left[\mathbb{P}_h\left(V_{h+1}^* - V_{h+1}^{\pi_k}\right)\right]\left(x_h^k, y_h^k\right) \\
&= \alpha_{k-1}^0 H + \sum_{i=1}^{k-1}\alpha_{k-1}^i\phi_{h+1}^i + c\sqrt{\frac{H^3\iota}{k-1}} + \left[\left(\mathbb{P}_h - \hat{\mathbb{P}}_h^k\right)\left(V_{h+1}^* - V_{h+1}^{\pi_k}\right)\right]\left(x_h^k, y_h^k\right) \\
&\quad + \left(V_{h+1}^* - V_{h+1}^{\pi_k}\right)\left(x_{h+1}^k\right) \\
&\overset{(3)}{=} \alpha_{k-1}^0 H + \sum_{i=1}^{k-1}\alpha_{k-1}^i\phi_{h+1}^i + c\sqrt{\frac{H^3\iota}{k-1}} - \phi_{h+1}^k + \Delta_{h+1}^k + \xi_{h+1}^k
\end{aligned} \tag{A.29}$$

where $\xi_{h+1}^k := \left[\left(\mathbb{P}_h - \hat{\mathbb{P}}_h^k\right)\left(V_{h+1}^* - V_{h+1}^{\pi_k}\right)\right]\left(x_h^k, y_h^k\right)$ is a martingale difference sequence. Inequality (1) holds because $V_h^k\left(x_h^k\right) \leq \max_{\text{feasible } y' \text{ given } x} Q_h^k\left(x_h^k, y'\right) = Q_h^k\left(x_h^k, y_h^k\right)$, and Inequality (2) holds by Lemma 18 and the Bellman equations. Inequality (3) holds by definition $\Delta_{h+1}^k - \phi_{h+1}^k = \left(V_{h+1}^* - V_{h+1}^{\pi_k}\right)\left(x_{h+1}^k\right)$.

In order to compute $\sum_{k=1}^{K}\Delta_1^k$, we need to first bound the first term in Equation

A.29. Since $\alpha_k^0 = 0, \forall k \geq 1$, we know that $\sum_{k=1}^K \alpha_{k-1}^0 H \leq H$.

Now we bound the sum of the second term in Equation A.29 over the episodes by regrouping:

$$\sum_{k=2}^{K} \sum_{i=1}^{k-1} \alpha_{k-1}^i \phi_{h+1}^i \leq \sum_{i=1}^{K-1} \phi_{h+1}^i \sum_{k=i+1}^{\infty} \alpha_{k-1}^i \leq \sum_{i=1}^{K-1} \phi_{h+1}^i \sum_{k'=i}^{\infty} \alpha_{k'}^i \leq \left(1 + \frac{1}{H}\right) \sum_{k=1}^{K} \phi_{h+1}^k \tag{A.30}$$

where the last inequality uses $\sum_{t=i}^{\infty} \alpha_t^i = 1 + \frac{1}{H}$ for every $i \geq 1$ from Lemma 11.

Plugging the above Equation (A.30) and $\sum_{k=1}^K \alpha_k^0 H \leq H$ back into Equation (A.29), we have:

$$\begin{aligned}
\sum_{k=1}^{K} \Delta_h^k \leq & H + \sum_{k=2}^{K} \Delta_h^k \\
\leq & H + H + \left(1 + \frac{1}{H}\right) \sum_{k=1}^{K} \phi_{h+1}^k - \sum_{k=2}^{K} \phi_{h+1}^k + \sum_{k=2}^{K} \Delta_{h+1}^k + \sum_{k=2}^{K} c\sqrt{\frac{H^3 \iota}{k-1}} \\
& + \sum_{k=2}^{K} \xi_{h+1}^k \\
\leq & 2H + \phi_{h+1}^1 + \frac{1}{H} \sum_{k=2}^{K} \phi_{h+1}^k + \sum_{k=2}^{K} \Delta_{h+1}^k + \sum_{k=2}^{K} c\sqrt{\frac{H^3 \iota}{k-1}} + \sum_{k=2}^{K} \xi_{h+1}^k \\
\leq & 3H + \left(1 + \frac{1}{H}\right) \sum_{k=2}^{K} \Delta_{h+1}^k + \sum_{k=2}^{K} c\sqrt{\frac{H^3 \iota}{k-1}} + \sum_{k=}^{K} \xi_{h+1}^k
\end{aligned} \tag{A.31}$$

where the last inequality uses $\phi_{h+1}^k \leq \Delta_{h+1}^k$. By recursing on $h = 1, 2, \ldots, H$, and because $\Delta_{H+1}^K = 0$, we have:

$$\sum_{k=1}^{K} \Delta_1^k \leq \mathcal{O}\left(\sum_{h=1}^{H} \sum_{k=1}^{K} \left(c\sqrt{\frac{H^3 \iota}{k-1}} + \xi_{h+1}^k\right)\right)$$

where
$$\sum_{h=1}^{H}\sum_{k=1}^{K} c\sqrt{\frac{H^3\iota}{k-1}} = \mathcal{O}(H\sqrt{H^3\log(SAT/p)}\sqrt{K}) = \tilde{\mathcal{O}}(\sqrt{H^4T})$$

On the other hand, by Azuma-Hoeffding inequality, with probability $1-p$, we have

$$\begin{aligned}\left|\sum_{h=1}^{H}\sum_{k=1}^{K}\xi_{h+1}^{k}\right| &= \left|\sum_{h=1}^{H}\sum_{k=1}^{K}\left[\left(\mathbb{P}_h - \hat{\mathbb{P}}_h^k\right)\left(V_{h+1}^{*} - V_{h+1}^{\pi_k}\right)\right](x_h^k, y_h^k)\right| \\ &\leq cH\sqrt{T_l} \ \leq \tilde{\mathcal{O}}(\sqrt{H^4T})\end{aligned} \tag{A.32}$$

which establishes $\sum_{k=1}^{K}\Delta_1^k \leq \tilde{\mathcal{O}}(H^2\sqrt{T})$.

$$\text{Regret}_{total}(K) = \text{Regret}_{MDP}(K) + \text{Regret}_{MDP}(gap) = \mathcal{O}(H^2\sqrt{n(L+1)T\log T}) \tag{A.33}$$

We multiply the constant $O\big(n \cdot \max(F_h, M|o_h|, M|b_h|)\big)$ back because we previously scaled the costs to have the reward for each time period bounded by 1. This yields a $\mathcal{O}(H^2 n\sqrt{n(L+1)} \cdot \max(F_h, M|o_h|, M|b_h|))$ total dependence on setting parameters for our $\tilde{\mathcal{O}}(T)$ regret. □

When $L = 0, n = 1, F_h = 0$, the total regret of *FQL* on the single-product backlogging model with a lead time and an order limit is $\tilde{\mathcal{O}}(\sqrt{T})$ with an

$$\mathcal{O}\big(H^2 M \max(|o_h|, |b_h|)\big)$$

dependence on all constant parameters. This is smaller than the dependence of *HQL* applied on the single-product backlogging model with a lead time and an order limit by a factor of H.

A.7 Regret analysis for *MimicQL*

Proof: Proof of Theorem 5:

Let ℓ denote the maximum order limit. Let o denote the maximum unit holding cost. The expected amount of time until synchronization is no more than $\mathcal{O}(nL\ell\gamma)$. During each of these time steps the holding cost is a constant nMo. Then the additional cost *Mimic-FQL* incurs each time by not discarding is bounded by $\mathcal{O}(n^2\ell ML\gamma o)$. This is a constant term, but since discarding happens at the end of every episode, the total additional regret incurred is $\mathcal{O}(Kn^2\ell ML\gamma o)$. Note that this term is linear in T when $K = \Theta(T)$, and we will perform additional techniques to obtain a total regret that is sublinear in T.

With positive lead time, the optimal policy OPT in the non-discarding model can have a larger or smaller total expected cost than the optimal policy $\underline{OPT}$ for the intermediate MDP. We want to show that Cost of the optimal policy OPT in the non-discarding model, will not be too much lower than the $\underline{\text{Cost}}$ of the optimal policy $\underline{OPT}$ for the intermediate MDP.

Consider a policy π_1 on the intermediate MDP. At the beginning of each episode, π_1 starts with zero inventory and zero replenishment because discarding at the end of the previous episode. In the first L time steps of the second episode, π_1 orders the replenishment in a way that it ends up with the same inventory vector and replenishment vector as OPT at the end of L time steps. Starting at time step $(L+1)$, π_1 completely copies OPT under the beginning of the next episode, where π_1 starts with zero inventory and replenishment again. For each episode, the cost of this policy π_1 is at most $\mathcal{O}\Big(LF + LnMc\Big)$ more than the cost of OPT. Therefore, the total expected cost of π_1 is at most $\mathcal{O}\Big(KLF + KLnMc\Big)$ more than the total expected cost of OPT. On the other hand, since by definition, the total expected cost of the optimal policy on the intermediate MDP is no higher than the total expected cost of π_1, we know that the total expected cost of $\underline{OPT}$ is at most $\mathcal{O}\Big(KLF + KLnMc\Big)$ more than the total expected cost of OPT.

So far we have argued that in the case of zero lead time,

$$\begin{aligned}
&1.\ \underline{Cost_T}(\underline{OPT}) - \mathcal{O}\Big(KLF + KLnMc\Big) \leq \text{Cost of } OPT \\
&2.\ \underline{\text{Cost}} \text{ of } FQL \leq \text{Cost of } \textit{Mimic-FQL} \leq \underline{\text{Cost}} \text{ of } FQL + \mathcal{O}\big(Kn^2\ell ML\gamma o\big)
\end{aligned} \tag{A.34}$$

Then we know that

$$\begin{aligned}
\text{Regret}_{Mimic-FQL} :=& \text{Cost of } \textit{Mimic-FQL} - \text{Cost of } OPT \\
\leq& \text{Cost of } \textit{Mimic-FQL} - \text{Cost of } \underline{OPT} + \text{Cost of } \underline{OPT} \\
&- \text{Cost of } OPT \\
\leq& \text{Cost of } \textit{Mimic-FQL} - \text{Cost of } \underline{OPT} + \mathcal{O}\Big(KLF + KLnMc\Big) \\
\leq& \Big(\text{Cost of } \textit{Mimic-FQL} - \underline{\text{Cost}} \text{ of } FQL\Big) + \Big(\underline{\text{Cost}} \text{ of } FQL \\
&- \underline{\text{Cost}} \text{ of } \underline{OPT}\Big) + \mathcal{O}\Big(KLF + KLnMc\Big) \\
\leq& \mathcal{O}\big(Kn^2\ell ML\gamma o\big) + \tilde{\mathcal{O}}\big(J^2\sqrt{T}\big) + \mathcal{O}\Big(KLF + KLnMc\Big)
\end{aligned} \tag{A.35}$$

where we recall that the second term on the last line is bounded by Theorem 1 for the episodic model.

Since $K := \frac{T}{J}$, we know that $\text{Regret}_{Mimic-FQL} \leq \mathcal{O}\big(\frac{T(n^2\ell ML\gamma o + LF + LnMc)}{J}\big) + \tilde{\mathcal{O}}\big(J^2\sqrt{T}\big)$. Choosing J to be a multiple of H of size $J = \Theta(T^{1/6})$ now yields the regret bound $\tilde{\mathcal{O}}\big(T^{5/6}\big)$. Note that $T^{1/6}$ might not be an integer multiple of H, then we take J to be the closest multiple of H to $T^{1/6}$. $\square$

A.8 General definitions and assumptions for wider application of our policies

A.8.1 Full Feedback.

The formal definition of *full feedback* is as follows. Immediately after taking an action a_t at time t, once the environmental randomness D_t is realized, the agent learns what the counterfactual reward $r_t(s, a)$ and next state $s_{t+1}(s, a)$ would have been for all feasible state action pairs $(s, a) \in \mathcal{S} \times \mathcal{A}$ for that specific time step t.

We notice this feedback structure in the backlogging inventory problems: in the backlogging model, we observe the actual realized demand, which allows us to deduce what the cost and leftover inventory would be for any action. Trivially, the backlogging model also possesses one-sided feedback.

For problems that possess the full-feedback structure, *FQL* is applicable with our regret bound guarantee.

A.8.2 One-Sided Feedback.

The formal definition of *one-sided feedback* is as follows. Immediately after taking an action a at step t, once the environmental randomness D_t is realized, we learn what the reward and next state would have been if any action that lie on *one side* of a is taken, i.e., all $a' \leq a$ for the *lower*-sided-feedback structure for that specific time step t (or all $a' \geq a$ for the *higher*-sided-feedback structure). This implies that the action space can be embedded in a compact subset of $\mathbb{R}$.

We notice this feedback structure is in the lost-sales inventory control problem: once the demand D_t is realized for that time step, if the demand is lower than our chosen base-stock level y_t, we will observe the actual D_t; otherwise we will observe the $\min(y_t, D_t)$ part of the demand, which lets us deduce what the pseudo-cost and leftover inventory would be if the agent had taken any action (base-stock level) lower than y_t. Mathematically, for any $y_t' \leq y_t$, $\min(y_t', D_t) = \min(y_t', \min(y_t, D_t))$.

We list a number of assumptions that need to hold for the *lower*-sided-feedback setting. In the case of the *higher*-sided-feedback setting, Assumptions 2 and 3 would be symmetric to Assumptions 2 and 3 below. If the set of feasible actions at any time is unaffected by the current state, then the assumptions below are unnecessary. However, in that case, even though our algorithm still applies, the MDP problem can be reduced to a number of bandit problems.

Assumptions (lower-sided):

1. The optimal Q-value functions are concave.
2. The current feasible action set at time t is of the form $\mathcal{A} \cap [a, \infty)$, for some $a \in \mathbb{R}$ non-decreasing in x_t.
3. Conditioned on the environmental randomness, the next state $x_{t+1}(\cdot)$ is non-decreasing in y_t.
4. The reward and transition only depend on the action, the time step and the environmental randomness, even though the feasible action set can depend on the state. So $Q(x, y)$ can be simplified to $Q(y), \forall y$ feasible for x.

These assumptions impose a specific structure on the problem, which is often satisfied in important OR and finance problems, e.g. inventory control, portfolio management, airline's overbook policy, online second price auctions, etc. See an overview of these applications in Section 3.4.

Appendix B

Appendix for Chapter 4

B.1 Proof of Theorem 7

Before we prove the above theorem, consider first the special case of algorithms that always match a user to some available product if such a matching is possible. Suppose we have a single unit of a single product with reward 1 and the following arrival sequences,

- Sequence A: A single user with usage time duration ∞ (never returns the product).
- Sequence B: A user with usage duration ∞, followed by T users that return the product right away i.e., $\mathbb{P}(d_t = 0) = 1$ for all $t \in \{2, \dots, T+1\}$.

In order to be competitive on sequence A, the algorithm must match the arrival with the only available product. Consequently, even on sequence B the algorithm will match the product to the first user and earn a net reward of 1. An optimal offline algorithm would earn total reward T on sequence B hence, an online algorithm that always matches an arriving costumer if possible can never have competitive ratio better than $O\big(\frac{1}{T}\big)$.

Let c denote the capacity of the resource. For the general case, consider the following family of arrival sequences and subsequent lemma.

- Sequence $C(c,t)$: cT^t users, each with identical usage duration distribution where the item is either returned immediately with probability $p_t = 1 - \frac{1}{T^t}$ or never returned i.e., $\mathbb{P}(d_t = 0) = p_t = 1 - \frac{1}{T^t}$ and $\mathbb{P}(d_t = \infty) = 1 - p_t = \frac{1}{T^t}$.

In the following, we focus on *equitable* algorithms that treat all units of the product equally. This simplifies the arguments and is without loss of generality for the overall result. Formally, since all c units of the product are identical, w.l.o.g., each time a unit of the product is to be matched we let the algorithm randomly pick an available unit for the match. If the algorithm has net expected revenue at least R, then this ensures that the expected revenue from allocating any individual unit is R/c as all units are treated equitably. Any algorithm can be turned into an equitable one without change in total revenue.

It is worth noting that an algorithm may for computational reasons differentiate between units of the same resources. Indeed, subsequent work (Goyal et al., 2021) introduces this idea and demonstrates that such differentiation can help in addressing reusability. Our notion of equity applies not at the computational level but to the final allocation. Borrowing an algorithmic idea from Goyal et al. (2021), we explain this in more detail through an example. Consider an instance with a 2 identical units of a single reusable resource that we refer to as unit A and B and a sequence of arrivals all requiring a unit of the resource. Suppose we have an algorithm that computationally maintains a state $(s_1(t), s_2(t)) \in \{0,1\}^2$ and decides whether to allocate a unit of the resource to arrival t based on the state. Initially $(s_1(1), s_2(1)) = (1,1)$

and we ensure that $s_1(t) + s_2(t) \geq 1$ only if at least one unit of the resource is available at t. The following describes the state based allocation rule and corresponding state update: (i) if the state at t is $(1,1)$ or $(1,0)$ the algorithm allocates a unit at t and updates $s_1(t+1) = 0$, (ii) if the state is $(0,1)$ then with probability (w.p.) 0.5 it allocates a unit and updates $s_2(t+1) = 0$ and w.p. 0.5 it rejects t, (iii) in state $(0,0)$ it rejects t. When a unit returns from use at t and the unit was allocated while in state $(1,1)$ or $(1,0)$, we update $s_1(t) = 1$. In all other cases when a unit returns at t we update $s_2(t) = 1$. Observe that both state $(1,0)$ and state $(0,1)$ indicate that exactly one unit is available, but the algorithm behaves differently in the two scenarios. However, this is a purely computational differentiation. Whenever the algorithm makes an allocation, if we have both A and B available we pick one for allocation uniformly randomly. Thus, the expected number of times A and B are matched is the same and the algorithm is equitable. With this in mind consider the following lemma,

Lemma 19 *Given capacity $c \geq 1$, $t \in [T]$, and arrival sequence $C(c,t)$. If an equitable algorithm generates expected revenue at least $c\,\frac{1-p_t^{\alpha T^t}}{1-p_t}$ for some $\alpha \in [0,1]$, then for every individual unit of the product, the probability that the unit is consumed forever after the last arrival is at least $1 - p_t^{\alpha T^t}$.*

Proof: Suppose that a single unit, i, is attempted to be matched y times i.e., unit i is matched repeatedly every time it returns from a finite use, for up to y times in total. Then the expected total reward from matching i is,

$$R_i(y) = (1-p_t) + 2p_t(1-p_t) + \cdots + yp_t^{y-1} = \frac{1-p_t^y}{1-p_t}.$$

Observe that the expected reward is $\frac{1}{1-p_t}$ times the probability $1-p_t^y$ that the unit is consumed forever (extinguished) when matched up to y times. Now, for any unit i of the resource the maximum number, y, of match attempts is a random variable. Formally, define random variable $\mathbf{Y}_i$ as the number of times unit i is matched given that the unit always has a finite usage duration. Further, we independently sample usage durations for i and let $\boldsymbol{\tau}_i$ denote the (random) number of finite usage durations before a duration of $+\infty$. Clearly, $\mathbf{Y}_i$ is independent of $\boldsymbol{\tau}_i$ and therefore, the expected revenue from matching unit i in the algorithm is

$$\begin{aligned}\mathbb{E}[R_i(\mathbf{Y}_i, \boldsymbol{\tau}_i)] &= \sum_y \mathbb{P}[\mathbf{Y}_i = y]\, \mathbb{E}[R_i(y, \boldsymbol{\tau}_i)] \\ &= \sum_y \mathbb{P}[\mathbf{Y}_i = y] \frac{\mathbb{P}[i \text{ extinguished} \mid \mathbf{Y}_i = y]}{1 - p_t} \\ &= \frac{\mathbb{P}[i \text{ extinguished}]}{1 - p_t}.\end{aligned}$$

So if the expected revenue from matching unit i in the algorithm is at least $\frac{1-p_t^{\alpha T^t}}{1-p_t}$, then the probability that i is extinguished is at least, $1 - p_t^{\alpha T^t}$. To complete the proof, recall that in an equitable algorithm the expected revenue from an individual unit is $1/c$ fraction of the total expected revenue. Given an equitable algorithm with total revenue at least $c\,\frac{1-p_t^{\alpha T^t}}{1-p_t}$, we have revenue at least $\frac{1-p_t^{\alpha T^t}}{1-p_t}$ from an individual unit. So the probability that any given unit i survives is at most $p_t^{\alpha T^t}$. Observe that for $T \to \infty$, $1 - p_t^{\alpha T^t} \to 1 - e^{-\alpha}$.

□

Corollary 4 *For any given capacity c and $t \in [T]$, the maximum expected revenue generated by any algorithm (online or offline) on arrival sequence $C(c,t)$ is at least $c\,\frac{1-p_t^{T^t}}{1-p_t}$ and at most $c\,T^t$.*

Proof: Clearly, the maximum revenue is at most cT^t. For the lower bound, consider the algorithm that attempts to match each unit of the product to T^t arrivals. From the analysis of Lemma 19, we have that a single unit of the product generates maximum expected revenue $\frac{1-p_t^{T^t}}{1-p_t} = \Theta(T^t)$. □

We are now ready to prove Theorem 7.

Proof: Proof of Theorem 7: For arbitrary capacity $c \geq 1$, consider T sequences $D(t) = \{C(c,1), \ldots, C(c,T)\}$ for $t \in [T]$ that begin with cT users arriving from sequence $C(c,1)$ followed by cT^2 users from sequence $C(c,2)$ and so on in order till $C(c,T)$. For any sequence $D(t)$ the maximum possible expected revenue is $\Theta(cT^t)$, since it is lower bounded by $c\,(1 - p_t^{T^t})T^t = \Omega(cT^t)$ (matching only the users in $C(c,t)$ while ignoring earlier users and using Corollary 4) and is upper bounded by, $c\,\left(\sum_{k=1}^{t} T^t\right) = O(cT^t)$.

We prove by contradiction. Consider a β-competitive online algorithm and assume $\beta = \Omega\big(\frac{\log T}{T}\big)$ (otherwise we are done). W.l.o.g., let the algorithm be equitable towards units of the products. From the assumption on competitiveness and using Corollary 4, on arrival sequence $D(1)$ the expected revenue of the online algorithm from an individual unit must be at least $\beta(1 - p_1^T)T$. From Lemma 19 we have that the probability the unit is available after all arrivals is at most $1 - \beta(1 - p_1^T) \to 1 - \beta(1 - 1/e)$. Now similar to case of $D(1)$, in order to be β-competitive on sequence $D(2)$ where the maximum expected profit is $\Theta(cT^2)$, the expected reward generated from the $C(c,2)$ part of sequence $D(2)$ must be at least $\beta c\,(1-p_2^{T^2})T^2$, as the contribution from arrivals $C(c,1)$ is at most $\Theta(cT) = c \times o(\beta T^2)$ for $\beta = \Omega\big(\frac{\log T}{T}\big)$. Focusing again on an individual unit and applying Lemma 19, the probability of the unit surviving after all arrivals from $C(c,2)$ part of sequence $D(2)$, conditioned on the unit surviving after arrivals from $C(c,1)$ part of $D(2)$, is at most $1 - \beta(1 - p_2^{T^2}) \to 1 - \beta(1 - 1/e)$. Thus, the probability of the unit surviving after all arrivals in $D(2)$ is at most $(1 - \beta(1 - 1/e))^2$. More generally, it follows that the probability of an individual unit surviving after arrivals from sequence $D(t)$ is at most $(1 - \beta(1 - 1/e))^t$. Therefore, on sequence $D(T)$ there is at most a $(1 - \beta(1 - 1/e))^{T-1}$ probability that an individual unit survives until the first arrival from the $C(c, T)$ part of $D(T)$. Hence, the overall expected revenue on $D(T)$ is, $c \times O\left(\max\left\{(1 - \beta(1 - 1/e))^{T-1}\, T^T, T^{T-1}\right\}\right)$. Therefore, the competitive ratio β of the algorithm must satisfy, $\beta \leq O\left(\max\left\{(1 - \beta(1 - 1/e))^{T-1}, \frac{1}{T}\right\}\right)$. This translates to $\beta \leq O(\frac{1}{T})$ for $\beta \geq \frac{2\log T}{T}$, a contradiction. Therefore, β is no larger than $\frac{2\log T}{T}$. Note that a more refined argument can be used to further tighten the log factor.

$\square$

Appendix C

Appendix for Chapter 5

C.1 Example for Remark 3

We show an example where the constraints of the second stage problem, equipped with a different integral objective function, do not admit integral optimal solutions; this implies that the constraint matrix is not totally unimodular. Suppose there are two demands, two data centers, two days, and one supplier with sufficient capacity. Suppose the demands are compatible with both data centers and the costs have the form $\sum_{d,\ell,t} c_{d,\ell,t} z_{d,\ell,t}$, where $c_{d_1,\ell_2,t_1} = c_{d_1,\ell_1,t_2} = c_{d_2,\ell_2,t_2} = 1$ and other $c_{d,\ell,t}$ are large enough so that the corresponding $z_{d,\ell,t}$ are forced to take the value 0. By constraint (5.14),

$$z_{d_1,\ell_2,t_1} + z_{d_1,\ell_1,t_2} \leq 1.$$

In addition, we assume all δ and ρ are 1. By constraint (5.17),

$$z_{d_1,\ell_2,t_1} + z_{d_2,\ell_2,t_2} \leq 1.$$

If we set $p_1 = \{\ell_1, \ell_2\}$ and $(p_1, t_2) \in \mathcal{H}$, then

$$z_{d_1,\ell_1,t_2} + c_{d_2,\ell_2,t_2} \leq 1.$$

Clearly, any feasible solution z has at most one entry 1, but $z_{d_1,\ell_2,t_1} = z_{d_1,\ell_1,t_2} = z_{d_2,\ell_2,t_2} = 0.5$ and others $z_{d,\ell,t}$'s being 0 is a feasible solution for the LP relaxation, and it cannot be written as a convex combination of the feasible binary solutions.

C.2 Proof for Lemma 8

Let z, w be a feasible solution of the LP relaxation of cost c_1. Consider the flow $f(e), e = (u, v)$, with value

- $\sum_{d,\ell,t\in\mathcal{T}_2} z_{d,\ell,t}$, where u is the sink and v is the source.
- $\sum_{d} w_{d,s}$, where u is the source and v is supplier s.
- $w_{d,s}$, where u is supplier s and v is demand d, i.
- $\sum_{\ell\in\mathcal{L}_d,t\in\mathcal{T}_2} z_{d,\ell,t}$, where u is demand d, i and v is demand d, ii.
- $\sum_{\ell\in p} z_{d,\ell,t}$, where u is demand d, ii and v is throughput p, t, i.
- $\sum_{d,\ell\in p} z_{d,\ell,t}$, where u is throughput p, t, i and v is throughput p, t, ii.
- $\sum_{d,\ell\in p_0} z_{d,\ell,t}$, where u is throughput p, t, ii and v is throughput p_0, t_0, i.
- $\sum_{d} z_{d,\ell,t}$, where u is throughput p, t, ii and v is data center ℓ.
- $\sum_{d,t\in\mathcal{T}_2} z_{d,\ell,t}$, where u is data center ℓ and v is the sink.

It can be verified from the constraints of the LP relaxation that f is a feasible flow of cost $c_1 - \sum_d u_d$.

To show the other direction, we first show that an integral flow can be mapped to an integral feasible solution of the LP relaxation. Indeed, an integral flow f can be decomposed into circulations of flow 1 (in $O(mn)$ time, see Theorem 8.8 and Proposition 9.5 in Korte and Vygen (2018)), though such decomposition may be nonunique. We claim that each such circulation corresponds to a valid assignment of a demand. Note that each circulation passes through the arc from demand d, i to demand d, ii for a unique d. For each circulation g associated with the demand d_g, we take $w_{d_g,s} = 1$ if g passes through supplier s; and we take $z_{d_g,\ell,t} = 1$ if g passes through data center ℓ and time t. We set the remaining variables to be 0.

The construction above yields a feasible integral solution of the LP relaxation of cost $c_2 + \sum_d u_d$. In general, a feasible flow of cost c_2 can be written as a convex combination of integral flows. We then construct a feasible solution of the LP relaxation by considering the convex combination of the feasible integral solutions corresponding to the integral flows, and such feasible solution gives an objective value $c_2 + \sum_d u_d$.

C.3 Proof of Lemma 9

It suffices to show that the second stage problem subsumes the set packing problem as a special case, then the hardness results (Cornuéjols (2001), Håstad (1999), and Williamson and Shmoys (2011)) of the set packing problem also apply to our problem.

Consider the following construction where $T = T_1 + 1$ and $\mathcal{L}_d = \mathcal{L}$ (hence Property 2 is satisfied trivially). Suppose $c_{d,t} = h_{d,s} = 0$, $u_d = \rho_{\ell,2} = \delta_{p,t,2} = 1$, and we take $\sigma_{s,2}$ large enough so that constraints (5.15) and (5.18) can always be satisfied for any given $z_{d,\ell,t}$'s. The original second stage problem then becomes

$$\begin{aligned} |\mathcal{D}| - \max_{z_{d,\ell,T} \in \{0,1\}} \quad & \sum_{d \in \mathcal{D}} \sum_{\ell \in \mathcal{L}} z_{d,\ell,T} \\ \text{s.t.} \quad & \sum_{\ell \in \mathcal{L}} z_{d,\ell,T} \leq 1 && \forall d \in \mathcal{D} \; (5.14) \\ & 1 \geq \sum_{d \in \mathcal{D}} \sum_{\ell \in p} z_{d,\ell,T} && \forall (p,T) \in \mathcal{H} \; (5.16) \\ & 1 \geq \sum_{d \in \mathcal{D}} z_{d,\ell,T} && \forall \ell \in \mathcal{L} \; (5.17). \end{aligned}$$

Since $\sum_{\ell} z_{d,\ell,T}$ can only be either 0 or 1, it maximizes the number of demands that can be deployed under constraints (5.16) and (5.17). If in addition we assume that $|\mathcal{D}| \geq |\mathcal{L}|$, then the problem is equivalent to the set packing problem

$$\max_{y_\ell \in \{0,1\}} \sum_{\ell \in \mathcal{L}} y_\ell \quad \text{s.t.} \sum_{\ell \in p} y_\ell \leq 1 \quad \forall (p,T) \in \mathcal{H}.$$

Indeed, for one direction, if we take a feasible solution z in the maximization, then $y_\ell = \sum_{d \in \mathcal{D}} z_{d,\ell,T}$ is a feasible solution with the same objective value. On the other hand, let y be a feasible solution of the set packing problem, since $|\mathcal{D}| \geq |\mathcal{L}|$, we can find feasible z such that $y_\ell = \sum_{d \in \mathcal{D}} z_{d,\ell,T}$, and such z yields the same objective value.

C.4 Proof of Lemma 10

Consider the following example. Suppose there are three demands, two data centers, two days, and two suppliers; demand d_1 is only compatible with data center ℓ_1 and supplier s_1, demand d_2 is only compatible with data center ℓ_1 and supplier s_2, and demand d_3 is only compatible with data center ℓ_2 and supplier s_2. Suppose in addition the costs $c_{d_1,t_1}, c_{d_2,t_2}, c_{d_3,t_1}$ are 0 and $c_{d_1,t_2}, c_{d_2,t_1}, c_{d_3,t_2}$ are large enough so that $z_{d_1,\ell_1,t_2}, z_{d_2,\ell_1,t_1}, z_{d_3,\ell_2,t_2}$ are forced to take the value 0. Thus, the free stage variables are $z_{d_1,\ell_1,t_1}, z_{d_2,\ell_1,t_2}, z_{d_3,\ell_2,t_1}, w_{d_1,s_1}, w_{d_2,s_2}$ and w_{d_3,s_2}. If the values of δ, ρ, σ are all 1, then by constraint (5.17),

$$z_{d_1,\ell_1,t_1} + z_{d_2,\ell_1,t_2} \leq 1;$$

by constraint (5.15) and (5.18),

$$z_{d_2,\ell_1,t_2} + z_{d_3,\ell_2,t_1} = w_{d_2,s_2} + w_{d_3,s_2} \leq 1;$$

if we set $p_1 = \{\ell_1, \ell_2\}$ and $(p_1, t_1) \in \mathcal{H}$, then by constraint (5.16),

$$z_{d_1,\ell_1,t_1} + z_{d_3,\ell_2,t_1} \leq 1.$$

Clearly, any binary vector z satisfying the constraints must have at most one entry 1. Consider the costs $u_d = 1$ and $h_{d,s} = 0$, the optimal value of the LP relaxation is at most 1.5 whereas the optimal value of the original formulation is 2, this implies the integrality gap is at least $\frac{4}{3}$.

C.5 Risk Measures and Subgradients

A key ingredient in our numerical approach is the subgradient of a convex function f. In this section, we show how to evaluate the subgradient of functions $f = \mathcal{R}[\tilde{Q}(\cdot, \xi)]$ for various risk measures $\mathcal{R}$, when the distribution is the empirical distribution based on a sample of scenarios $(\xi_1, \ldots, \xi_N)$ of size N. We first show how to evaluate the subgradient of $Q(\cdot, \xi_i)$ for each individual scenario ξ_i.

Subgradient for each scenario. We denote $f_i(x) = \tilde{Q}(x, \xi_i)$ for each $i = 1, \ldots, N$. Since each $f_i(x)$ corresponds to a linear program with x in the right hand side of the constraints, its dual formulation will contain x only in the objective function. In particular, its dual has the form $f_i(x) = \max_{y \in \mathcal{Y}_i} b_x' y$. A subgradient $\partial f_i(x)$ is given by an optimal solution y^* of the dual problem.

We next present several monotone and convex risk measures $\mathcal{R}$ we employed in the numerical experiments. In general, since $\tilde{Q}(\cdot, \xi)$ is convex, the composition $\mathcal{R}[\tilde{Q}(\cdot, \xi)]$ is also convex, hence a subgradient exists (the interested readers could find references regarding risk measures in Artzner et al. (1999) and Shapiro et al. (2014)).

Expectation. For the expectation $\mathbb{E}[\tilde{Q}(x, \xi)] = N^{-1} \sum_{i=1}^{N} f_i(x)$, a subgradient can be directly derived as $N^{-1} \sum_{i=1}^{N} \partial f_i(x)$.

Conditional Value-at-Risk (CVaR). As explained in Section 5.5.2, for a threshold $\alpha \in (0, 1)$, the Conditional Value-at-Risk is defined as

$$\mathcal{R}_\alpha(Z) = \alpha^{-1} \int_{1-\alpha}^{1} VaR_{1-t}(Z)\, dt,$$

where

$$VaR_\alpha(Z) = \inf\{t : Pr(Z \leq t) \geq 1 - \alpha\}$$

is the Value-at-Risk. Then, for the composition $\rho_\alpha(x) = \mathcal{R}_\alpha[\tilde{Q}(x, \xi)]$ we have:

$$\rho_\alpha(x) = \mathcal{R}_\alpha[\tilde{Q}(x, \xi)] = \frac{1}{\alpha N} \sum_{i=1}^{\lfloor \alpha N \rfloor} f_{(i)}(x) + \left(1 - \frac{\lfloor \alpha N \rfloor}{\alpha N}\right) f_{(\lfloor \alpha N \rfloor + 1)}(x),$$

where (i) is an ordering of $1, \ldots, N$ (depending on x) such that $f_{(i_1)}(x) \geq f_{(i_2)}(x)$

whenever $i_1 < i_2$. A subgradient of ρ_α at x is given by

$$\partial \rho_\alpha(x) = \frac{1}{\alpha N} \sum_{i=1}^{\lfloor \alpha N \rfloor} \partial f_{(i)}(x) + \left(1 - \frac{\lfloor \alpha N \rfloor}{\alpha N}\right) \partial f_{(\lfloor \alpha N \rfloor + 1)}(x).$$

Mean-deviation risk measure of order 1 **(MeanDev)**. For a threshold $c \in [0, 1/2]$, consider

$$\mathcal{R}_c(Z) = \mathbb{E}[Z] + c\mathbb{E}[|Z - \mathbb{E}[Z]|].$$

The composition $\rho_c(x) = \mathcal{R}_c[\tilde{Q}(x, \xi)]$ is then

$$\begin{aligned}
\rho_c(x) &= N^{-1} \sum_{i=1}^{N} f_i(x) + cN^{-1} \sum_{i=1}^{N} \left| f_i(x) - N^{-1} \sum_{i=1}^{N} f_i(x) \right| \\
&= N^{-1} \sum_{i=1}^{N} f_i(x) + cN^{-1} \sum_{i=1}^{N} \max \left(f_i(x) - N^{-1} \sum_{i=1}^{N} f_i(x), N^{-1} \sum_{i=1}^{N} f_i(x) - f_i(x) \right) \\
&= \max_{b \in \{\pm 1\}^N} N^{-1} \sum_{i=1}^{N} \left(f_i(x) + cb_i \left(f_i(x) - N^{-1} \sum_{i=1}^{N} f_i(x) \right) \right) \\
&=: \max_{b \in \{\pm 1\}^N} g_b(x).
\end{aligned}$$

It can be verified that each g_b is a convex combination of $f_i' s$, so g_b is also convex. A subgradient of ρ_c at x is given by a subgradient of g_b which attains the maximum.

C.6 Benders Decomposition

For completeness, we present a brief summary of the Benders decomposition Benders (1962). Consider the mixed integer convex program

$$\min_{x\in\mathbb{Z}^{n_1}\times\mathbb{R}^{n_2}} \quad c'x + f(x) \tag{C.1}$$

$$\text{s.t} \quad Ax \leq b, \tag{C.2}$$

where f is a convex function. Note that due to the convex approximation obtained in the previous section, our two-stage stochastic mixed integer program (5.2) - (5.12) can be written in this form with $f = \mathcal{R}[\tilde{Q}(\cdot,\xi)]$ for a convex and monotone (e.g. coherent) risk measure $\mathcal{R}$. The Benders decomposition proceeds by gradually refining a lower convex approximation of f and eventually finding a (near) optimal solution when the approximation is good enough. The procedure generates a sequence of trial points $x_0, x_1, \ldots$, and the lower convex approximation engaged is the convex piecewise linear function $\max_i f(x_i) + s_i'(x - x_i)$, where s_i is a subgradient of f at x_i.

Initialization. We start with an intial trial point x_0, e.g., it could be the optimal solution of the subproblem

$$\min_{x\in\mathbb{Z}^{n_1}\times\mathbb{R}^{n_2}} \quad c'x \tag{C.3}$$

$$\text{s.t} \quad Ax \leq b. \tag{C.4}$$

We generate a subgradient $s_0 = \partial f(x_0)$ of f at x_0 and add the constraint $\theta \geq$

$f(x_0) + s_0'(x - x_0)$ to the subproblem, which becomes

$$\min_{x \in \mathbb{Z}^{n_1} \times \mathbb{R}^{n_2}, \theta} \quad c'x + \theta \tag{C.5}$$

$$\text{s.t} \quad Ax \leq b \tag{C.6}$$

$$\theta \geq f(x_0) + s_0'(x - x_0). \tag{C.7}$$

Iteration k. At iteration k, we generate an optimal solution x_k of the subproblem

$$\min_{x \in \mathbb{Z}^{n_1} \times \mathbb{R}^{n_2}, \theta} \quad c'x + \theta \tag{C.8}$$

$$\text{s.t} \quad Ax \leq b \tag{C.9}$$

$$\theta \geq f(x_i) + s_i'(x - x_i), \quad \forall i < k. \tag{C.10}$$

A subgradient s_k of f is evaluated at x_k, and the constraint $\theta \geq f(x_k) + s_k'(x - x_k)$ is added to the subproblem. Note that the subproblem is equivalent to

$$\min_{x \in \mathbb{Z}^{n_1} \times \mathbb{R}^{n_2}, \theta} \quad c'x + \max_{i<k} \{f(x_i) + s_i'(x - x_i)\} \tag{C.11}$$

$$\text{s.t} \quad Ax \leq b. \tag{C.12}$$

Since the lower approximation $\max_i f(x_i) + s_i'(x - x_i)$ is tight at the trial points $x_0, x_1, \ldots$, it can be shown that the Benders decomposition never visits a suboptimal solution twice. Note that the first stage decision variables in our problem are bounded and integral, hence the procedure finds an optimal solution in a finite number of iterations.

C.6.1 General Row Building Lead Times

For simplicity of exposition, in the main sections of the chapter we focused on the basic optimization model where the duration L for building a new row is the same as the duration of the first stage T_1 for which the demands are purely deterministic. We now remove this assumption and generalize this model to allow arbitrary row building times, as well as to account for rows whose building is in progress at the start of the horizon.

Building in progress at the start of the horizon. In Section 5.3, we assume that the cloud provider may have already built some rows in the data centers before the horizon starts; these are denoted by $\rho_{\ell,1}$ for each data center ℓ and are available to use at the start of the horizon. In practice, however, since row building requires weeks to complete, rows may be at various stages of building, and hence they will become available at different times over the course of the horizon.

To capture this temporal aspect, let $\rho_{\ell,t,1}$ denote the number of existing rows in data center ℓ that will be ready before or at time t (this is the number of rows that will be available for the demands at time t assuming none were used before that time). We similarly extend the state variables $\rho_{\ell,t,2}$ to express the aggregate number of rows that are available at time t for the second stage demands.

Non-uniform row building durations. Instead of assuming a uniform row building time equal to the duration of the first stage T_1, let τ_ℓ denote the time required to build a new row in data center ℓ. Depending on the value of τ_ℓ, in this setting new rows may become available during either stage.

Updated formulation. Following these changes, the row availability constraints (5.7) - (5.9) in the first stage problem are now introduced for each time step t, and they are updated as follows:

$$\rho_{\ell,t,1} \geq \sum_{d \in \mathcal{D}^1 : \ell \in \mathcal{L}(d)} \sum_{t' \in \mathcal{T}_1 : t' \leq t} z_{d,\ell,t'} \qquad \forall \ell \in \mathcal{L}, t \in \mathcal{T}_1 : t < \tau_\ell \tag{C.13}$$

$$\rho_{\ell,t,1} + x_\ell \geq \sum_{d \in \mathcal{D}^1 : \ell \in \mathcal{L}(d)} \sum_{t' \in \mathcal{T}_1 : t' \leq t} z_{d,\ell,t'} \qquad \forall \ell \in \mathcal{L}, t \in \mathcal{T}_1 : t \geq \tau_\ell \tag{C.14}$$

$$\rho_{\ell,t,2} = \rho_{\ell,t,1} - \sum_{d \in \mathcal{D}^1 : \ell \in \mathcal{L}(d)} \sum_{t' \in \mathcal{T} : t' \leq t} z_{d,\ell,t'} \qquad \forall \ell \in \mathcal{L}, t \in \mathcal{T}_2 : t < \tau_\ell \tag{C.15}$$

$$\rho_{\ell,t,2} = \rho_{\ell,t,1} + x_\ell - \sum_{d \in \mathcal{D}^1 : \ell \in \mathcal{L}(d)} \sum_{t' \in \mathcal{T} : t' \leq t} z_{d,\ell,t'} \qquad \forall \ell \in \mathcal{L}, t \in \mathcal{T}_2 : t \geq \tau_\ell \tag{C.16}$$

$$\zeta_\ell \geq \rho_{\ell,T,1} + x_\ell \qquad \forall \ell \in \mathcal{L} \tag{C.17}$$

$$\rho_{\ell,t,2} \geq 0 \qquad \forall \ell \in \mathcal{L}, t \in \mathcal{T} \tag{C.18}$$

Constraints (C.13) and (C.14) ensure that demands that dock during $\mathcal{T}_1$ can only use rows that are available at that time; note that we now allow using new rows during $\mathcal{T}_1$ as long as their building is complete (C.14). Constraints (C.15) and (C.16) enforce similar limitations during $\mathcal{T}_2$ and make any unused rows available for the stochastic demands. Constraints (C.17) capture the data center capacity limits, assuming $\rho_{\ell,T,1}$ is the total number of existing rows whose building was already planned. The second stage problem constraints (5.17) are updated similarly to account for rows becoming available over the course of $\mathcal{T}_2$:

$$\rho_{\ell,t,2} \geq \sum_{d \in \mathcal{D}^1} \sum_{t' \in \mathcal{T}_2 : t' \leq t} z_{d,\ell,t'} \qquad \forall \ell \in \mathcal{L}, t \in \mathcal{T}_2 \tag{C.19}$$

Note that if for some times $t', \ldots, t'+i$ the number of available rows $\rho_{\ell,t,2}$ remains the same, we only need to introduce constraint (C.19) for the last step $t'+i$ of this range. Similarly, we can skip some time steps in constraints (C.13)-(C.16) (and the corresponding variables) if the number of existing rows will remain the same in the next time step and no new rows may become available at that time. In the above formulation, we introduced these constraints for each time step in order to maintain a cleaner description. It is straightforward to extend the network construction and the results from Section 5.5 for this more general formulation. In particular, the data center nodes in the network now need to be time-dependent, so we introduce a node for each (ℓ, t)-pair and update the edges accordingly.

Bibliography

Abbasi-Yadkori, Y., Bartlett, P. L., and Szepesvári, C. (2013). Online learning in markov decision processes with adversarially chosen transition probability distributions. *CoRR*, abs/1303.3055.

Agrawal, S. and Jia, R. (2022). Learning in structured mdps with convex cost functions: Improved regret bounds for inventory management. *Operations Research*, 70.

Ahmed, S., King, A. J., and Parija, G. (2003). A multi-stage stochastic integer programming approach for capacity expansion under uncertainty. *Journal of Global Optimization*, 26(1):3–24.

Arbabian, M. E., Chen, S., and Moinzadeh, K. (2021). Capacity expansions with bundled supplies of attributes: An application to server procurement in cloud computing. *Manufacturing & Service Operations Management*, 23(1):191–209.

Ardestani-Jaafari, A. and Delage, E. (2016). Robust optimization of sums of piecewise linear functions with application to inventory problems. *Operations Research*, 64(2):474–494.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., Stoica, I., et al. (2009). Above the clouds: A berkeley view of cloud computing. Technical report, Technical Report UCB/EECS-2009-28, EECS Department, University of California ....

Artzner, P., Delbaen, F., Eber, J.-M., and Heath, D. (1999). Coherent measures of risk. *Mathematical Finance*, 9(3):203–228.

Atamtürk, A. and Zhang, M. (2007). Two-stage robust network flow and design under demand uncertainty. *Operations Research*, 55(4):662–673.

Aviv, Y. and Federgruen, A. (1997). Stochastic inventory models with limited production capacity and periodically varying parameters. *Probability in the Engineering and Informational Sciences*, 11(1):107–135.

Baek, J. and Ma, W. (2019). Bifurcating constraints to improve approximation ratios for network revenue management with reusable resources. *Available at SSRN*.

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

Balseiro, S. R., Golrezaei, N., Mahdian, M., Mirrokni, V. S., and Schneider, J. (2018). Contextual bandits with cross-learning. *CoRR*, abs/1809.09582.

Bean, J. C., Higle, J. L., and Smith, R. L. (1992). Capacity expansion under stochastic demands. *Operations Research*, 40(3-supplement-2):S210–S216.

Ben-Tal, A., Golany, B., Nemirovski, A., and Vial, J.-P. (2005). Retailer-supplier flexible commitments contracts: A robust optimization approach. *Manufacturing & Service Operations Management*, 7(3):248–271.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252.

Berbeglia, G., Garassino, A., and Vulcano, G. (2018). A comparative empirical study of discrete choice models in retail operations. *Working Paper*.

Bernstein, F., Kök, A. G., and Xie, L. (2015). Dynamic assortment customization with limited inventories. *Manufacturing & Service Operations Management*, 17(4):538–553.

Bertsimas, D., Nasrabadi, E., and Stiller, S. (2013). Robust and adaptive network flows. *Operations Research*, 61(5):1218–1242.

Bertsimas, D. and Sim, M. (2003). Sim, m.: Robust discrete optimization and network flows. math. prog. 98, 49-71. *Mathematical Programming*, 98:49–71.

Bertsimas, D. and Thiele, A. (2006). A robust optimization approach to inventory theory. *Operations Research*, 54:150–168.

Bienstock, D. and Özbay, N. (2008). Computing robust basestock levels. *Discrete Optimization*, 5(2):389–414. In Memory of George B. Dantzig.

Birge, J. R. and Louveaux, F. (2011). *Introduction to stochastic programming.* Springer Science & Business Media.

Blanchet, J. H., Gallego, G., and Goyal, V. (2016). A Markov chain approximation to choice modeling. *Operations Research*, 64(4):886–905.

Buchbinder, N., Fairstein, Y., Mellou, K., Menache, I., and Naor, J. (2021). Online virtual machine allocation with lifetime and load predictions. In *Abstract Proceedings of the 2021 ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*, pages 9–10.

Budhiraja, A., Mukherjee, D., Wu, R., et al. (2019). Supermarket model on graphs. *The Annals of Applied Probability*, 29(3):1740–1777.

Chan, C. W. and Farias, V. F. (2009). Stochastic depletion problems: Effective myopic policies for a class of dynamic optimization problems. *Mathematics of Operations Research*, 34(2):333–350.

Chao, X., Chen, H., and Zheng, S. (2009). Dynamic capacity expansion for a service firm with capacity deterioration and supply uncertainty. *Operations research*, 57(1):82–93.

Chatwin, R. E. (1998). Multiperiod airline overbooking with a single fare class. *Operations Research*, 46(6):805–819.

Chen, B. (2019). Data-driven inventory control with shifting demand. *Available at SSRN 3503139.*

Chen, B. and Shi, C. (2019). Tailored base-surge policies in dual-sourcing inventory systems with demand learning. *Available at SSRN 3456834.*

Chen, L., Kyng, R., Liu, Y. P., Peng, R., Gutenberg, M. P., and Sachdeva, S. (2022). Maximum flow and minimum-cost flow in almost-linear time. *CoRR*, abs/2203.00671.

Chen, S., Moinzadeh, K., Song, J.-S. J., and Zhong, Y. (2020). Cloud computing value chains: Research from the operations management perspective. *Available at SSRN.*

Chen, Y., Levi, R., and Shi, C. (2017). Revenue management of reusable resources with advanced reservations. *Production and Operations Management*, 26(5):836–859.

Cheung, W. C., Simchi-Levi, D., and Zhu, R. (2020). Reinforcement learning for non-stationary markov decision processes: The blessing of (more) optimism. *CoRR*, abs/2006.14389.

Cornuéjols, G. (2001). *Combinatorial Optimization: Packing and Covering.* SIAM.

Cruise, J., Jonckheere, M., and Shneer, S. (2020). Stability of jsq in queues with general server-job class compatibilities. *Queueing Systems*, 95:271–279.

Dann, C., Mansour, Y., Mohri, M., Sekhari, A., and Sridharan, K. (2020). Reinforcement learning with feedback graphs.

Davis, J. M., Gallego, G., and Topaloglu, H. (2014). Assortment optimization under variants of the nested logit model. *Operations Research*, 62(2):250–273.

Davoodi, M., Katehakis, M. N., and Yang, J. (2019). Dynamic inventory control with fixed setup costs and unknown discrete demand distribution. *Available at SSRN 3435085*.

Devanur, N. R., Jain, K., and Kleinberg, R. D. (2013). Randomized primal-dual analysis of ranking for online bipartite matching. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '13, page 101–107, USA. Society for Industrial and Applied Mathematics.

Dickerson, J. P., Sankararaman, K. A., Srinivasan, A., and Xu, P. (2018). Allocation problems in ride-sharing platforms: Online matching with offline reusable resources. In McIlraith, S. A. and Weinberger, K. Q., editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), February 2-7, 2018*, pages 1007–1014. AAAI Press.

Dillon, M., Oliveira, F., and Abbasi, B. (2017). A two-stage stochastic programming model for inventory management in the blood supply chain. *International Journal of Production Economics*, 187:27–41.

Dong, S., Roy, B. V., and Zhou, Z. (2019). Provably efficient reinforcement learning with aggregated states.

Ehrenthal, J., Honhon, D., and Woensel, T. V. (2014). Demand seasonality in retail inventory management. *European Journal of Operational Research*, 238(2):527 – 539.

Eppen, G. D., Martin, R. K., and Schrage, L. (1989). Or practice—a scenario approach to capacity planning. *Operations research*, 37(4):517–527.

Feng, Y., Niazadeh, R., and Saberi, A. (2019). Linear programming based online policies for real-time assortment of reusable resources. *Available at SSRN 3421227*.

Feng, Y., Niazadeh, R., and Saberi, A. (2020). Near-optimal bayesian online assortment of reusable resources. *Available at SSRN: https://ssrn.com/abstract=3714338*.

Gallego, G., Iyengar, G., Phillips, R., and Dubey, A. (2004). Managing flexible products on a network. *CORC Technical Report TR-2004-01*.

Gallego, G. and Topaloglu, H. (2014). Constrained assortment optimization for the nested logit model. *Management Science*, 60(10):2583–2601.

Gartner (2021). Gartner forecasts worldwide public cloud end-user spending to grow 23% in 2021. `https://www.gartner.com/en/newsroom/press-releases/2021-04-21-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-grow-23-percent-in-2021`. Accessed: 2021-11-15.

Golrezaei, N., Nazerzadeh, H., and Rusmevichientong, P. (2014). Real-time optimization of personalized assortments. *Management Science*, 60(6):1532–1551.

Gong, X.-Y., Goyal, V., Iyengar, G. N., Simchi-Levi, D., Udwani, R., and Wang, S. (2022). Online assortment optimization with reusable resources. *Management Science*, 68(7):4772–4785.

Gong, X.-Y. and Simchi-Levi, D. (2021). Bandits atop reinforcement learning: Tackling online inventory models with cyclic demands. *Available at SSRN 3637705*.

Gorissen, B. and den Hertog, D. (2011). Robust counterparts of inequalities containing sums of maxima of linear functions. *European Journal of Operational Research*, 227.

Goyal, V., Iyengar, G., and Udwani, R. (2020a). Online allocation of reusable resources: Achieving optimal competitive ratio. *arXiv preprint arXiv:2002.02430*.

Goyal, V., Iyengar, G., and Udwani, R. (2020b). Online allocation of reusable resources via algorithms guided by fluid approximations.

Goyal, V., Iyengar, G., and Udwani, R. (2021). Asymptotically optimal competitive ratio for online allocation of reusable resources.

Goyal, V. and Udwani, R. (2020). Online matching with stochastic rewards: Optimal competitive ratio via path based formulation. In *Proceedings of the 21st ACM Conference on Economics and Computation*, EC '20, page 791, New York, NY, USA. Association for Computing Machinery.

Hadary, O., Marshall, L., Menache, I., Pan, A., Greeff, E. E., Dion, D., Dorminey, S., Joshi, S., Chen, Y., Russinovich, M., et al. (2020). Protean:{VM} allocation service at scale. In *14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20)*, pages 845–861.

Huh, W. T., Janakiraman, G., Muckstadt, J. A., and Rusmevichientong, P. (2009). An adaptive algorithm for finding the optimal base-stock policy in lost sales inventory systems with censored demand. *Mathematics of Operations Research*, 34(2):397–416.

Huh, W. T., Levi, R., Rusmevichientong, P., and Orlin, J. B. (2011). Adaptive data-driven inventory control with censored demand based on kaplan-meier estimator. *Operations Research*, 59(4):929–941.

Huh, W. T. and Rusmevichientong, P. (2009a). A nonparametric asymptotic analysis of inventory planning with censored demand. *Mathematics of Operations Research*, 34(1):103–123.

Huh, W. T. and Rusmevichientong, P. (2009b). A nonparametric asymptotic analysis of inventory planning with censored demand. *Mathematics of Operations Research*, 34(1):103–123.

Huh, W. T. and Rusmevichientong, P. (2014). Online sequential optimization with biased gradients: Theory and applications to censored demand. *INFORMS Journal on Computing*, 26(1):150–159.

Håstad, J. (1999). Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182(1):105 – 142.

Jin, C., Allen-Zhu, Z., Bubeck, S., and Jordan, M. I. (2018). Is Q-learning provably efficient? In *Advances in Neural Information Processing Systems*, pages 4863–4873.

Kaggle (2015). Rossmann store sales.

Karlin, S. (1960). Optimal policy for dynamic inventory process with stochastic demands subject to seasonal variations. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):611–629.

Karp, R. M., Vazirani, U. V., and Vazirani, V. V. (1990). An optimal algorithm for on-line bipartite matching. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, STOC '90, page 352–358, New York, NY, USA. Association for Computing Machinery.

Kleywegt, A., Shapiro, A., and Homem-de Mello, T. (2002). The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502.

Kök, A., Fisher, M., and Vaidyanathan, R. (2015). Assortment planning: Review of literature and industry practice. In Agrawal, N. and Smith, S. A., editors, *Retail Supply Chain Management*, volume 223 of *International Series in Operations Research & Management Science*, pages 175–236. Springer US.

Korte, B. and Vygen, J. (2018). *Combinatorial Optimization: Theory and Algorithms*. Springer Berlin Heidelberg, sixth edition.

LaFleur, K. (2020). The pandemic forced businesses to move online — it's time for fundraising to also go digital. `https://www.forbes.com/sites/forbestechcouncil/2020/12/14/the-pandemic-forced-businesses-to-move-online---its-time-for-fundraising-to-also-go-digital/?sh=773f52d15232`. Accessed: 2021-11-15.

Lemaréchal, C., Nemirovskii, A., and Nesterov, Y. (1995). New variants of bundle methods. *Mathematical Programming*, 69(1):111–147.

Levi, R. and Radovanović, A. (2010). Provably near-optimal lp-based policies for revenue management in systems with reusable resources. *Operations Research*, 58(2):503–507.

Lim, V. (2016). How poor inventory management ruined Target Canada.

Liu, Q. and van Ryzin, G. J. (2008). On the choice-based linear programming model for network revenue management. *Manufacturing & Service Operations Management*, 10(2):288–310.

Luce, R. D. (1959). *Individual Choice Behavior: A Theoretical Analysis*. Wiley.

Luss, H. (1982). Operations research and capacity expansion problems: A survey. *Operations research*, 30(5):907–947.

Luxner, T. (2021). Cloud computing trends: 2021 state of the cloud report. `https://www.flexera.com/blog/cloud/cloud-computing-trends-2021-state-of-the-cloud-report/`. Accessed: 2021-11-15.

Ma, W. and Simchi-Levi, D. (2020). Algorithms for online matching, assortment, and pricing with tight weight-dependent competitive ratios. *Operations Research*, 68(6):1787–1803.

Maguluri, S. T., Srikant, R., and Ying, L. (2012). Stochastic models of load balancing and scheduling in cloud computing clusters. In *2012 Proceedings IEEE Infocom*, pages 702–710. IEEE.

Manne, A. S. (1967). *Investments for capacity expansion: size, location, and time-phasing*, volume 5. Mit Press.

Markowitz, H. (1952). Portfolio selection*. *The Journal of Finance*, 7(1):77–91.

McFadden, D. (1973). Conditional logit analysis of qualitative choice behavior. *in P. Zarembka, ed., Frontiers in Econometrics*.

McFadden, D. et al. (1978). *Modelling the choice of residential location*. Institute of Transportation Studies, University of California.

McFadden, D. and Train, K. (2000). Mixed MNL models for discrete response. *Journal of Applied Econometrics*, 15(5):447–470.

Mehta, A. et al. (2013). Online matching and ad allocation. *Foundations and Trends® in Theoretical Computer Science*, 8(4):265–368.

Mukherjee, D., Borst, S. C., and van Leeuwaarden, J. S. H. (2018). Asymptotically optimal load balancing topologies. *Proc. ACM Meas. Anal. Comput. Syst.*, 2(1):14:1–14:29.

Owen, Z. and Simchi-Levi, D. (2018). Price and assortment optimization for reusable resources. *Available at SSRN 3070625*.

Perakis, G. and Roels, G. (2008). Regret in the newsvendor model with partial information. *Operations Research*, 56(1):188–203.

Plackett, R. L. (1975). The analysis of permutations. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 24(2):193–202.

Porteus, E. (2002). *Foundations of Stochastic Inventory Theory*. Business/Marketing. Stanford University Press.

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Rockafellar, R. T. and Uryasev, S. (2000). Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42.

Rusmevichientong, P., Sumida, M., and Topaloglu, H. (2020). Dynamic assortment optimization for reusable products with random usage durations. *Management Science*, 66(7):2820–2844.

Schrijver, A. (2003). *Combinatorial Optimization: Polyhedra and Efficiency*. Springer Berlin Heidelberg.

Shapiro, A., Dentcheva, D., and Ruszczyński, A. (2014). *Lectures on stochastic programming: modeling and theory*. SIAM and MPS, Philadelphia, second edition.

Shapiro, A., Dentcheva, D., and Ruszczynski, A. (2021). *Lectures on stochastic programming: modeling and theory*. SIAM.

Sheopuri, A., Janakiraman, G., and Seshadri, S. (2010). New policies for the stochastic inventory control problem with two supply sources. *Operations Research*, 58(3):734–745.

Sidford, A., Wang, M., Wu, X., Yang, L. F., and Ye, Y. (2018). Near-optimal time and sample complexities for solving markov decision processes with a generative model. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, page 5192–5202, Red Hook, NY, USA. Curran Associates Inc.

Simchi-Levi, D., Wang, H., and Wei, Y. (2019). Constraint generation for two-stage robust network flow problems. *INFORMS Journal on Optimization*, 1(1):49–70.

Sinclair, S., Banerjee, S., and Yu, C. (2019). Adaptive discretization for episodic reinforcement learning in metric spaces. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3:1–44.

Slivkins, A. (2019). Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12(1-2):1–286.

Solyalı, O., Cordeau, J.-F., and Laporte, G. (2016). The impact of modeling on robust inventory management under demand uncertainty. *Management Science*, 62(4):1188–1201.

Song, J.-S., Xiao, L., Zhang, H., and Zipkin, P. (2017). Optimal policies for a dual-sourcing inventory problem with endogenous stochastic lead times. *Operations Research*, 65(2):379–395.

Song, J.-S., Xiao, L., Zhang, H., and Zipkin, P. (2021). Smart policies for multisource inventory systems and general tandem queues with order tracking and expediting. *Operations Research*, 0(0):null.

Song, J.-S. and Zipkin, P. (2009). Inventories with multiple supply sources and networks of queues with overflow bypasses. *Management Science*, 55(3):362–372.

Statista (2021). Total size of the public cloud computing market from 2008 to 2020. `https://www.statista.com/statistics/510350/worldwide-public-cloud-computing/`. Accessed: 2021-11-15.

Stein, C., Truong, V.-A., and Wang, X. (2018). Advance service reservations with heterogeneous customers. *arXiv preprint arXiv:1805.05554*.

Stolyar, A. L. and Zhong, Y. (2013). A large-scale service system with packing constraints: Minimizing the number of occupied servers. *ACM SIGMETRICS Performance Evaluation Review*, 41(1):41–52.

Sun, J. and Van Mieghem, J. A. (2019). Robust dual sourcing inventory management: Optimality of capped dual index policies and smoothing. *Manufacturing & Service Operations Management*, 21(4):912–931.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

Talluri, K. and Van Ryzin, G. (2004). Revenue management under a general discrete choice model of consumer behavior. *Management Science*, 50(1):15–33.

Topaloglu, H. (2013). Joint stocking and product offer decisions under the multinomial logit model. *Production and Operations Management*, 22(5):1182–1199.

Train, K. E. (2009). *Discrete Choice Methods with Simulation*. Cambridge University Press.

Van Mieghem, J. A. (2003). Commissioned paper: Capacity management, investment, and hedging: Review and recent developments. *Manufacturing & Service Operations Management*, 5(4):269–302.

Wang, X., Truong, V.-A., and Bank, D. (2018). Online advance admission scheduling for services with customer preferences. *arXiv preprint arXiv:1805.10412*.

Watkins, C. and Dayan, P. (1992). Technical note: Q-learning. *Machine Learning*, 8:279–292.

Weng, W., Zhou, X., and Srikant, R. (2020). Optimal load balancing with locality constraints. *Proc. ACM Meas. Anal. Comput. Syst.*, 4(3):45:1–45:37.

Williams, H. W. (1977). On the formation of travel demand models and economic evaluation measures of user benefit. *Environment and Planning A*, 3(9):285–344.

Williamson, D. P. and Shmoys, D. B. (2011). *The Design of Approximation Algorithms*. Cambridge University Press.

Xin, L. (2022). 1.79-approximation algorithms for continuous review single-sourcing lost-sales and dual-sourcing inventory models. *Operations Research*, 70(1):111–128.

Xiong, X., Li, Y., Yang, W., and Shen, H. (2022). Data-driven robust dual-sourcing inventory management under purchase price and demand uncertainties. *Transportation Research Part E: Logistics and Transportation Review*, 160:102671.

Yuan, H., Luo, Q., and Shi, C. (2021). Marrying stochastic gradient descent with bandits: Learning algorithms for inventory systems with fixed costs. *Management Science*.

Zhang, C., Kumbhare, A. G., Manousakis, I., Zhang, D., Misra, P. A., Assis, R., Woolcock, K., Mahalingam, N., Warrier, B., Gauthier, D., et al. (2021). Flex: High-availability datacenters with zero reserved power. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 319–332. IEEE.

Zhang, H., Chao, X., and Shi, C. (2020). Closing the gap: A learning algorithm for lost-sales inventory systems with lead times. *Management Science*, 66(5):1962–1980.

Zhao, H. and Chen, W. (2019). Stochastic one-sided full-information bandit. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD'2019)*.

Zipkin, P. (1989). Critical number policies for inventory models with periodic data. *Management Science*, 35(1):71–80.

Zipkin, P. (2000). *Foundations of Inventory Management*. McGraw-Hill Companies,Incorporated.